제2유형_연습하기_당뇨진척정도(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]:
```
###############  실기환경 복사 영역  ###############
import pandas as pd
import numpy as np
# 실기 시험 데이터셋으로 셋팅하기 (수정금지)
from sklearn.datasets import load_diabetes
# diabetes 데이터셋 로드
diabetes = load_diabetes()
x = pd.DataFrame(diabetes.data, columns=diabetes.feature_names)
y = pd.DataFrame(diabetes.target)

# 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']
###############  실기환경 복사 영역  ###############


### 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.
```

당뇨병 환자의 질병 진행정도를 예측해보자

- 데이터의 결측치, 이상치, 변수들에 대해 전처리하고

- 회귀모델을 사용하여 Rsq, MSE 값을 산출하시오.

- 제출은 cust_id, target 변수를 가진 dataframe 형태로 제출하시오.

In [3]:
```
# 데이터 설명
print(diabetes.DESCR)
```

```
.. _diabetes_dataset:

Diabetes dataset
----------------

Ten baseline variables, age, sex, body mass index, average blood
pressure, and six blood serum measurements were obtained for each of n =
442 diabetes patients, as well as the response of interest, a
quantitative measure of disease progression one year after baseline.

**Data Set Characteristics:**

  :Number of Instances: 442

  :Number of Attributes: First 10 columns are numeric predictive values

  :Target: Column 11 is a quantitative measure of disease progression one year after baseline

  :Attribute Information:
      - age     age in years
      - sex
      - bmi     body mass index
      - bp      average blood pressure
      - s1      tc, total serum cholesterol
      - s2      ldl, low-density lipoproteins
      - s3      hdl, high-density lipoproteins
      - s4      tch, total cholesterol / HDL
      - s5      ltg, possibly log of serum triglycerides level
      - s6      glu, blood sugar level

Note: Each of these 10 feature variables have been mean centered and scaled by the standard deviation times the
square root of `n_samples` (i.e. the sum of squares of each column totals 1).

Source URL:
https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html

For more information see:
Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angle Regression," Annals of S
tatistics (with discussion), 407-499.
(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)
```


✅ 2. 데이터 탐색(EDA)

In [4]:
```
# 데이터의 행/열 확인

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

```
(353, 11)
(89, 11)
(353, 2)
```

In [5]:
```
# 초기 데이터 확인

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

```
   cust_id       age       sex       bmi        bp        s1        s2  \
0        4  0.005383 -0.044642 -0.036385  0.021872  0.003935  0.015596
1      318  0.088931 -0.044642  0.006728  0.025315  0.030078  0.008707
2      301 -0.001882  0.050680 -0.024529  0.052858  0.027326  0.030001

         s3        s4        s5        s6
0  0.008142 -0.002592 -0.031988 -0.046641
1  0.063367 -0.039493  0.009434  0.032059
2  0.030232 -0.002592 -0.021395  0.036201
   cust_id       age       sex       bmi        bp        s1        s2  \
0      280  0.009016  0.050680  0.018584  0.039087  0.017694  0.010586
1      412  0.074401 -0.044642  0.085408  0.063187  0.014942  0.013091
2       68  0.038076  0.050680 -0.029918 -0.040099 -0.033216 -0.024174

         s3        s4        s5        s6
0  0.019187 -0.002592  0.016307 -0.017646
1  0.015505 -0.002592  0.006207  0.085907
2 -0.010266 -0.002592 -0.012909  0.003064
   cust_id  target
0        4   135.0
1      318   109.0
2      301    65.0
```


In [6]:
```
# 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: 353 entries, 0 to 352
Data columns (total 11 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   cust_id  353 non-null    int64  
 1   age      353 non-null    float64
 2   sex      353 non-null    float64
 3   bmi      353 non-null    float64
 4   bp       353 non-null    float64
 5   s1       353 non-null    float64
 6   s2       353 non-null    float64
 7   s3       353 non-null    float64
 8   s4       353 non-null    float64
 9   s5       353 non-null    float64
 10  s6       353 non-null    float64
dtypes: float64(10), int64(1)
memory usage: 30.5 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 89 entries, 0 to 88
Data columns (total 11 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   cust_id  89 non-null     int64  
 1   age      89 non-null     float64
 2   sex      89 non-null     float64
 3   bmi      89 non-null     float64
 4   bp       89 non-null     float64
 5   s1       89 non-null     float64
 6   s2       89 non-null     float64
 7   s3       89 non-null     float64
 8   s4       89 non-null     float64
 9   s5       89 non-null     float64
 10  s6       89 non-null     float64
dtypes: float64(10), int64(1)
memory usage: 7.8 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 353 entries, 0 to 352
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   cust_id  353 non-null    int64  
 1   target   353 non-null    float64
dtypes: float64(1), int64(1)
memory usage: 5.6 KB
None
```

In [7]:
```
# x_train 과 x_test 데이터의 기초통계량을 잘 비교해보세요.

print(x_train.describe())  # x_train.describe().T 둘중에 편한거 사용하세요
print(x_test.describe())
print(y_train.describe())
```

```
          cust_id         age         sex         bmi          bp          s1  \
count  353.000000  353.000000  353.000000  353.000000  353.000000  353.000000
mean   212.634561    0.000804    0.000724    0.000640   -0.000326    0.001179
std    126.668903    0.047617    0.047673    0.048141    0.046585    0.047891
min      0.000000   -0.107226   -0.044642   -0.084886   -0.112399   -0.126781
25%    105.000000   -0.038207   -0.044642   -0.035307   -0.033213   -0.033216
50%    210.000000    0.005383   -0.044642   -0.006206   -0.005670   -0.002945
75%    322.000000    0.038076    0.050680    0.030440    0.032201    0.027326
max    441.000000    0.110727    0.050680    0.170555    0.125158    0.153914

               s2          s3          s4          s5          s6
count  353.000000  353.000000  353.000000  353.000000  353.000000
mean     0.001110   -0.000452    0.000901    0.001446    0.000589
std      0.048248    0.048600    0.048045    0.047160    0.048122
min     -0.115613   -0.102307   -0.076395   -0.126097   -0.137767
25%     -0.029184   -0.039719   -0.039493   -0.033246   -0.034215
50%     -0.001314   -0.006584   -0.002592    0.000272    0.003064
75%      0.031567    0.030232    0.034309    0.033654    0.032059
max      0.198788    0.181179    0.185234    0.133597    0.135612
          cust_id         age         sex         bmi          bp          s1  \
count   89.000000   89.000000   89.000000   89.000000   89.000000   89.000000
mean   251.696629   -0.003188   -0.002871   -0.002537    0.001292   -0.004676
std    127.901365    0.047761    0.047563    0.045665    0.051777    0.046493
min      9.000000   -0.099961   -0.044642   -0.090275   -0.108956   -0.091006
25%    148.000000   -0.034575   -0.044642   -0.030996   -0.036656   -0.037344
50%    280.000000   -0.001882   -0.044642   -0.009439   -0.005670   -0.009825
75%    366.000000    0.030811    0.050680    0.034751    0.042529    0.031454
max    436.000000    0.096197    0.050680    0.137143    0.132044    0.119515

               s2          s3          s4          s5          s6
count   89.000000   89.000000   89.000000   89.000000   89.000000
mean    -0.004401    0.001792   -0.003575   -0.005737   -0.002334
std      0.045030    0.043723    0.045980    0.049252    0.045757
min     -0.089935   -0.080217   -0.076395   -0.104366   -0.129483
25%     -0.030437   -0.028674   -0.039493   -0.038460   -0.030072
50%     -0.014153   -0.002903   -0.002592   -0.014960   -0.005220
75%      0.020607    0.022869    0.003312    0.024055    0.019633
max      0.130208    0.122273    0.141322    0.133597    0.135612
          cust_id      target
count  353.000000  353.000000
mean   212.634561  152.943343
std    126.668903   75.324692
min      0.000000   37.000000
25%    105.000000   90.000000
50%    210.000000  141.000000
75%    322.000000  208.000000
max    441.000000  346.000000
```


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

```
   cust_id  target
0        4   135.0
1      318   109.0
2      301    65.0
3      189    79.0
4      288    80.0
```

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

```
         count        mean         std   min    25%    50%    75%    max
cust_id  353.0  212.634561  126.668903   0.0  105.0  210.0  322.0  441.0
target   353.0  152.943343   75.324692  37.0   90.0  141.0  208.0  346.0
```

✓ 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
age        0
sex        0
bmi        0
bp         0
s1         0
s2         0
s3         0
s4         0
s5         0
s6         0
dtype: int64
cust_id    0
age        0
sex        0
bmi        0
bp         0
s1         0
s2         0
s3         0
s4         0
s5         0
s6         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 필요하기 때문에 별도로 저장

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'])
```


데이터 분리

In [16]:
```
# 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)
```



```
(282, 10)
(71, 10)
(282,)
(71,)
```

✅ 4. 모델링 및 성능평가

In [17]:
```
# 랜덤포레스트 모델 사용 (참고 : 분류모델은 RandomForestClassifier)
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(random_state=2023)
model.fit(x_train, y_train)
```

Out[17]:
```
RandomForestRegressor(random_state=2023)
```

In [18]:
```
# 모델을 사용하여 테스트 데이터 예측
y_pred = model.predict(x_val)
```

In [19]:
```
# 모델 성능 평가 (평균 제곱 오차 및 R-squared)
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(y_val, y_pred)  # (실제값, 예측값)
r2 = r2_score(y_val, y_pred)             # (실제값, 예측값)
```

In [20]:
```
# MSE(mean_squared_error)
print(mse)
```

```
2563.5036816901406
```

In [21]:
```
# R2 score(R-squared)
print(r2)
```



```
0.5250663004710372
```

In [22]:
```
# RMSE
rmse = mse**0.5
print(rmse)
```

```
50.631054518843875
```

✅ 5. 예측값 제출

(주의) x_test 를 모델에 넣어 나온 예측값을 제출해야함

In [23]:
```
# (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      280  186.51
1      412  255.92
2       68   77.97
3      324  185.64
4      101  111.14
```

In [24]:
```
# ★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())
```