Ch2 앙상블 기법에 대해 알아보기

대표적인 알고리즘 RandomForest

학습 내용

- 앙상블이 무엇인지 알아본다.
- 랜덤 포레스트 알고리즘을 이용해 본다.
- Tips 데이터 셋의 라벨 인코딩을 수행해 본다.
- Tip을 예측하는 모델 만들어보기(회귀)

목차

01 tip 예측 모델 만들기

목차로 이동하기

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

02 데이터 전처리

목차로 이동하기

In [4]:

```
sex_dict = { 'Female':0, "Male":1 }
tips['sex'] = tips['sex'].map(sex_dict).astype('int')

# day 라벨 인코딩
print( tips['day'].unique() )
day_dict = {"Thur":0, "Fri":1, "Sat":2, "Sun":3 }
tips['day'] = tips['day'].map(day_dict).astype('int')

# time 라벨 인코딩
print( tips['time'].unique() )
time_dict = {"Lunch":0, "Dinner":1 }
tips['time'] = tips['time'].map(time_dict).astype('int')

# smoker 라벨 인코딩
print( tips['smoker'].unique() )
smoker_dict = {"No":0, "Yes":1 }
tips['smoker'] = tips['smoker'].map(smoker_dict).astype('int')
```

```
['Sun', 'Sat', 'Thur', 'Fri']
Categories (4, object): ['Thur', 'Fri', 'Sat', 'Sun']
['Dinner', 'Lunch']
Categories (2, object): ['Lunch', 'Dinner']
['No', 'Yes']
Categories (2, object): ['Yes', 'No']
```

03 머신러닝 과제

목차로 이동하기

In [5]:

```
tips.shape
```

Out[5]:

```
(244, 7)
```

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

주어진 데이터를 토대로 이용 고객을 예측해 보자.

우선 데이터 만들어보기

In [6]:

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

tips_new = tips_new.drop(["tip"], axis=1)

tips_have.shape, tips_new.shape
```

Out[6]:

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

In [7]:

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

Out[7]:

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

03 머신러닝 과제 수행

목차로 이동하기

In [8]:

```
tips_have.head()
```

Out[8]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	0	0	3	1	2
1	10.34	1.66	1	0	3	1	3
2	21.01	3.50	1	0	3	1	3
3	23.68	3.31	1	0	3	1	2
4	24.59	3.61	0	0	3	1	4

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

In [9]:

```
sel = ['total_bill', 'sex', 'smoker', 'day', 'time', 'size']
```

- 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 [11]:

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

X = tips_have[sel]
y = tips_have['tip']  # 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 [12]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [13]:

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

Out[13]:

```
array([1.7753, 2.104 , 1.5641, 2.2022, 2.0179, 3.439 , 1.8931, 3.2047,
       2.1622, 3.7503, 4.1024, 2.3066, 1.8401, 1.6089, 2.0958, 1.6091,
       2.0438, 3.2046, 4.661 , 3.2753, 3.7255, 3.506 , 3.1525, 2.433 ])
```

04 우리가 만든 모델 평가

목차로 이동하기

- 내가 만든 모델이 어느정도 좋은 성능을 가지는지 현재로서는 알기가 어렵다.
  - 해결 방안 1. tips_have에는 출력 tip이 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 [14]:

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

In [15]:

```
# 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)
```

In [16]:

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

Out[16]:

```
array([2.803 , 2.8472, 2.1464, 2.2139, 3.1024, 3.8019, 2.9998, 3.5765,
       3.5871, 1.5325, 1.4874, 4.3599, 3.0382, 1.9417, 1.9197, 3.0517,
       1.9484, 3.4526, 3.2648, 1.7675, 2.693 , 2.3385, 4.1946, 2.9458,
       2.1834, 3.0368, 2.4177, 2.4179, 2.7284, 2.0693, 3.2563, 3.9861,
       1.9542, 3.901 , 1.7927, 3.1169, 1.6914, 2.804 , 1.9663, 4.2911,
       3.0235, 3.2696, 3.3403, 2.2963, 1.8286, 3.2605, 2.9282, 3.7451,
       2.2386, 1.5275, 2.9601, 2.9664, 2.7267, 4.0704, 3.1787])
```

여기서 예측한 pred와 y_test는 비교하여 얼마나 오차가 있는지 확인 가능하다.

In [17]:

```
pred - y_test
```

Out[17]:

```
152    0.0630
74     0.6472
71    -0.8536
161   -0.2861
162    1.1024
143   -1.1981
63    -0.7602
153    1.5765
219    0.4971
135    0.2825
149   -0.5126
5     -0.3501
90     0.0382
168    0.3317
202   -0.0803
191   -1.1383
201   -0.0616
96    -0.5474
106   -0.7952
75     0.5175
55    -0.8170
12     0.7685
157    0.4446
64     0.3058
37    -0.8866
130    1.5368
101   -0.5823
61     0.4179
8      0.7684
18    -1.4307
179   -0.2937
15     0.0661
139   -0.7958
7      0.7810
124   -0.7273
159    1.1169
136   -0.3086
144    0.5040
199   -0.0337
155   -0.8489
66     0.5535
33     0.8196
89     0.3403
158   -0.3137
196   -0.1714
173    0.0805
185   -2.0718
207    0.7451
16     0.5686
145    0.0275
200   -1.0399
146    1.6064
22     0.4967
183   -2.4296
```

45 0.1787

In [18]:

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, pred)
```

Out[18]:

0.6639545454545457

In [19]:

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

0.9205093342341426
0.42502604302344027

05 다른 모델의 정확도는 어떨까? 확인해 보자.

목차로 이동하기

In [21]:

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
```

In [22]:

```
model = KNeighborsRegressor() # 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([2.95 , 3.25 , 2.404, 2.094, 2.492, 3.856, 2.73 , 3.806, 3.934,
       1.596, 1.418, 4.118, 3.25 , 2.448, 1.896, 3.216, 1.822, 3.422,
       2.95 , 1.692, 2.686, 2.792, 4.118, 3.152, 2.492, 3.158, 3.006,
       2.718, 2.762, 2.752, 4.598, 3.824, 1.896, 3.792, 1.822, 2.658,
       1.712, 2.686, 2.036, 4.034, 2.762, 3.1  , 2.902, 2.412, 1.774,
       4.122, 3.1  , 4.462, 1.77 , 1.596, 3.392, 3.316, 2.612, 3.638,
       3.076])
```

In [24]:

```
print("mse 값 : ", mean_absolute_error(y_test, pred) )
print("학습 결정계수 : ", model.score(X_train, y_train) )
print("테스트 결정계수 : ", model.score(X_test, y_test) )
```

mse 값 : 0.7037818181818182
학습 결정계수 : 0.5826625545543387
테스트 결정계수 : 0.33748841292974685

In [25]:

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

Out[25]:

```
array([3.5 , 3.  , 2.  , 2.5 , 3.71, 4.2 , 2.5 , 3.61, 5.16, 1.48, 1.44,
       4.34, 2.56, 2.  , 2.  , 3.  , 2.23, 3.14, 3.21, 2.  , 3.  , 1.64,
       3.61, 3.  , 2.  , 2.83, 2.02, 1.68, 3.23, 1.01, 3.11, 3.5 , 2.  ,
       5.07, 1.5 , 3.  , 1.83, 2.5 , 1.68, 2.  , 2.  , 2.  , 3.  , 2.  ,
       1.73, 5.  , 2.  , 3.  , 1.66, 1.48, 4.  , 4.  , 3.16, 3.61, 3.  ])
```

In [26]:

```
print("mse 값 : ", mean_absolute_error(y_test, pred) )
print("학습 결정계수 : ", model.score(X_train, y_train) )
print("테스트 결정계수 : ", model.score(X_test, y_test) )
```

```
mse 값 :  0.8532727272727273
학습 결정계수 :  1.0
테스트 결정계수 :  -0.14791046025190835
```

결과 확인

- 일반적인 모델 사용 결과 knn이 의사결정트리보다 오차가 좋고
- 랜덤 포레스트 모델이 가장 오차가 적다.