사이킷런 소개와 첫번째 머신러닝 애플리케이션 만들어 보기 - 붓꽃(Iris) 품종 예측

[사이킷런 소개]

- 파이썬 기반의 다른 머신러닝 패키지도 사이킷런 스타일의 API를 지향할 정도로 쉽고 가장 파이썬스러운 API를 제공
- 머신러닝을 위한 매우 다양한 알고리즘과 개발을 위한 편리한 프레임워크와 API를 제공
- 오랜 기간 실전 환경에서 검증됐으며, 매우 많은 환경에서 사용되는 성숙한 라이브러리
- 주로 Numpy와 Scipy 기반 위에서 구축된 라이브러리

[머신러닝을 위한 용어 정리]

- 피처(Feature)
  - 피처는 데이터 세트의 일반 속성
  - 머신러닝은 2차원 이상의 다차원 데이터에서도 많이 사용되므로 타겟값을 제외한 나머지 속성을 모두 피처로 지정
- 레이블, 클래스, 타겟(값), 결정(값)
  - 타겟값 또는 결정값은 지도 학습 시 데이터 학습을 위해 주어지는 정답 데이터
  - 지도 학습 중 분류의 경우에는 이 결정값을 레이블 또는 클래스로 지칭

[지도학습 - 분류]

- 분류(classification)
  - 대표적인 지도학습 (Supervised Learning) 방법의 하나
- 지도학습
  - 학습을 위한 다양한 피처와 분류 결정값인 레이블(label) 데이터로 모델을 학습한 뒤, 별도의 테스트 데이터 세트에서 미지의 레이블 예측
  - 명확한 정답이 주어진 데이터를 먼저 학습한 뒤 미지의 정답을 예측하는 방식
- 학습 데이터 세트
  - 학습을 위해 주어진 데이터 세트
- 테스트 데이터 세트
  - 머신러닝 모델의 예측 성능을 평가하기 위해 별도로 주어진 데이터 세트

[사이킷런을 이용한 붓꽃 데이터 분류]

붓꽃 데이터 세트로 붓꽃의 품종을 분류(Classification) 하는 모델

- 꽃잎의 길이와 너비, 꽃받침의 길이와 너비 피처(Feature)를 기반으로 꽃의 품종 예측

[붓꽃 데이터 분류 예측 프로세스]

1. 데이터 세트 분리: 데이터를 학습 데이터와 테스트 데이터로 분리

```
X_train, X_test, y_train, y_test

        = train_test_split(iris_data, iris_label, test_size=0.2, random_state=11)
```

관습적으로 X: Feature, y: 결정값/타겟값을 사용

2. 모델 학습: 학습 데이터를 기반으로 ML 알고리즘을 적용해 모델을 학습시킴

```
dt_clf.fit(X_train, y_train)
```

학습용 피처 데이터 세트에 대해서 학습용 데이터 세트를 매핑해서 학습 수행

3. 예측 수행: 학습된 ML 모델을 이용해 테스트 데이터의 분류(즉, 붓꽃 종류)를 예측

```
pred = dt_clf.predict(X_test)
```

인자는 피처 데이터 세트

테스트 데이터 세트로 예측을 수행하고, 예측 값 반환

4. 평가: 이렇게 예측된 결과값과 테스트 데이터의 실제 결과값을 비교해 ML 모델 성능을 평가

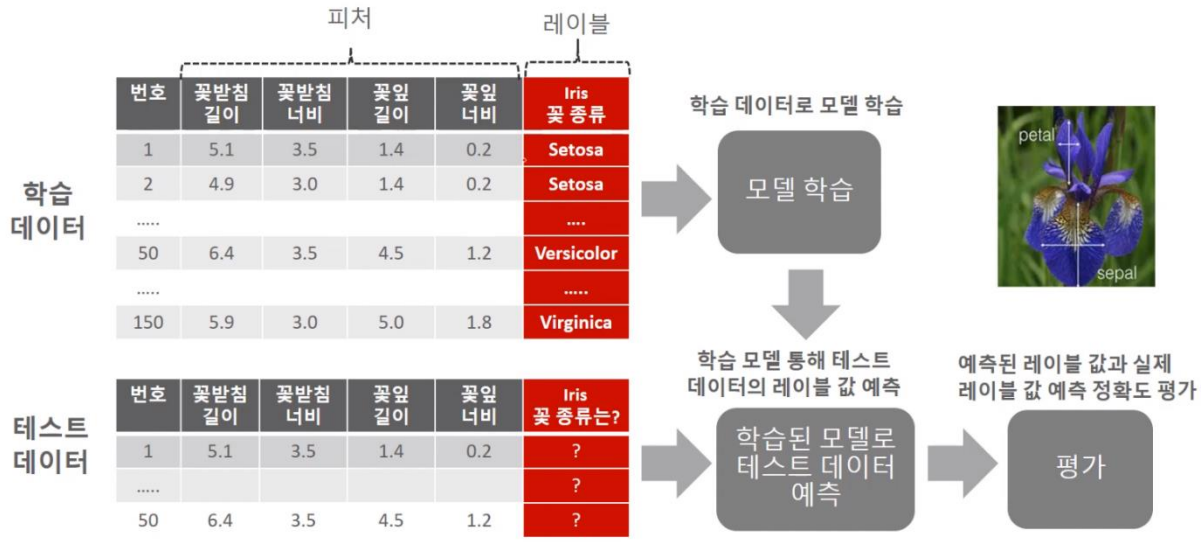
```
accuracy_score(y_test, pred)
```

정확도를 구할 때 사용.

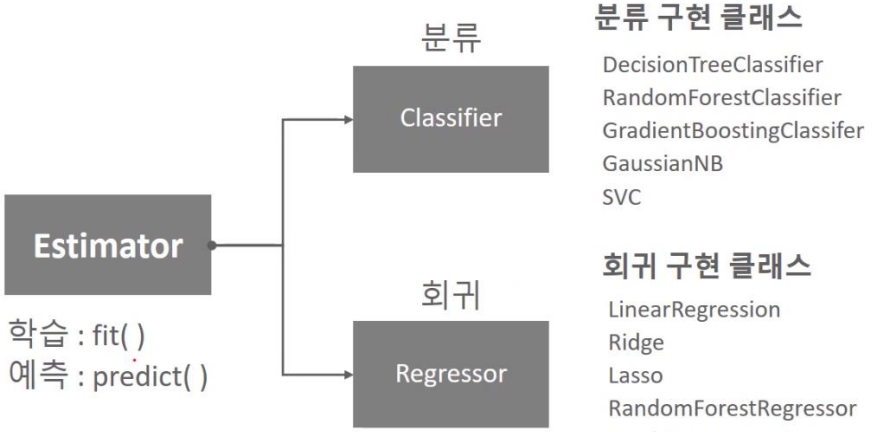
인자로 실제 타겟값과 예측한 타겟값을 넣으면 예측 정확도 반환

사이킷런의 기반 프레임 워크 익히기 - 주요 API/모듈 및 내장 예제 데이터 세트 소개

[붓꽃 데이터 분류 예측 프로세스]



[사이킷런 기반 프레임워크 - Estimator와 fit(), predict()]



분류 구현 클래스

DecisionTreeClassifier
RandomForestClassifier
GradientBoostingClassifer
GaussianNB
SVC

회귀 구현 클래스

LinearRegression
Ridge
Lasso
RandomForestRegressor
GradientBoostingRegressor

모든 구현 클래스를 통틀어서 사이킷런에서는 Estimator라고 부름

[사이킷런의 주요 모듈]

분류	모듈명	설명
예제 데이터	sklearn.datasets	사이킷런에 내장되어 예제로 제공하는 데이터 세트
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
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 우선순위 대로 셀렉션 작업을 수행하는 다양한 기능 제공
	sklearn.feature_extraction	텍스트 데이터나 이미지 데이터의 벡터화된 피처를 추출하는 데 사용됨. 예를 들어 텍스트 데이터에서 Count Vectorizer 나 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

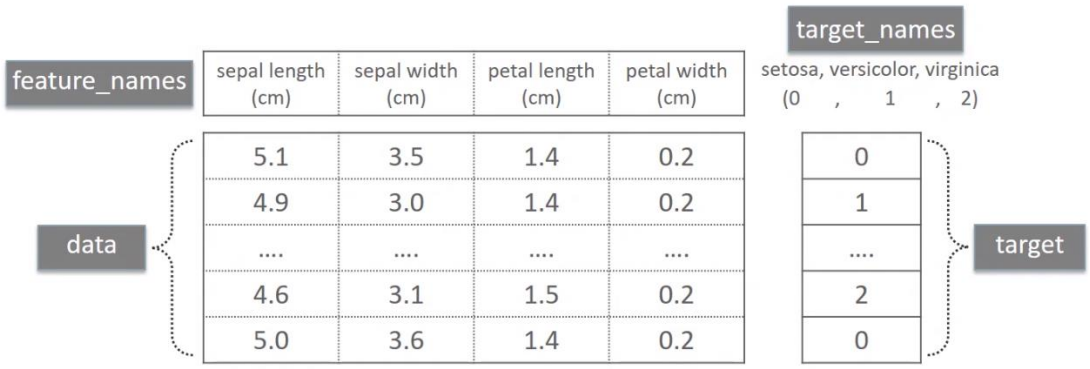
분류	모듈명	설명
평가	sklearn.metrics	분류, 회귀, 클러스터링, 페어와이즈(Pairwise)에 대한 다양한 성능 측정 방법 제공 Accuracy, Precision, Recall, ROC-AUC, RMSE 등 제공
ML 알고리즘	sklearn.ensemble	앙상블 알고리즘 제공 랜덤 포레스트, 에이다 부스트, 그래디언트 부스팅 등을 제공
	sklearn.linear_model	주로 선형 회귀, 릿지(Ridge), 라쏘(Lasso) 및 로지스틱 회귀 등 회귀 관련 알고리즘을 지원. 또한 SGD(Stochastic Gradient 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

[사이킷런 내장 예제 데이터 셋 - 분류 및 회귀용]

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

[내장 예제 데이터 셋 구성]

feature_names, data, target_names, target



- 키는 보통 data, target, target_name, feature_names, DESCR로 구성
  - data: 피처의 데이터 세트
  - target: 분류-레이블 값, 회귀-숫자 결과값 데이터 세트
  - target_names: 개별 레이블의 이름
  - feature_names: 피처 이름
  - DESCR: 데이터 세트에 대한 설명과 각 피처의 설명

학습과 테스트 데이터 세트의 분리

[Model Selection 소개 - 학습 데이터와 테스트 데이터]

- 학습 데이터 세트

  - 머신러닝 알고리즘의 학습을 위해 사용

  - 데이터의 속성들과 결정값(레이블)값 모두를 가지고 있음

  - 학습 데이터를 기반으로 머신러닝 알고리즘이 데이터 속성과 결정값의 패턴을 인지하고 학습

- 테스트 데이터 세트

  - 테스트 데이터 세트에서 학습된 머신러닝 알고리즘을 테스트

  - 테스트 데이터는 속성 데이터만 머신러닝 알고리즘에 제공하며, 머신러닝 알고리즘은 제공된 데이터를 기반으로 결정값을 예측

  - 테스트 데이터는 학습 데이터와 별도의 데이터 세트로 제공되어야 함

[학습 데이터와 테스트 데이터 분리 - train_test_split()]

- sklearn.model_selection의 train_test_split() 함수

```
X_train, X_test, y_train, y_test

  = train_test_split(iris_data.data, iris_data.target, test_size=0.3, random_state=121)
```

  - test_size: 전체 데이터에서 테스트 데이터 세트 크기를 얼마로 샘플링할 것인지 결정. 디폴트는 0.25, 즉 25%
  - train_size: 전체 데이터에서 학습용 데이터 세트 크기를 얼마로 샘플링 할 것인지 결정. test_size parameter를 통상적으로 사용하기 때문에 train_size는 잘 사용되지 않음
  - shuffle: 데이터를 분리하기 전에 데이터를 미리 섞을지 결정. 디폴트는 True. 데이터를 분산시켜서 좀 더 효율적인 학습 및 데이터 세트를 만드는데 사용됨
  - random_state: 호출할 때마다 동일한 학습/데이터 세트를 생성하기 위해 주어지는 난수 값. train_test_split()는 호출 시 무작위로 데이터를 분리하므로 random_state를 지정하지 않으면 수행할 때마다 다른 학습/테스트용 데이터 생성함

- 넘파이 ndarray 뿐만 아니라 판다스 DataFrame/Series도 train_test_split( )로 분할 가능

교차검증 - K-Fold와 Stratified K-Fold

[교차 검증]

- 학습 데이터 세트
  - 학습 데이터를 다시 분할하여 학습 데이터와 학습된 모델의 성능을 일차 평가하는 검증 데이터로 나눔.
  - 학습 데이터 ---분할-→ 학습 데이터 세트 | 검증 데이터 세트
- 테스트 데이터 세트
  - 모든 학습/검증 과정이 완료된 후 최종적으로 성능을 평가하기 위한 데이터 세트

[K 폴드 교차 검증]

- K개의 폴드 세트에 K번의 학습과 검증 평가 반복 수행 (K: 폴드 교차 검증 횟수)
- 교차 검증 최종 평가 = (검증 평가 전체의) 평균

- 일반 K 폴드
- Stratified K 폴드
  - 불균형한 분포도를 가진 레이블(결정 클래스) 데이터 집합을 위한 K 폴드 방식
  - 학습 데이터와 검증 데이터 세트가 가지는 레이블 분포도가 유사하게 검증데이터 추출

```
kfold = KFold(n_splits=5)
```

5개의 폴드 세트로 분리하는 K폴드 객체 생성 (기본은 3개 분리)

```
kfold.split(features) // features=iris.data
```

폴드 별 학습용, 검증용 테스트의 로우 인덱스를 array로 반환

```
skfold.split(features, label)
```

StratifiedKFold의 split( ) 호출 시 반드시 레이블 데이터 셋도 추가 입력 필요

교차검증 성능평가 cross_val_score()와 하이퍼 파라미터 튜닝을 위한 GridSearchCV

[교차 검증을 보다 간편하게 - cross_val_score()]

- KFold 클래스를 이용한 교차 검증 방법
    1. 폴드 세트 설정
    2. For 루프에서 반복적으로 학습/검증 데이터 추출 및 학습과 예측 수행
    3. 폴드 세트별로 예측 성능을 평균하여 최종 성능 평가

⇨ cross_val_score() 함수로 폴드 세트 추출, 학습/예측, 평가를 한 번에 수행

```
cross_val_score(estimator, X, y=None, scoring=None, cv=None, n_jobs=1, verbose=0, fit_params=None, pre_dispatch='2*n_jobs')
```

[GridSearchCV - 교차 검증과 최적 하이퍼 파라미터 튜닝을 한 번에]

사이킷런은 GridSearchCV를 이용해 Classifier나 Regressor 와 같은 알고리즘에 사용되는 하이퍼 파라미터를 순차적으로 입력하면서 편리하게 최적의 파라미터를 도출할 수 있는 방안을 제공

- GridSearchCV()
    - refit=True가 디폴트. True이면 가장 좋은 파라미터 설정으로 재 학습시킴
      refit=True인 객체가 fit() 수행하면 학습이 완료된 estimator를 내포하고 있으므로 predict()을 통해 예측도 가능
    - 결과는 cv_results_ 라는 딕셔너리로 저장
- best_estimator_
    - GridSearchCV의 refit으로 이미 학습이 된 estimator 반환
    - 이미 최적 하이퍼 파라미터로 학습이 됨

데이터 전처리 - 인코딩과 스케일링

[데이터 전처리(Preprocessing)]

- 데이터 클린징
- 결손값 처리(Null/NaN 처리)
- 데이터 인코딩(레이블, 원-핫 인코딩)
- 데이터 스케일링
- 이상치 제거
- Feature 선택, 추출 및 가공

[데이터 인코딩]

머신러닝 알고리즘은 문자열 데이터 속성 입력 받지 않음. 모든 데이터는 숫자형으로 표현되어야 함. 문자형 카테고리형 속성은 모두 숫자 값으로 변환/인코딩 되어야 함.

- 레이블 인코딩

원본 데이터

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

상품 분류를 레이블 인코딩 한 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

[TV, 냉장고, 전자레인지, 컴퓨터, 선풍기, 믹서] → [0, 1, 4, 5, 3, 2]

- 원-핫(One-Hot) 인코딩
  - 피처 값의 유형에 따라 새로운 피처를 추가해 고유값에 해당하는 컬럼에만 1을 표시하고 나머지 컬럼에는 0을 표시하는 방식

TV:0, 냉장고:1, 전자레인지:4, 컴퓨터:5, 선풍기: 3, 믹서:2 를 원-핫 인코딩

원본 데이터

상품 분류
TV
냉장고
전자렌지
컴퓨터
선풍기
선풍기
믹서
믹서

원-핫 인코딩

상품분류_TV	상품분류_냉장고	상품분류_믹서	상품분류_선풍기	상품분류_전자렌지	상품분류_컴퓨터
1	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	0	0	0

[사이킷런 - 원-핫 인코딩]

원본 데이터 -> 숫자로 인코딩 -> 원-핫 인코딩

[판다스 get_dummies()를 이용한 원-핫 인코딩]

pd.get_dummies(DataFrame)

[피처 스케일링]

- 표준화
  - 데이터의 피처 각각이 평균이 0, 분산이 1인 가우시안 정규 분포를 가진 값으로 변환
- 정규화
  - 서로 다른 피처의 크기를 통일하기 위해 크기를 변환

[사이킷런 피처 스케일링 지원]

- StandardScaler
  - 평균이 0이고, 분산이 1인 정규 분포 형태로 변환
- MinMaxScaler
  - 데이터값을 0과 1 사이의 범위 값으로 변환(음수 값이 있으면 -1에서 1값으로 변환)

사이킷런으로 수행하는 타이타닉 생존자 예측

[타이타닉 생존자 ML 예측 구현]

- 데이터 전처리
  - Null 처리
  - 불필요한 속성 제거
  - 인코딩 수행
- 모델 학습 및 검증/예측/평가
  - 결정트리, 랜덤포레스트, 로지스틱 회귀 학습 비교
  - K 폴드 교차 검증
  - cross_val_score()와 GridSearchCV() 수행

[2장 Summary]

머신러닝 지도 학습 프로세스

1. 데이터 전처리
   - -데이터 클린징
   - 결손값 처리(Null/NaN 처리)
   - 데이터 인코딩(레이블, 원-핫 인코딩)
   - 데이터 스케일링
   - 이상치 제거
   - Feature 선택, 추출 및 가공
2. 데이터 세트 분리
   - 학습 데이터 / 테스트 데이터 분리
3. 모델 학습 및 검증 평가
   - 알고리즘 학습
   - 교차검증 | cross_val_score() | GridSearchCV
4. 예측 수행
   - 테스트 데이터로 예측 수행
5. 평가 – 예측 평가