지도학습 ( 분류 / 회귀 / 추천 시스템 / 시각 / 음성 / NLP )

비지도학습 ( 클러스터링 / 차원 축소 / 강화학습 )

# 1장

· ndarray의 데이터 타입

list3 = [1.2, 3.0]

array3 = np.array(list3)

print(array3, array3.dtype)

⇒ [1.2.3.] float64

list2 = [1, 2, 'test']

array2 = np.array(list2)

print(array2, array2.dtype)

⇒ ndarray는 데이터 값이 모두 같은 데이터 타입이어야 한다.

array_int = np.array([1,2,3])

array_float = array_int.astype('float64')

print(array_float, array_float.dtype)

⇒ [1.2.3.] float64

astype( )을 이용하면 원하는 타입변경 가능.

zeros( )는 모든 값을 0으로 채운 해당 shape을 가진 ndarray를 반환.

cf) ones( )는 1로 반환.

zero_array = np.zeros((3,2), dtype='int32')

(행, 열)

default: int64

print(zero_array)

print(zero_array.dtype, zero_array.shape)

⇒ [[0 0]
[0 0]
[0 0]]

int32 (3,2)

· reshape을 이용하여 차원크기 변경가능

↳ 사이즈 크게 주의

· reshape 에서 -1 은 ndarray와 호환되도록 자동 일을 해준다.

⇒ reshape(-1,1)은 항상 2차원.

인덱싱/슬라이싱

array1 = np.arange(start=1, stop=10) ⇒ [1,2, ... 9]

array4 = array1[:3]

print(array4)

⇒ [1 2 3]

[1 2 3 4 .. 9]
0 1 2 3

array2d[0:2, 0:2]

array2d[1:3, 0:3]

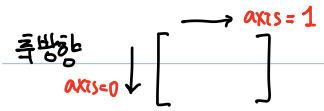
array2d[[0:1], 2]

array2d[[0,1]]

(행 = row, 열 = column)

오름차순 정렬: np.sort( ), ndarray.sort( )

내림차순 정렬: np.sort( )[::-1]




정렬된 행렬의 기존 원본 행렬의 원소에 대한 인덱스: np.argsort()

```
org_array = np.array([3,1,9,5])
                     0 1 2 3
sort_indices = np.argsort(org_array)
                     1 0 3 2
print(sort_indices)
```

⇒ [1032]

행렬내적 (행렬의 곱): np.dot  전치행렬: np.transpose()

```
A=np.array([[1,2,3],
            [4,5,6]])

B=np.array([[7,8],
            [9,10],
            [11,12]])

dot_product = np.dot(A,B)

print(dot_product)
```

⇒ [[ 58 64]
[139 154]]

Pandas 이용

- pd.read_csv( 경로 ) : DataFrame으로 파일 로드
- DataFrame.head(n) : 위의 n개의 로우를 반환.
- DataFrame.shape : 행,열을 튜플 형태로 반환.
- pf.info() : 총 데이터 수, 데이터 타입, Null 건수
- pf.describe() : 요약통계량
- value_counts() : 지정된 칼럼의 데이터값 건수를 반환.

```
value_counts = titanic_df['Pclass'].value_counts()
print(value_counts)
```

⇒ 3 491
1 216
2 184
Name: Pclass, dtype: int64

iloc[], loc[] 연산자

- data_df_rest.iloc[0,1]
⇒ 'Chulmin'
- data_df.loc['one','Name']
⇒ 'Chulmin'

→ iloc[]은 불린 인덱싱 X
loc[]은 명칭기반

<정렬, Aggregation, GroupBy>

titanic_sorted = titanic_df.sort_values(by=['Name'])

titanic_sorted = titanic_df.sort_values(by=['Pclass','Name'], ascending=False)

titanic_df.count()
- min(), max(), sum()

titanic_groupby = titanic_df.groupby(by='Pclass')

titanic_df.groupby('Pclass')['Age'].agg([max, min]) ⇒ Pclass 기준 분류, Age에 대한 Max, min 만을 결과로

apply lambda 식으로 데이터 가공

```
a=[1,2,3]
squares=map(lambda x: x**2, a)
list(squares)
```

⇒ [1, 4, 9]

## 2장 - 군도참고.

사이킷런.

교차검증: 학습 데이터 set → 분할 → 학습 데이터 set, 검증 데이터 set. / 테스트 데이터 set.

K 폴드 교차 검증: K개의 데이터 폴드 set → K번만큼 각 폴드 set에 학습과 검증을 반복.

Stratified K 폴드: 불균형한 분포를 가진 레이블에 이용.

데이터 전처리(Data Preprocessing)

- LabelEncoder : 객체 생성 후, fit(), transform()으로 레이블 인코딩 수행
- One Hot Encoder : 새로운 피처를 추가하여 고유값에 해당하는 칼럼에만 1을 표시, 나머지는 0을 표시.
- Standard Scaler : 표준화. 개별 피처를 평균이 0, 분산이 1인 값으로 변환.
- Min Max Scaler : 데이터 값을 0과 1 사이의 값으로 변환.

## 3장 - 평가.

정확도: 예측 결과가 동일한 데이터 건수 / 전체 예측 데이터 건수

성능평가 지표
- 정확도
- 오차행렬
- 정밀도
- 재현율
- F1 스코어
- ROC AUC

1) 정확도 (Base Estimator)

| | | Predicted Class: Negative(0) | Predicted Class: Positive(1) |
|---|---|---|---|
| Actual Class | Negative(0) | TN (True negative) | FP (False positive) |
| | Positive(1) | FN (False negative) | TP (True positive) |

2) 혼동행렬 (confusion matrix)

3) 정확도 : (TN + TP) / (전체)

정밀도 : TP / (FP + TP)

재현율 : TP / (FN + TP)

4) F1 score (import f1_score)

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

5) ROC 곡선 / AUC

ROC 곡선 : FPR이 변할 때 TPR이 어떻게 변하는지..

FPR = FP / (FP+TN) = 1 - TNR = 1 - 특이성