Ch2 앙상블 기법에 대해 알아보기

대표적인 알고리즘 RandomForest

학습 내용

- 앙상블이 무엇인지 알아본다.
- 랜덤 포레스트 알고리즘을 이용해 본다.
- 기타 모델로 다시 만들어 여러 모델을 비교해 본다.

01 앙상블(ensemble)란 무엇일까?

- **여러 머신러닝 모델을 연결**하여 더 강력한 모델을 만드는 기법이다.

02 랜덤 포레스트(RandomForest)는 무엇인가?

- 원리
  - 01 트리를 많이 만든다.
  - 02 각각의 모델이 예측한다.
  - 03 예측한 값들의 평균값을 구한다. 이를 최종 예측값으로 이용

03 실습

In [1]:

```
import seaborn as sns
```

In [2]:

```
tips = sns.load_dataset("tips")
tips.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype   
---  ------      --------------  -----   
 0   total_bill  244 non-null    float64 
 1   tip         244 non-null    float64 
 2   sex         244 non-null    category
 3   smoker      244 non-null    category
 4   day         244 non-null    category
 5   time        244 non-null    category
 6   size        244 non-null    int64   
dtypes: category(4), float64(2), int64(1)
memory usage: 7.4 KB
```

In [3]:

```
tips.head()
```

Out[3]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

- 데이터 셋 내용
  - total_bill : 총 지불 비용
  - tip : 팁
  - sex : 성별
  - smoker : 담배를 피는 안피는지
  - day : 이용한 요일
  - time : 점심인지 저녁인지
  - size : 식당 이용 인원

04 머신러닝 과제

In [4]:

```
tips.shape
```

Out[4]:

```
(244, 7)
```

- 조건 1. 우리에게는 지금까지 이용한 고객의 220개의 데이터가 있다.
- 조건 2. 이후에 몇명이 이용할지 모른다.
- 조건 3. 우리는 size을 예측하는 머신러닝 시스템을 만들어, 이를 토대로 앞으로의 고객 서비스에 반영해보자.

주어진 데이터를 토대로 이용 고객을 예측해 보자.

우선 데이터 만들어보기

In [5]:

```
tips_have = tips.iloc[ 0:220, :]  # 현재 가진 고객 데이터
tips_new  = tips.iloc [220: , :]  # 미래의 고객 데이터

tips_new.drop(["size"], axis=1, inplace=True)

tips_have.shape, tips_new.shape
```

```
C:₩Users₩totofriend₩AppData₩Local₩Temp₩ipykernel_2576₩841681958.py:4: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  tips_new.drop(["size"], axis=1, inplace=True)
```

Out[5]:

```
((220, 7), (24, 6))
```

In [6]:

```
tips_have.columns, tips_new.columns
```

Out[6]:

```
(Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'], dtype='objec
t'),
 Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time'], dtype='object'))
```

05 머신러닝 과제 수행

In [7]:

```
tips_have.head()
```

Out[7]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

- 머신러닝은 숫자 데이터를 좋아하고 이해할 수 있다.
  - 그러면 total_bill, size의 컬럼(변수)를 사용해서 'tip'을 예측하는 과제를 수행한다.

In [8]:

```
sel = ['total_bill', 'tip']
```

- 01 머신러닝에서 모델 만들고 예측해보기

머신러닝은 다음과 같은 과정을 거친다.

- 모델 만들고
- 선택된 모델을 준비된 데이터(입력, 출력)로 학습을 시키고
- 마지막으로 학습된 모델로 새로운 데이터를 예측을 수행한다.

우리의 과제

- 모델에 사용할 데이터를 준비한다.
  - 학습-입력(X_train), 학습-출력(y_train)
  - 예측에 사용할 새로운 데이터(X_test), y_test(는 예측되므로 없음)

In [9]:

```
# sel = ['total_bill', 'tip']

X = tips_have[sel]
y = tips_have['size']  # 우리가 예측할 컬럼(변수)

test_X = tips_new[sel]       # 예측할 친구는 다른 데이터 셋
```

랜덤 포레스트 이용

- 예측하려는 타깃(레이블)이 수치형일때는 RandomForestRegressor를 활용
- 예측하려는 타깃(레이블)이 범주형일때는 RandomForestClassifier를 활용

In [10]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [11]:

```
model = RandomForestClassifier() # 모델 만들기
model.fit(X, y)     # 모델 훈련시키기  model.fit(입력, 출력)
pred = model.predict(test_X)    # 학습된 모델로 예측하기
pred
```

Out[11]:

```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 5, 4, 4, 4, 4,
       2, 4], dtype=int64)
```

06 우리가 만든 모델이 좋은지 아닌지 어떻게 평가할 수 없을까?

- 내가 만든 모델이 어느정도 좋은 성능을 가지는지 현재로서는 알기가 어렵다.
  - 해결 방안 1. tips_have에는 출력 size가 있다. tips_new는 없다. 그러면 우선 tips_have을 잘 데이터로 나누어 학습과 예측을 하여, 가진 답으로 맞추어보고 검증을 해보자.

- train_test_split 함수를 이용하여 학습용, 테스트용으로 나눌 수 있다.

In [12]:

```
from sklearn.model_selection import train_test_split
```

In [13]:

```
# random_state는 난수 발생기의 패턴을 고정시키기 위해 사용한다.
# 이를 통해 우리는 X(입력), y(출력)이 각각 학습용, 테스트용으로 나누어진다.
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

07 다시 모델을 만들고, 이제는 평가가 가능하다. 살펴보자.

In [14]:

```
model = RandomForestClassifier() # 모델 만들기
model.fit(X_train, y_train)     # 모델 훈련시키기  model.fit(입력, 출력)
pred = model.predict(X_test)    # 학습된 모델로 예측하기
pred
```

Out[14]:

```
array([2, 2, 2, 2, 3, 3, 2, 2, 2, 2, 2, 2, 5, 2, 2, 2, 2, 4, 2, 2, 3, 2,
       4, 2, 2, 2, 2, 2, 2, 2, 3, 4, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2,
       2, 2, 2, 4, 3, 2, 2, 2, 2, 2, 2], dtype=int64)
```

여기서 예측한 pred와 y_test는 비교하여 얼마나 오차가 있는지 확인 가능하다.

In [15]:

```
pred == y_test
```

Out[15]:

```
152    False
74      True
71     False
161     True
162     True
143    False
63     False
153    False
219    False
135     True
149     True
5      False
90     False
168     True
202     True
191     True
201     True
96     False
106     True
75      True
55     False
12      True
157     True
64     False
37     False
130     True
101     True
61      True
8       True
18     False
179    False
15     False
139     True
7      False
124     True
159    False
136     True
144     True
199     True
155    False
66      True
33     False
89      True
158     True
196     True
173     True
185    False
207     True
16      True
145     True
200    False
146    False
22      True
183    False
```

```
45      True
Name: size, dtype: bool
```

In [16]:

```
# 오차를 계산해 보자.
(pred == y_test).sum()
```

Out[16]:

```
32
```

In [17]:

```
### model.score()를 이용해서 구하기
print( model.score(X_train, y_train) )
print( model.score(X_test, y_test) )
```

```
1.0
0.5818181818181818
```

정확도가 58.2%이다.

08 다른 모델의 정확도는 어떨까? 확인해 보자.

In [18]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
```

In [19]:

```
model = KNeighborsClassifier() # 모델 만들기
model.fit(X_train, y_train)    # 모델 훈련시키기  model.fit(입력, 출력)
pred = model.predict(X_test)   # 학습된 모델로 예측하기
pred
```

Out[19]:

```
array([2, 2, 2, 2, 2, 4, 2, 2, 3, 2, 2, 2, 3, 2, 2, 2, 2, 3, 3, 2, 2, 2,
       3, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 3, 2, 2, 2, 2, 2, 4, 2, 2, 3, 2,
       2, 2, 2, 4, 2, 2, 2, 4, 2, 2, 2], dtype=int64)
```

In [21]:

```
# 정확도
(pred == y_test).sum() / len(pred)  * 100
```

Out[21]:

```
52.72727272727272
```

In [22]:

```
model = DecisionTreeClassifier() # 모델 만들기
model.fit(X_train, y_train)     # 모델 훈련시키기  model.fit(입력, 출력)
pred = model.predict(X_test)    # 학습된 모델로 예측하기
pred
```

Out[22]:

```
array([2, 2, 2, 2, 3, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 3, 2,
       2, 2, 2, 4, 2, 2, 2, 2, 3, 4, 2, 2, 2, 3, 3, 2, 2, 4, 3, 2, 2, 2,
       3, 2, 2, 4, 3, 2, 2, 4, 2, 2, 2], dtype=int64)
```

In [23]:

```
import numpy as np
```

In [24]:

```
# 정확도
# (pred == y_test).sum() / len(pred)  * 100
np.mean(pred == y_test)
```

Out[24]:

```
0.5090909090909091
```

결과 확인

- 일반적인 모델 사용 결과 knn보다 의사결정트리가 좋고,
- 의사결정트리보다 랜덤포레스트 모델이 좋다.
- 랜덤포레스트는 많은 개수의 트리를 사용해서, 많은 트리를 사용하는 것이 좋은 것으로 보여진다.

- 과제
  - 기타 다른 변수들을 머신러닝 모델에 맞게 변경해서 모델에 적용시켜보기