

7/27 수업

2020년 7월 27일 월요일 오전 8:48

■ 11기 2020년 7월 다섯번째주 수업 상세계획

월(7월 27일)	화(7월 28일)	수(7월 29일)	목(7월 30일)	금(7월 31일)
머신러닝 (의사결정, 회귀)	머신러닝 (k-means, 마프리오리 등)	CNN 1장 CNN 2장	CNN 2장	휴강

- 파이썬을 이용한 머신러닝 수업 목표 :
" 파이썬의 머신러닝 스크립트로 KAGGLE 상위권 1%에 도전 "
우리 이력서에 추가 할 수 있도록
- 딥러닝 수업의 목표 : 딥러닝 개발자 / 연구원으로 취업
 - 지난번에 스크롤링한 사진을 분류하는 신경망 (홈페이지로 구현)
(유니크한 포트폴리오 : 박지성과 유해진 사진 분류 신경망)
 - 홈페이지 url을 이력서에 제출

✂ 판다스를 이용한 머신러닝 수업

- 1장 : 판다스 데이터 프레임과 시리즈
- 2장 : 외부 데이터 파일을 파이썬으로 불러오는 방법
- 3장 : 판다스 데이터 프레임 기본 활용 (통계, 간단한 시각화)
- 4장 : 데이터 살펴보기 (시각화 : matplotlib, seaborn)
- 5장 : 데이터 전처리 1
- 6장 : 데이터 전처리 2
- 7장 : 파이썬으로 머신러닝 구현하기 (ncs 평가 = KAGGLE 순위)
 - knn (유방암, 타이타닉)
 - naivebayes (유방암, 타이타닉)
 - Decision Tree
 - Random Forest
 - 회귀분석 (단순회귀, 다중회귀)
 - 로지스틱 회귀

✂ 파이썬으로 Decision Tree 구현하기

Decision tree 는 의사결정 나무 라는 뜻 입니다.

컴퓨터 알고리즘에서 즐겨 사용하는 알고리즘이 Tree 구조인데,

이 tree 구조를 사용하고 각 분기점(node) 에는 분석 대상의 속성(설명변수)들이 위치한다.

각 분기점마다 목표값을 가장 잘 분류 할 수 있는 속성을 찾아서 배치하고 해당 속성이 갖는 값을 이용하여 새로운 가지를 만든다.

각 분기점에서 최적의 속성을 선택 할 때는 해당 속성을 기준으로 분류한 값들이 구분되는 정도를 측정한다.

이 측정 척도가 바로 "엔트로피(entropy)" 이다.

예제 : 유방암 데이터의 악성 종양과 양성 종양을 예측하는 의사결정 머신러닝 모델

% 머신러닝 데이터 분석 순서 %

1. 데이터 불러오기
2. 데이터 살펴보기 (데이터 전처리)
3. 데이터 정규화

4. 훈련 데이터와 테스트 데이터를 나눈다.
5. 훈련 데이터로 의사결정트리 모델을 생성한다.
6. 테스트 데이터로 모델의 성능을 확인한다.
7. 모델의 성능을 향상시킨다.

```
# -*- coding: utf-8 -*-
```

```
### 기본 라이브러리 불러오기
```

```
import pandas as pd
import numpy as np
```

```
'''
```

[Step 1] 데이터 준비/ 기본 설정

```
'''
```

```
# Breast Cancer 데이터셋 가져오기 (출처: UCI ML Repository)
```

```
uci_path = 'https://archive.ics.uci.edu/ml/machine-learning-databases/\
breast-cancer-wisconsin/breast-cancer-wisconsin.data'
```

```
df = pd.read_csv(uci_path, header=None)
```

```
# 열 이름 지정
```

```
df.columns = ['id', 'clump', 'cell_size', 'cell_shape', 'adhesion', 'epithelial',
              'bare_nuclei', 'chromatin', 'normal_nucleoli', 'mitoses', 'class']
```

```
# 컬럼 설명: 유방 종양의 크기와 거칠기등에 대한 전자수치 정보
```

	id	clump	cell_size	cell_shape	adhesion	epithelial	bare_nuclei	chromatin	normal_nucleoli	mitoses	class
0	1000025	5	1	1	1	2	1	3	1	1	2
1	1002945	5	4	4	5	7	10	3	2	1	2
2	1015425	3	1	1	1	2	2	3	1	1	2
3	1016277	6	8	8	1	3	4	3	7	1	2
4	1017023	4	1	1	3	2	1	3	1	1	2

label

```
# IPython 디스플레이 설정 - 출력할 열의 개수 한도 늘리기
```

```
pd.set_option('display.max_columns', 15)
```

```
'''
```

[Step 2] 데이터 탐색

```
'''
```

```
# 데이터 살펴보기
```

```
print(df.head())
```

```
print('\n')
```

```
# 데이터 자료형 확인
```

```
print(df.info())
```

```
print('\n')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699 entries, 0 to 698
Data columns (total 11 columns):
#   Column      Non-Null Count  Dtype
---  -
0    id          699 non-null    int64
1    clump       699 non-null    int64
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699 entries, 0 to 698
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  ---
0    id                    699 non-null    int64
1    clump                 699 non-null    int64
2    cell_size             699 non-null    int64
3    cell_shape            699 non-null    int64
4    adhesion              699 non-null    int64
5    epithelial            699 non-null    int64
6    bare_nuclei           699 non-null    object
7    chromatin              699 non-null    int64
8    normal_nucleoli       699 non-null    int64
9    mitoses               699 non-null    int64
10   class                 699 non-null    int64
dtypes: int64(10), object(1)
memory usage: 60.2+ KB
None
```

bare_nuclei 컬럼만 object이다. 이 컬럼의 데이터 타입을 int(숫자형)으로 변환 해줘야 한다.

```
# 데이터 통계 요약정보 확인
print(df.describe())
print('\n')
```

	id	clump	cell_size	cell_shape	adhesion \
count	6.990000e+02	699.000000	699.000000	699.000000	699.000000
mean	1.071704e+06	4.417740	3.134478	3.207439	2.806867
std	6.170957e+05	2.815741	3.051459	2.971913	2.855379
min	6.163400e+04	1.000000	1.000000	1.000000	1.000000
25%	8.706885e+05	2.000000	1.000000	1.000000	1.000000
50%	1.171710e+06	4.000000	1.000000	1.000000	1.000000
75%	1.238298e+06	6.000000	5.000000	5.000000	4.000000
max	1.345435e+07	10.000000	10.000000	10.000000	10.000000

	epithelial	chromatin	normal_nucleoli	mitoses	class
count	699.000000	699.000000	699.000000	699.000000	699.000000
mean	3.216023	3.437768	2.866953	1.589413	2.689557
std	2.214300	2.438364	3.053634	1.715078	0.951273
min	1.000000	1.000000	1.000000	1.000000	2.000000
25%	2.000000	2.000000	1.000000	1.000000	2.000000
50%	2.000000	3.000000	1.000000	1.000000	2.000000
75%	4.000000	5.000000	4.000000	1.000000	4.000000
max	10.000000	10.000000	10.000000	10.000000	4.000000

bare_nuclei 열의 자료형 변경 (문자열 ->숫자)

```
print(df['bare_nuclei'].unique())
print('\n')
```

bare_nuclei 열의 고유값 확인

```
['1' '10' '2' '4' '3' '9' '7' '?' '5' '8' '6']
```

```
df['bare_nuclei'].replace('?', np.nan, inplace=True)
df.dropna(subset=['bare_nuclei'], axis=0, inplace=True)
df['bare_nuclei'] = df['bare_nuclei'].astype('int')
```

'?'을 np.nan으로 변경

누락데이터 행을 삭제

문자열을 정수형으로 변환

```
print(df.info())
print('\n')
```

데이터 자료형 확인

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 683 entries, 0 to 698
Data columns (total 11 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   id               683 non-null    int64
1   clump            683 non-null    int64
2   cell_size        683 non-null    int64
3   cell_shape       683 non-null    int64
4   adhesion         683 non-null    int64
5   epithlial        683 non-null    int64
6   bare_nuclei      683 non-null    int32
7   chromatin        683 non-null    int64
8   normal_nucleoli  683 non-null    int64
9   mitoses          683 non-null    int64
10  class            683 non-null    int64
dtypes: int32(1), int64(10)
memory usage: 61.4 KB
None
```

```
'''
```

[Step 3] 데이터셋 구분 - 훈련용(train data)/ 검증용(test data)

```
'''
```

id 컬럼은 필요없으므로 제외시키고 다른 훈련할 때 필요한 컬럼들을 선별한다.

속성(변수) 선택

```
X=df[['clump','cell_size','cell_shape', 'adhesion','epithlial',
      'bare_nuclei','chromatin','normal_nucleoli', 'mitoses']] #설명 변수 X
y=df['class'] #예측 변수 Y
```

설명 변수 데이터를 정규화

```
from sklearn import preprocessing
X = preprocessing.StandardScaler().fit(X).transform(X)
```

pd.DataFrame(X).describe()

numpy를 dataframe으로 만들어줘서 통계 요약정보를 확인한다.

	0	1	2	3	4	5	6	7	8
count	6.830000e+02	6.830000e+02	6.830000e+02	6.830000e+02	6.830000e+02	6.830000e+02	6.830000e+02	6.830000e+02	6.830000e+02
mean	2.813757e-16	-2.899909e-16	-4.431139e-16	2.144047e-16	-5.439768e-16	-1.563415e-15	-2.223697e-16	-7.737425e-17	1.050404e-15
std	1.000733e+00	1.000733e+00	1.000733e+00	1.000733e+00	1.000733e+00	1.000733e+00	1.000733e+00	1.000733e+00	1.000733e+00
min	-1.221191e+00	-7.022120e-01	-7.417736e-01	-6.393655e-01	-1.005763e+00	-6.988531e-01	-9.988531e-01	-6.129274e-01	-3.483997e-01
25%	-8.664174e-01	-7.022120e-01	-7.417736e-01	-6.393655e-01	-5.556085e-01	-6.988531e-01	-5.903401e-01	-6.129274e-01	-3.483997e-01
50%	-1.568693e-01	-7.022120e-01	-7.417736e-01	-6.393655e-01	-5.556085e-01	-6.988531e-01	-1.818272e-01	-6.129274e-01	-3.483997e-01
75%	5.526787e-01	6.037398e-01	5.976352e-01	4.086824e-01	3.447014e-01	6.743249e-01	6.351988e-01	3.705403e-01	-3.483997e-01
max	1.971775e+00	2.236180e+00	2.271896e+00	2.504778e+00	3.045631e+00	1.772867e+00	2.677764e+00	2.337476e+00	4.849690e+00

train data 와 test data로 구분(7:3 비율)

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=10)
```

```
print('train data 개수: ', X_train.shape)
```

```
print('test data 개수: ', X_test.shape)
```

```
print('\n')
```

```
train data 개수: (478, 9)
test data 개수: (205, 9)
```

```
'''
```

[Step 4] Decision Tree 분류 모형 - sklearn 사용

```
'''
```

```
# sklearn 라이브러리에서 Decision Tree 분류 모형 가져오기
from sklearn import tree
```

```
# 모형 객체 생성 (criterion='entropy' 적용)
tree_model = tree.DecisionTreeClassifier(criterion='entropy', max_depth=5)
```

설명 : 분류정도를 평가하는 기준으로 'entropy' 값을 사용하겠다는 의미
max_depth = 5 는 트리 레벨을 5로 지정해서 가지의 확장을 5단계까지 확장시키겠다는 의미
레벨이 많아질수록 모형 학습에 사용하는 훈련 데이터에 대한 예측은 정확해진다.
그러나 모형이 훈련 데이터에 대해서만 지나치게 최적화 되어 실제 데이터의 예측 능력은 떨어지는 문제가 발생한다. (오버피팅 문제가 발생)
머신러닝 데이터 분석시 적절한 max_depth를 찾는게 데이터 분석가의 역할이다.

면접문제 :

8. 의사결정트리에서 트리의 depth가 깊게 되었을 때의 장단점을 설명해보시오.

답 : max_depth = 5 는 트리 레벨을 5로 지정해서 가지의 확장을 5단계까지 확장시키겠다는 의미
레벨이 많아질수록 모형 학습에 사용하는 훈련 데이터에 대한 예측은 정확해진다.
그러나 모형이 훈련 데이터에 대해서만 지나치게 최적화 되어 실제 데이터의 예측 능력은 떨어지는 문제가 발생한다. (오버피팅 문제가 발생)
머신러닝 데이터 분석시 적절한 max_depth를 찾는게 데이터 분석가의 역할이다.

```
# train data를 가지고 모형 학습
tree_model.fit(X_train, y_train)
```

```
# test data를 가지고 y_hat을 예측 (분류)
y_hat = tree_model.predict(X_test)      # 2: benign(양성), 4: malignant(악성)
```

```
print(y_hat[0:10])
print(y_test.values[0:10])
print('\n')
```

```
[4 4 4 4 4 4 2 2 4 4]
[4 4 4 4 4 4 2 2 4 4]
```

```
# 모형 성능 평가 - Confusion Matrix 계산
from sklearn import metrics
tree_matrix = metrics.confusion_matrix(y_test, y_hat)
print(tree_matrix)
print('\n')
```

```
[[127  4]
 [ 2  72]]
```

```
# 모형 성능 평가 - 평가지표 계산
tree_report = metrics.classification_report(y_test, y_hat)
print(tree_report)
```

	precision	recall	f1-score	support
2	0.98	0.97	0.98	131
4	0.95	0.97	0.96	74
accuracy			0.97	205
macro avg	0.97	0.97	0.97	205
weighted avg	0.97	0.97	0.97	205

✕ 문제230. class 를 value_counts 하여 각각 건수가 어떻게 되는지 확인하시오

```
df['class'].value_counts()
```

```
2    444
4    239
Name: class, dtype: int64
```

2: 양성

4: 악성

✕ 문제231. 위의 의사결정트리 모델을 아산병원에서 사용할 수 있도록 FN을 0으로 만드는 max_depth를 알아내시오

```
'''
```

```
[Step 4] Decision Tree 분류 모형 - sklearn 사용
```

```
'''
```

```
# sklearn 라이브러리에서 Decision Tree 분류 모형 가져오기
from sklearn import tree
```

```
# 모형 객체 생성 (criterion='entropy' 적용)
tree_model = tree.DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

```
# train data를 가지고 모형 학습
tree_model.fit(X_train, y_train)
```

```
# test data를 가지고 y_hat을 예측 (분류)
y_hat = tree_model.predict(X_test)      # 2: benign(양성), 4: malignant(악성)
```

```
print(y_hat[0:10])
print(y_test.values[0:10])
print('\n')
```

```
# 모형 성능 평가 - Confusion Matrix 계산
from sklearn import metrics
tree_matrix = metrics.confusion_matrix(y_test, y_hat)
print(tree_matrix)
print('\n')
```

```
# 모형 성능 평가 - 평가지표 계산
```

```
tree_report = metrics.classification_report(y_test, y_hat)
print(tree_report)
```

```
[[126  5]
 [  0 74]]
```

	precision	recall	f1-score	support
2	1.00	0.96	0.98	131
4	0.94	1.00	0.97	74
accuracy			<u>0.98</u>	205
macro avg	0.97	0.98	0.97	205
weighted avg	0.98	0.98	0.98	205

✕ 문제232. seaborn의 타이타닉 데이터를 이용해서 의사결정트리 모델을 만들고 테스트 데이터의 정확도를 확인하시오

```
import seaborn as sns

tt = sns.load_dataset('titanic')

pd.set_option('display.max_columns', 15)

'''

[Step 2] 데이터 탐색

'''

tt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
#   Column      Non-Null Count  Dtype
---  -
0   survived    891 non-null    int64
1   pclass      891 non-null    int64
2   sex         891 non-null    object
3   age         714 non-null    float64
4   sibsp       891 non-null    int64
5   parch       891 non-null    int64
6   fare        891 non-null    float64
7   embarked    889 non-null    object
8   class       891 non-null    category
9   who         891 non-null    object
10  adult_male  891 non-null    bool
11  deck        203 non-null    category
12  embark_town 889 non-null    object
13  alive       891 non-null    object
14  alone       891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.6+ KB
```

```
tt.describe()
```

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```

rdf = tt.drop(['deck', 'embark_town'], axis=1)
rdf = rdf.dropna(subset=['age'], how='any', axis=0)
ndf = rdf[['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'embarked']]

gender = pd.get_dummies(ndf['sex'])
ndf = pd.concat([ndf, gender], axis=1)

onehot_embarked = pd.get_dummies(ndf['embarked'], prefix='town')
onehot_embarked

ndf = pd.concat([ndf, onehot_embarked], axis=1)
ndf.drop(['sex', 'embarked'], axis=1, inplace=True)

'''

[Step 3] 데이터셋 구분 - 훈련용(train data)/ 검증용(test data)

'''

X = ndf[['pclass', 'age', 'sibsp', 'parch', 'female', 'male', 'town_C', 'town_Q', 'town_S']]
y = ndf['survived']

# 설명 변수 데이터를 정규화
from sklearn import preprocessing
X = preprocessing.StandardScaler().fit(X).transform(X)

pd.DataFrame(X).describe()

# train data 와 test data로 구분(7:3 비율)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=10)

print('train data 개수: ', X_train.shape)
print('test data 개수: ', X_test.shape)
print('\n')

train data 개수: (499, 9)
test data 개수: (215, 9)

'''

[Step 4] Decision Tree 분류 모형 - sklearn 사용

'''

# sklearn 라이브러리에서 Decision Tree 분류 모형 가져오기
from sklearn import tree

```



```
# 모형 객체 생성 (criterion='entropy' 적용)
tree_model = tree.DecisionTreeClassifier(criterion='entropy', max_depth=5)

# train data를 가지고 모형 학습
tree_model.fit(X_train, y_train)

# test data를 가지고 y_hat을 예측 (분류)
y_hat = tree_model.predict(X_test)      # 2: benign(양성), 4: malignant(악성)

print(y_hat[0:10])
print(y_test.values[0:10])
print('\n')

# 모형 성능 평가 - Confusion Matrix 계산
from sklearn import metrics
tree_matrix = metrics.confusion_matrix(y_test, y_hat)
print(tree_matrix)
print('\n')

# 모형 성능 평가 - 평가지표 계산
tree_report = metrics.classification_report(y_test, y_hat)
print(tree_report)
```

```
[[121  4]
 [ 35 55]]
```

	precision	recall	f1-score	support
0	0.78	0.97	0.86	125
1	0.93	0.61	0.74	90
accuracy			0.82	215
macro avg	0.85	0.79	0.80	215
weighted avg	0.84	0.82	0.81	215

age를 최빈값으로 채웠을 경우

```
freq = rdf['age'].value_counts(dropna=True).idxmax()
rdf['age'].fillna(freq, inplace=True)
```

```
[[153  21]
 [ 22  72]]
```

	precision	recall	f1-score	support
0	0.87	0.88	0.88	174
1	0.77	0.77	0.77	94
accuracy			0.84	268
macro avg	0.82	0.82	0.82	268
weighted avg	0.84	0.84	0.84	268

0.8395522388059702

선생님 답

```
from sklearn import metrics
import numpy as np
```

```

# 1단계 csv --> 데이터 프레임으로 변환
import pandas as pd
import seaborn as sns
df = sns.load_dataset('titanic')

# 컬럼이 모두 다 출력될 수 있도록 출력할 열의 개수 한도를 늘리기
pd.set_option('display.max_columns',15)

# 2단계 결측치 확인하고 제거하거나 치환한다.

# 2.1 타이타닉 데이터 프레임의 자료형을 확인한다.

mask4 = (df.age<10) | (df.sex=='female')

df['child_women']=mask4.astype(int)

# 2.2 결측치 (NaN) 을 확인한다.
# 2.3 deck 컬럼과 embark_town 컬럼을 삭제한다.
# 설명 : deck 결측치가 많아서 컬럼을 삭제해야함.
#          embark 와 embark_town 이 같은 데이터여서 embark 컬럼을 삭제해야함

rdf = df.drop(['deck','embark_town'], axis =1)

# 2.4 age(나이) 열에 나이가 없는 모든행을 삭제한다.
# 데이터가 한개라도 없으면 drop 해라 (how = 'any')
# 모든 데이터가 없으면 drop 해라 (how = 'all')
rdf = rdf.dropna( subset=['age'], how='any', axis=0)

# 2.5 embark 열의 NaN 값을 승선도시중 가장 많이 출현한 값으로 치환하기
most_freq = rdf['embarked'].value_counts().idxmax()
rdf['embarked'].fillna(most_freq, inplace = True)

# 3단계 범주형 데이터를 숫자형으로 변환하기
# 3.1 feature selection (분석에 필요한 속성을 선택)
ndf = rdf[['survived','pclass','sex','age','sibsp','parch','embarked','child_women']]

# 선택된 컬럼중 2개(sex, embarked) 가 범주형이다.

#3.2 범주형 데이터를 숫자로 변환하기(원핫 인코딩)

gender = pd.get_dummies(ndf['sex'])
ndf = pd.concat([ndf,gender], axis= 1)

onehot_embarked = pd.get_dummies(ndf['embarked'])

ndf = pd.concat([ndf,onehot_embarked],axis=1)

ndf.drop(['sex','embarked'], axis=1, inplace = True)

# 4단계 정규화
# 4.1 독립변수와 종속변수(라벨) 을 지정한다.
# survived pclass age sibsp parch female male C Q S
# 라벨                데이터

```

```

# 종속변수                독립변수

x = ndf[ ['pclass', 'age' , 'sibsp', 'parch' , 'female', 'male', 'C' , 'Q' , 'S',      'child_women']] ]
y = ndf['survived'] # 종속변수

# 4.2 독립변수들을 정규화 한다.
from sklearn import preprocessing
X = preprocessing.StandardScaler().fit(x).transform(x)

# 5단계 훈련 데이터를 훈련 데이터 / 테스트 데이터로 나눈다

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,

                                                random_state = 33)

# sklearn 라이브러리에서 나이브베이지 분류 모형 가져오기
from sklearn import tree

# 모형 객체 생성 (criterion='entropy' 적용)
tree_model = tree.DecisionTreeClassifier(criterion='entropy', max_depth=5)
tree_model.fit( X_train, y_train )

# 7단계 테스트 데이터로 예측을 한다.
y_hat = tree_model.predict( X_test )

# 8단계 모형의 예측능력을 평가한다.

from sklearn import metrics
randomforest_matrix = metrics.confusion_matrix( y_test, y_hat )
print( randomforest_matrix )

tn, fp, fn, tp = metrics.confusion_matrix( y_test, y_hat ).ravel()

f1_report = metrics.classification_report( y_test, y_hat )
print( f1_report )

#print(np.array([[tp,fp],[fn,tn]]))

from sklearn.metrics import accuracy_score
accuracy = accuracy_score( y_test, y_hat)
print(accuracy)

```

```

[[109 17]
 [ 22 67]]

```

	precision	recall	f1-score	support
0	0.83	0.87	0.85	126
1	0.80	0.75	0.77	89
accuracy			0.82	215
macro avg	0.81	0.81	0.81	215
weighted avg	0.82	0.82	0.82	215

```

0.8186046511627907

```

seaborn의 타이타닉 데이터를 의사결정트리 모델로 생성 할 때 주의사항

1. 결측치가 없어야 한다. 모델 생성할 때 에러가 난다.
2. 명목형 데이터가 없어야 한다. 다 숫자형으로 변환해야한다.

% 테스트 결과 표로 정리 %

머신러닝	정확도
knn	0.8246
의사결정트리	0.81

✂ 문제233. for loop 문을 이용해서 정확도가 높은 max_depth가 무엇인지 알아내시오

```
from sklearn import metrics
import numpy as np

# 1단계 csv ---> 데이터 프레임으로 변환
import pandas as pd
import seaborn as sns
df = sns.load_dataset('titanic')

# 컬럼이 모두다 출력될 수 있도록 출력할 열의 개수 한도를 늘리기
pd.set_option('display.max_columns',15)

# 2단계 결측치 확인하고 제거하거나 치환한다.

# 2.1 타이타닉 데이터 프레임의 자료형을 확인한다.

mask4 = (df.age<10) | (df.sex=='female')

df['child_women']=mask4.astype(int)

# 2.2 결측치 (NaN) 을 확인한다.
# 2.3 deck 컬럼과 embark_town 컬럼을 삭제한다.
# 설명 : deck 결측치가 많아서 컬럼을 삭제해야함.
#          embark 와 embark_town 이 같은 데이터여서 embark 컬럼을 삭제해야함

rdf = df.drop(['deck','embark_town'], axis =1)

# 2.4 age(나이) 열에 나이가 없는 모든행을 최빈값으로 대체한다.
# 데이터가 한개라도 없으면 drop 해라 (how = 'any')
# 모든 데이터가 없으면 drop 해라 (how = 'all')
freq = rdf['age'].value_counts(dropna=True).idxmax()
rdf['age'].fillna(freq, inplace=True)

# 2.5 embark 열의 NaN 값을 승선도시중 가장 많이 출현한 값으로 치환하기
most_freq = rdf['embarked'].value_counts().idxmax()
rdf['embarked'].fillna(most_freq, inplace = True)

# 3단계 범주형 데이터를 숫자형으로 변환하기
# 3.1 feature selection (분석에 필요한 속성을 선택)
ndf = rdf[['survived','pclass','sex','age','sibsp','parch','embarked','child_women']]
```

```
# 선택된 컬럼중 2개 (sex, embarked) 가 범주형이다.
```

```
#3.2 범주형 데이터를 숫자로 변환하기 (원핫 인코딩)
```

```
gender = pd.get_dummies(ndf['sex'])
ndf = pd.concat([ndf,gender], axis= 1)

onehot_embarked = pd.get_dummies(ndf['embarked'])

ndf = pd.concat([ndf,onehot_embarked],axis=1)

ndf.drop(['sex','embarked'], axis=1, inplace = True)
```

```
# 4단계 정규화
```

```
# 4.1 독립변수와 종속변수 (라벨) 을 지정한다.
```

```
# survived pclass age sibsp parch female male C Q S
```

```
# 라벨 데이터
```

```
# 종속변수 독립변수
```

```
x = ndf[ ['pclass', 'age' , 'sibsp', 'parch' , 'female' , 'male', 'C' , 'Q' , 'S', 'child_women' ] ]
y = ndf['survived'] # 종속변수
```

```
# 4.2 독립변수들을 정규화 한다.
```

```
from sklearn import preprocessing
X = preprocessing.StandardScaler().fit(x).transform(x)
```

```
for i in range(1,50) :
    print('random_state:', i)
    # 5단계 훈련 데이터를 훈련 데이터 / 테스트 데이터로 나눈다
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,
                                                random_state = i)
```

```
# sklearn 라이브러리에서 나이브베이지 분류 모형 가져오기
```

```
from sklearn import tree
```

```
for k in range(1,50) :
    print('max_depth: ', k)
    # 모형 객체 생성 (criterion='entropy' 적용)
    tree_model = tree.DecisionTreeClassifier(criterion='entropy', max_depth=k)
    tree_model.fit( X_train, y_train )
```

```
# 7단계 테스트 데이터로 예측을 한다.
y_hat = tree_model.predict( X_test )
```

```
# 8단계 모형의 예측능력을 평가한다.
```

```
from sklearn import metrics
randomforest_matrix = metrics.confusion_matrix( y_test, y_hat )
print( randomforest_matrix )
```

```
tn, fp, fn, tp = metrics.confusion_matrix( y_test, y_hat ).ravel()
```

```
f1_report = metrics.classification_report( y_test, y_hat )
# print( f1_report )
```

```
#print(np.array([[tp,fp],[fn,tn]]))
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score( y_test, y_hat)
print(accuracy, '\n')
```

```

if accuracy > max :
    max = accuracy
    print('random_state: ', i, 'max_depth: ', k, 'accuracy: ', accuracy)
# print (accuracy, '\n')

```

```

random_state: 1 max_depth: 1 accuracy: 0.75
random_state: 1 max_depth: 2 accuracy: 0.753731343283582
random_state: 1 max_depth: 3 accuracy: 0.7723880597014925
random_state: 1 max_depth: 4 accuracy: 0.7835820895522388
random_state: 2 max_depth: 3 accuracy: 0.8022388059701493
random_state: 3 max_depth: 3 accuracy: 0.8097014925373134
random_state: 4 max_depth: 1 accuracy: 0.8208955223880597
random_state: 4 max_depth: 2 accuracy: 0.8395522388059702
random_state: 5 max_depth: 4 accuracy: 0.8470149253731343
random_state: 6 max_depth: 3 accuracy: 0.8507462686567164
random_state: 33 max_depth: 4 accuracy: 0.8582089552238806
random_state: 33 max_depth: 5 accuracy: 0.8694029850746269

```

점심시간 문제

✕ 문제 234. 카페에 올린 for loop문 스크립트를 이용해서 정확도가 가장 좋은 random_state와 max_depth를 알아내시오

```

from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn import metrics
from sklearn.metrics import accuracy_score

b=[]
c=[]

for i in range(1, 50):
    X_train,X_test,y_train,y_test = train_test_split(x,y,test_size = 0.3, random_state = i)

    # sklearn 라이브러리에서 나이브베이지 분류 모형 가져오기
    # 모형 객체 생성 (criterion='entropy' 적용)

    for k in range(1,50):
        b.append((i,k))
        tree_model = tree.DecisionTreeClassifier(criterion='entropy', max_depth=k)
        tree_model.fit( X_train, y_train )

        # 7단계 테스트 데이터로 예측을 한다.
        y_hat = tree_model.predict( X_test )

        # 8단계 모형의 예측능력을 평가한다.
        randomforest_matrix = metrics.confusion_matrix( y_test, y_hat )
        accuracy = accuracy_score( y_test, y_hat)
        c.append(accuracy)

# print(np.max(c))

idx = c.index(np.max(c))
print('random_state: ', b[idx][0], 'max_depth: ', b[idx][1], 'accuracy:', c[idx])

```

```

random_state: 33      max_depth: 5      accuracy: 0.8694029850746269

```


