제2유형_연습하기_팁 예측하기(회귀)

✅ 데이터 분석 순서

1. 라이브러리 및 데이터 확인

2. 데이터 탐색(EDA)

3. 데이터 전처리 및 분리

4. 모델링 및 성능평가

5. 예측값 제출

✅ 1. 라이브러리 및 데이터 확인

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
###############  복사 영역  ###############
# 실기 시험 데이터셋으로 셋팅하기 (수정금지)

import seaborn as sns
# tips 데이터셋 로드
df = sns.load_dataset('tips')

x = df.drop(['tip'], axis=1)
y = df['tip']


# 실기 시험 데이터셋으로 셋팅하기 (수정금지)
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=2023)

x_test = pd.DataFrame(x_test.reset_index())
x_train = pd.DataFrame(x_train.reset_index())
y_train = pd.DataFrame(y_train.reset_index())

x_test.rename(columns={'index':'cust_id'}, inplace=True)
x_train.rename(columns={'index':'cust_id'}, inplace=True)
y_train.columns = ['cust_id', 'target']


### 참고사항 ###
# y_test 는 실기 문제상에 주어지지 않음

# ★Tip : X를 대문자로 쓰지말고 소문자 x로 쓰세요. 시험에서 실수하기 쉽습니다.(문제풀기 전에 소문자로 변경!)
# (참고 : 보통 X는 2차원 배열(행렬)이기 때문에 대문자로 쓰고, y는 1차원 배열(벡터)이기 때문에 소문자로 씀)

# (~23년 10월말) 실기시험 데이터 형식 (실제 시험장에서는 다를 수 있으니 반드시 체크)
# X_test = pd.read_csv("data/X_test.csv")
# X_train = pd.read_csv("data/X_train.csv")
# y_train = pd.read_csv("data/y_train.csv")

# ★(23년 10월말~) 기준으로 체험환경에서 제공되는 데이터셋이 조금 변경되었습니다.
# train = pd.read_csv("data/customer_train.csv")
# test = pd.read_csv("data/customer_test.csv")
# x_train과 y_train, x_test를 별도로 할당해주셔야 합니다.
```


레스토랑의 tip 예측 문제

- 데이터의 결측치, 이상치, 변수에 대해 처리하고

- 회귀모델을 사용하여 Rsq, MSE 값을 산출하시오.

데이터셋 설명

> - total_bill(총 청구액): 손님의 식사 총 청구액
> - tip(팁): 팁의 양
> - sex(성별): 손님의 성별
> - smoker(흡연자): 손님의 흡연 여부("Yes" 또는 "No")

- day(요일): 식사가 이루어진 요일
- time(시간): 점심 또는 저녁 중 언제 식사가 이루어졌는지
- size(인원 수): 식사에 참석한 인원 수

✅ 2. 데이터 탐색(EDA)

In [3]:
```
# 데이터의 행/열 확인

print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
```

```
(195, 7)
(49, 7)
(195, 2)
```

In [4]:
```
# 초기 데이터 확인

print(x_train.head(3))
print(x_test.head(3))
print(y_train.head(3))
```

```
   cust_id  total_bill     sex smoker  day    time  size
0      158       13.39  Female     No  Sun  Dinner     2
1      186       20.90  Female    Yes  Sun  Dinner     3
2       21       20.29  Female     No  Sat  Dinner     2
   cust_id  total_bill     sex smoker  day    time  size
0      154       19.77    Male     No  Sun  Dinner     4
1        4       24.59  Female     No  Sun  Dinner     4
2       30        9.55    Male     No  Sat  Dinner     2
   cust_id  target
0      158    2.61
1      186    3.50
2       21    2.75
```

In [5]:
```
# 변수명과 데이터 타입이 매칭이 되는지, 결측치가 있는지 확인해보세요

print(x_train.info())
print(x_test.info())
print(y_train.info())
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   cust_id     195 non-null    int64
 1   total_bill  195 non-null    float64
 2   sex         195 non-null    category
 3   smoker      195 non-null    category
 4   day         195 non-null    category
 5   time        195 non-null    category
 6   size        195 non-null    int64
dtypes: category(4), float64(1), int64(2)
memory usage: 6.0 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49 entries, 0 to 48
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   cust_id     49 non-null     int64
 1   total_bill  49 non-null     float64
 2   sex         49 non-null     category
 3   smoker      49 non-null     category
 4   day         49 non-null     category
 5   time        49 non-null     category
 6   size        49 non-null     int64
dtypes: category(4), float64(1), int64(2)
memory usage: 2.0 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   cust_id  195 non-null    int64
 1   target   195 non-null    float64
dtypes: float64(1), int64(1)
memory usage: 3.2 KB
None
```


In [6]:
```
# x_train 과 x_test 데이터의 기초통계량을 잘 비교해보세요.

print(x_train.describe())  # x_train.describe().T 들중에 편한걸 사용하세요
print(x_test.describe())
```

```
print(y_train.describe())
```

```
          cust_id  total_bill        size
count  195.000000  195.000000  195.000000
mean   122.056410   20.054667    2.543590
std     70.668034    8.961645    0.942631
min      0.000000    3.070000    1.000000
25%     59.500000   13.420000    2.000000
50%    121.000000   17.920000    2.000000
75%    182.500000   24.395000    3.000000
max    243.000000   50.810000    6.000000
          cust_id  total_bill       size
count   49.000000   49.000000  49.000000
mean   119.285714   18.716531   2.673469
std     70.918674    8.669864   0.987162
min      2.000000    5.750000   2.000000
25%     62.000000   12.740000   2.000000
50%    123.000000   16.660000   2.000000
75%    180.000000   21.010000   3.000000
max    239.000000   44.300000   6.000000
          cust_id      target
count  195.000000  195.000000
mean   122.056410    3.021692
std     70.668034    1.402690
min      0.000000    1.000000
25%     59.500000    2.000000
50%    121.000000    2.920000
75%    182.500000    3.530000
max    243.000000   10.000000
```

In [7]:
```
# object, category 데이터도 추가 확인
# print(x_train.describe(include='object'))
# print(x_test.describe(include='object'))

print(x_train.describe(include='category'))
print(x_test.describe(include='category'))
```

```
         sex smoker  day    time
count    195    195  195     195
unique     2      2    4       2
top     Male     No  Sat  Dinner
freq     125    120   71     142
         sex smoker  day    time
count     49     49   49      49
unique     2      2    4       2
top     Male     No  Sat  Dinner
freq      32     31   16      34
```

In [8]:
```
# y데이터도 구체적으로 살펴보세요.
print(y_train.head())
```

```
   cust_id  target
0      158    2.61
1      186    3.50
2       21    2.75
3       74    2.20
4       43    1.32
```

In [9]:
```
# y데이터도 구체적으로 살펴보세요.
print(y_train.describe().T)
```

```
         count        mean        std  min   25%     50%     75%    max
cust_id  195.0  122.056410  70.668034  0.0  59.5  121.00  182.50  243.0
target   195.0    3.021692   1.402690  1.0   2.0    2.92    3.53   10.0
```

✓ 3. 데이터 전처리 및 분리

1) 결측치, 2) 이상치, 3) 변수 처리하기

In [10]:
```
# 결측치 확인
print(x_train.isnull().sum())
print(x_test.isnull().sum())
print(y_train.isnull().sum())
```

```
cust_id       0
total_bill    0
sex           0
smoker        0
day           0
time          0
size          0
dtype: int64
cust_id       0
total_bill    0
sex           0
smoker        0
day           0
time          0
size          0
dtype: int64
cust_id    0
target     0
dtype: int64
```


In [11]:
```
# 결측치 제거
# df = df.dropna()
# print(df)

# 참고사항
# print(df.dropna().shape)  # 행 기준으로 삭제

# ★주의사항
# x_train의 행을 제거해야 하는 경우, 그에 해당하는 y_train 행도 제거해야 합니다.
# 해결방법 : train = pd.concat([x_train, y_train], axis=1)
# 위와 같이 데이터를 결합한 후에 행을 제거하고 다시 데이터 분리를 수행하면 됩니다.
# (만약 원데이터가 x_train/y_train이 결합된 형태로 주어진다면 전처리를 모두 수행한 후에 분리하셔도 됩니다)
```

In [12]:
```
# 결측치 대체(평균값, 중앙값, 최빈값)
# ** 주의사항 : train 데이터의 중앙값/평균값/최빈값 등으로 test 데이터의 결측치도 변경해줘야 함 **

# 연속형 변수 : 중앙값, 평균값
#  - df['변수명'].median()
#  - df['변수명'].mean()
# 범주형 변수 : 최빈값

# df['변수명'] = df['변수명'].fillna(대체할 값)
```

In [13]:
```
# 이상치 대체
# (참고) df['변수명'] = np.where( df['변수명'] >= 5, 대체할 값, df['변수명'] )
```


In [14]:
```
# 변수처리

# 불필요한 변수 제거
# df = df.drop(columns = ['변수1','변수2'])
# df = df.drop(['변수1','변수2'], axis=1)

# 필요시 변수 추가(파생변수 생성)
# df['파생변수명'] = df['A'] * df['B'] (파생변수 생성 수식)

# 원핫인코딩(가변수 처리)
# x_train = pd.get_dummies(x_train)
# x_test = pd.get_dummies(x_test)
# print(x_train.info())
# print(x_test.info())
```

In [15]:
```
# 변수처리

# 불필요한 변수(columns) 제거
# cust_id 는 불필요한 변수이므로 제거합니다.
# 단, test 셋의 cust_id가 나중에 제출이 필요하다면 별도로 저장해둠

cust_id = x_test['cust_id'].copy()

# 각 데이터에서 cust_id 변수 제거
x_train = x_train.drop(columns = ['cust_id'])   # drop(columns = ['변수1','변수2']) 변수 추가해서 여러개 삭제 가능
x_test = x_test.drop(columns = ['cust_id'])
```

In [16]:
```
# 변수처리(원핫인코딩)
print(x_train.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype   
---  ------      --------------  -----   
 0   total_bill  195 non-null    float64 
 1   sex         195 non-null    category
 2   smoker      195 non-null    category
 3   day         195 non-null    category
 4   time        195 non-null    category
 5   size        195 non-null    int64   
dtypes: category(4), float64(1), int64(1)
memory usage: 4.5 KB
None
```


In [17]:
```python
# 변수처리(원핫인코딩)
x_train = pd.get_dummies(x_train)
x_test = pd.get_dummies(x_test)
print(x_train.info())
print(x_test.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   total_bill   195 non-null    float64
 1   size         195 non-null    int64  
 2   sex_Male     195 non-null    uint8  
 3   sex_Female   195 non-null    uint8  
 4   smoker_Yes   195 non-null    uint8  
 5   smoker_No    195 non-null    uint8  
 6   day_Thur     195 non-null    uint8  
 7   day_Fri      195 non-null    uint8  
 8   day_Sat      195 non-null    uint8  
 9   day_Sun      195 non-null    uint8  
 10  time_Lunch   195 non-null    uint8  
 11  time_Dinner  195 non-null    uint8  
dtypes: float64(1), int64(1), uint8(10)
memory usage: 5.1 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49 entries, 0 to 48
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   total_bill   49 non-null     float64
 1   size         49 non-null     int64  
 2   sex_Male     49 non-null     uint8  
 3   sex_Female   49 non-null     uint8  
 4   smoker_Yes   49 non-null     uint8  
 5   smoker_No    49 non-null     uint8  
 6   day_Thur     49 non-null     uint8  
 7   day_Fri      49 non-null     uint8  
 8   day_Sat      49 non-null     uint8  
 9   day_Sun      49 non-null     uint8  
 10  time_Lunch   49 non-null     uint8  
 11  time_Dinner  49 non-null     uint8  
dtypes: float64(1), int64(1), uint8(10)
memory usage: 1.4 KB
None
```


데이터 분리

In [18]:
```python
# 데이터를 훈련 세트와 검증용 세트로 분할 (80% 훈련, 20% 검증용)
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(x_train,
                                                  y_train['target'],
                                                  test_size=0.2,
                                                  random_state=23)


print(x_train.shape)
print(x_val.shape)
print(y_train.shape)
print(y_val.shape)
```

```
(156, 12)
(39, 12)
(156,)
(39,)
```


✅ 4. 모델링 및 성능평가

In [19]:
```python
# 랜덤포레스트 모델 사용 (참고 : 분류모델은 RandomForestClassifier)
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(random_state=2023)
```

```
model.fit(x_train, y_train)
```

Out[19]:
```
RandomForestRegressor(random_state=2023)
```

In [20]:
```
# 모델을 사용하여 테스트 데이터 예측
y_pred = model.predict(x_val)
```

In [21]:
```
# 모델 성능 평가 (R-squared, MSE 등)
from sklearn.metrics import r2_score, mean_squared_error
r2 = r2_score(y_val, y_pred)             # (실제값, 예측값)
mse = mean_squared_error(y_val, y_pred)  # (실제값, 예측값)
```

In [22]:
```
# MSE
print(mse)
```

```
0.9812277338461534
```

In [23]:
```
# RMSE
rmse = mse**0.5
print(rmse)
```

```
0.990569398803614
```

In [24]:
```
# Rsq
print(r2)
```

```
0.4286497615634072
```

✓ 5. 예측값 제출

(주의) x_test 를 모델에 넣어 나온 예측값을 제출해야함

In [25]:
```
# (실기시험 안내사항)
# 아래 코드 예측변수와 수험번호를 개인별로 변경하여 활용
# pd.DataFrame({'cust_id': cust_id, 'target': y_result}).to_csv('003000000.csv', index=False)


# 모델을 사용하여 테스트 데이터 예측
y_result = model.predict(x_test)
result = pd.DataFrame({'cust_id': cust_id, 'target': y_result})
print(result[:5])
```

```
   cust_id  target
0      154  3.2266
1        4  4.1160
2       30  1.8966
3       75  1.8735
4       33  3.0267
```

In [26]:
```
# ★tip : 데이터를 저장한다음 불러와서 제대로 제출했는지 확인해보자
# pd.DataFrame({'result': y_result}).to_csv('수험번호.csv', index=False)
# df2 = pd.read_csv("수험번호.csv")
# print(df2.head())
```