

데이터 출처 : <https://www.datamanim.com/dataset/ADPpb/00/31.html>



31회 ADP 기출문제

✓ 기계학습 (60점)

✓ 데이터 설명

- 데이터 출처 : <https://www.kaggle.com/datasets/mandysia/obesity-dataset-cleaned-and-data-sinthetic> 후처리
- 데이터 링크 : https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/31/adp_31_1_obesity_v2.csv
- 데이터 설명 : 각 환자의 의료정보이다. NObeyesdad를 종속변수로 하는 분류모델을 만드려고 한다.

id: unique id for each row
 Gender: sex - male or female
 Age: age
 Height: height
 Weight: weight
 family_history_with_overweight: Has a family member suffered or suffers f from overweight? - yes or no
 FAVC: Frequent consumption of high caloric food - yes or no
 FCVC: Frequency of consumption of vegetables - Never, Sometimes, Always
 NCP: Number of main meals - 1, 2, 3, 4
 CAEC: Consumption of food between meals - No, Sometimes, Frequently, Always
 SMOKE: Do you smoke - yes o no
 CH2O: Consumption of water daily - Less than a litter, between 1 and 2 l, more than 2 l
 SCC: Calories consumption monitoring - yes or no
 FAF: Physical activity frequency - 0, 1 to 2, 2 to 4, 4 to 5
 TUE: Time using technology devices - 0 to 2, 3 to 5, >5
 CALC: Consumption of alcohol - no, sometimes, frequently, always
 MTRANS: Transportation used - automobile, motorbike, bike, public_transportation, walking
 BMI: Body mass index

(종속변수)

NObeyesdad: Type of obesity - overweight-level_i, obesity_type_i, obesity_type_ii, obesity_type_iii

- id: 각 행의 고유 ID
- Gender: 성별 - 남성 또는 여성
- Age: 나이
- Height: 키
- Weight: 몸무게
- family_history_with_overweight: 가족 구성원이 과체중을 앓고 있거나 앓았는지 여부 - 예 또는 아니오
- FAVC: 고칼로리 음식의 빈번한 섭취 - 예 또는 아니오
- FCVC: 채소 섭취 빈도 - 전혀 안함, 가끔, 항상
- NCP: 주요 식사 횟수 - 1회, 2회, 3회, 4회
- CAEC: 식사 사이 음식 섭취 - 아니오, 가끔, 자주, 항상
- SMOKE: 흡연 여부 - 예 또는 아니오
- CH2O: 하루 물 섭취량 - 1리터 미만, 1~2리터, 2리터 이상
- SCC: 칼로리 섭취 모니터링 여부 - 예 또는 아니오
- FAF: 신체 활동 빈도 - 0회, 12회, 24회, 4~5회
- TUE: 기술 장치 사용 시간 - 02시간, 35시간, 5시간 이상
- CALC: 알코올 섭취 - 아니오, 가끔, 자주, 항상
- MTRANS: 사용된 교통 수단 - 자동차, 오토바이, 자전거, 대중교통, 걷기
- BMI: 체질량지수

(종속 변수)

- NObeyesdad: 비만 유형 - 과체중 1단계, 비만 1형, 비만 2형, 비만 3형

1-1 EDA & 결측치 및 이상치를 판단하고 설명하라

```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/31/adp_31_1_obesity_v2.csv', encoding = 'cp949')
df.head()
df.columns
df = df.drop('Unnamed: 0', axis = 1)
df
```

↻

	id	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FA
0	4	male	27	1.800000	87	no	no	always	3	sometimes	no	between 1 and 2	no	1
1	11	male	26	1.850000	105	yes	yes	always	3	frequently	no	more than 2	no	1
2	14	male	41	1.800000	99	no	yes	sometimes	3	sometimes	no	between 1 and 2	no	1
3	18	female	29	1.530000	78	no	yes	sometimes	1	sometimes	no	between 1 and 2	no	
4	20	female	23	1.650000	70	yes	no	sometimes	1	sometimes	no	between 1 and 2	no	
...
1257	2,107	female	21	1.710730	131	yes	yes	always	3	sometimes	no	between 1 and 2	no	1
1258	2,108	female	22	1.748584	134	yes	yes	always	3	sometimes	no	between 1 and 2	no	1
1259	2,109	female	23	1.752206	134	yes	yes	always	3	sometimes	no	between 1 and 2	no	1
1260	2,110	female	24	1.739450	133	yes	yes	always	3	sometimes	no	more than 2	no	1
1261	2,111	female	24	1.738836	133	yes	yes	always	3	sometimes	no	more than 2	no	1

1262 rows × 19 columns

```
df.info()
print('')
주어진 데이터의 각 특성을 살펴보았다.
데이터의 유형을 살펴보면 범주형 데이터와 수치형데이터로 이루어진 것을 확인할 수 있다.
특히, SCC 데이터에서 결측을 확인할 수 있었다.
''')
```

↻

<class 'pandas.core.frame.DataFrame'>				
RangeIndex: 1262 entries, 0 to 1261				
Data columns (total 19 columns):				
#	Column	Non-Null	Count	Dtype
0	id	1262	non-null	object
1	Gender	1262	non-null	object
2	Age	1262	non-null	int64
3	Height	1262	non-null	float64
4	Weight	1262	non-null	int64
5	family_history_with_overweight	1262	non-null	object
6	FAVC	1262	non-null	object
7	FCVC	1262	non-null	object
8	NCP	1262	non-null	int64
9	CAEC	1262	non-null	object
10	SMOKE	1262	non-null	object
11	CH2O	1262	non-null	object
12	SCC	1259	non-null	object
13	FAF	1262	non-null	object
14	TUE	1262	non-null	object
15	CALC	1262	non-null	object
16	MTRANS	1262	non-null	object
17	NObeyesdad	1262	non-null	object

```

18 BMI                                1262 non-null    float64
dtypes: float64(2), int64(3), object(14)
memory usage: 187.5+ KB

```

주어진 데이터의 각 특성을 살펴보았다.

데이터의 유형을 살펴보면 범주형 데이터와 수치형데이터로 이루어진 것을 확인할 수 있다.
특히, SCC 데이터에서 결측치를 확인할 수 있었다.

```

print('''
주어진 데이터와 수치형 데이터의 항목별 값 분포를 알아보고, 수치형 변수의 n-percentile 값과
최대,최소 값들을 알 수 있었으며, 아래와 같이 전체적인 분포를 알아볼 수 있었다.
''')
print(df.describe())
import matplotlib.pyplot as plt
numeric_df = df.select_dtypes(exclude = 'object')
print('수치형 변수 분포 시각화')
df.hist(color = 'darkblue', figsize= (8,8), grid = False)
plt.show()

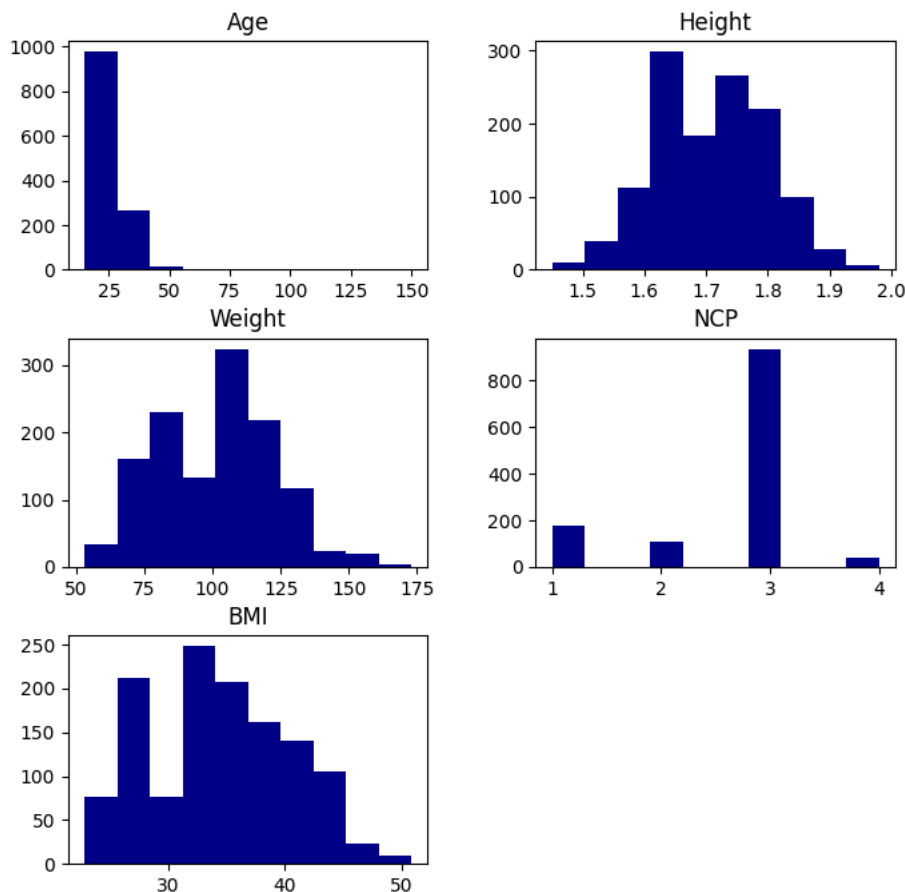
```



주어진 데이터와 수치형 데이터의 항목별 값 분포를 알아보고, 수치형 변수의 n-percentile 값과
최대,최소 값들을 알 수 있었으며, 아래와 같이 전체적인 분포를 알아볼 수 있었다.

	Age	Height	Weight	NCP	BMI
count	1262.000000	1262.000000	1262.000000	1262.000000	1262.000000
mean	25.749604	1.709184	101.083201	2.665610	34.439463
std	9.898088	0.091361	21.520848	0.757143	6.106623
min	15.000000	1.450000	53.000000	1.000000	22.826739
25%	21.000000	1.631576	82.000000	3.000000	30.725995
50%	24.000000	1.711095	105.000000	3.000000	34.332001
75%	27.000000	1.775768	116.000000	3.000000	38.920119
max	150.000000	1.980000	173.000000	4.000000	50.811753

수치형 변수 분포 시각화



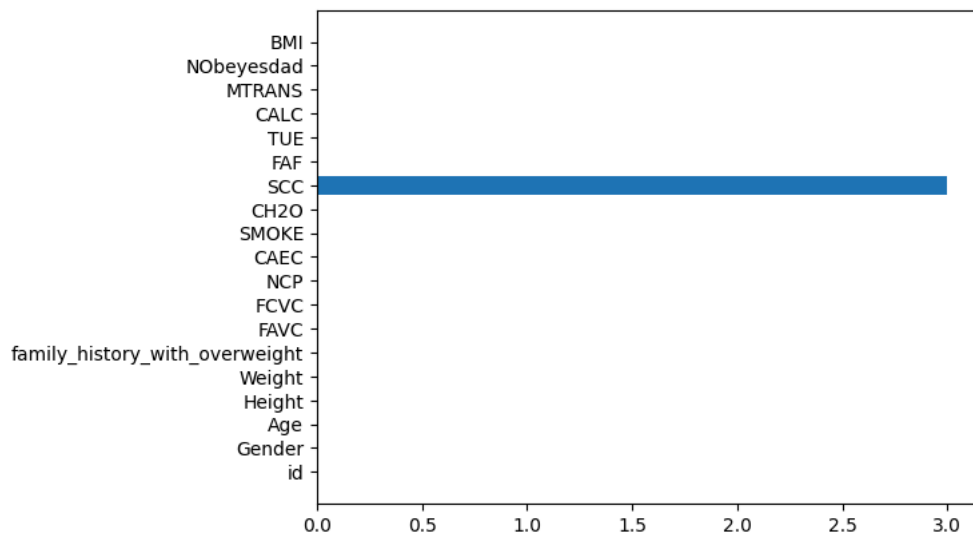
```

print('결측치 및 이상치 확인')
print('Nn 결측치 확인')
missing_data = df.isnull().sum()
plt.barh(missing_data.index, missing_data.values)
plt.show()
print('SCC데이터에 결측치가 있음을 확인할 수 있다.')

```

↔ 결측치 및 이상치 확인

결측치 확인

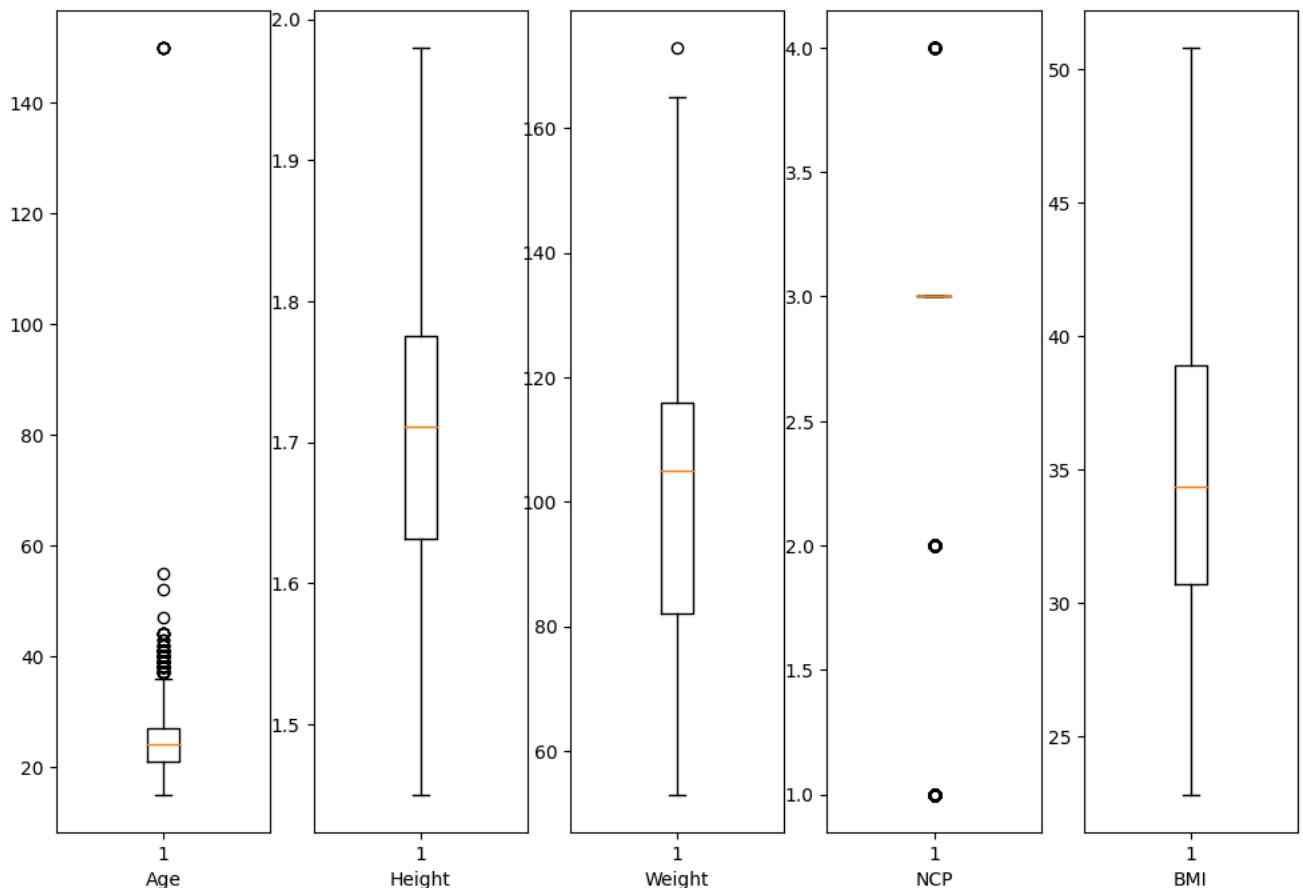


SCC데이터에 결측치가 있음을 확인할 수 있다.

```
print('이상치 확인')
print('')
이상치의 유무를 확인하기 위해 수치형 변수의 시각화를 진행하였다.
시각화 결과, 이상치로 보이는 변수 Age, Weight, NCP를 확인할 수 있다.
''')
numeric = df.select_dtypes(exclude = 'object').columns
fig, axes = plt.subplots(1,5, figsize = (12,8))
for ax, col in zip(axes.ravel(), df[numeric]):
    if col in numeric:
        ax.boxplot(df[col])
        ax.set_xlabel(col)
plt.show()
```

↔ 이상치 확인

이상치의 유무를 확인하기 위해 수치형 변수의 시각화를 진행하였다.
시각화 결과, 이상치로 보이는 변수 Age, Weight, NCP를 확인할 수 있다.



```
print('결측 및 이상치 처리')
df['SCC'] = df['SCC'].fillna(df['SCC'].mode()[0])

numeric = df.select_dtypes(exclude = 'object').columns

for col in numeric :
    IQR = df[col].quantile(0.75) - df[col].quantile(0.25)
    top = df[col].quantile(0.75) + IQR * 1.5
    down = df[col].quantile(0.25) - IQR * 1.5
    df[col] = df[col].clip(down, top)
```

 결측 및 이상치 처리

```
df['SCC'].mode()[0]
```


 'no'

✓ 1-2 데이터 전처리 기법 2가지를 설명하고 주어진 데이터에 적용시 어떤 효과가 있는지 설명하라

```
print('''
데이터 전처리 기법 2가지
```

1. 원핫 인코딩: 여러값 중 하나만 활성화하는 인코딩이다.
머신러닝 모델은 문자 데이터를 이해하지 못하기 때문에 문자로 구성된 범주형 데이터를 숫자로 바꿀 때 사용한다.
효과 : 현재 데이터는 object 타입의 범주형 데이터가 많으므로, 원 핫 인코딩을 통해 머신러닝 모델에 넣을 수 있는 데이터 형태로 변환할 수 있게된다.


2. 정규화 : 연속형 데이터를 특정 범위 내로 스케일 조정하는 방법이다.
주로 데이터의 범위를 0과 1사이로 제한할 때 사용할 수 있으며, 데이터의 상대적인 크기와 중요성을 유지할 수 있다.
효과 : 현재 데이터는 각 컬럼별 수치 범위가 다양하기 때문에 정규화를 통해 데이터를 0과 1사이로 맞추는 정규화를 진행할 수 있다.
''')

 데이터 전처리 기법 2가지

1. 원핫 인코딩: 여러값 중 하나만 활성화하는 인코딩이다.
머신러닝 모델은 문자 데이터를 이해하지 못하기 때문에 문자로 구성된 범주형 데이터를 숫자로 바꿀 때 사용한다.
효과 : 현재 데이터는 object 타입의 범주형 데이터가 많으므로, 원 핫 인코딩을 통해 머신러닝 모델에 넣을 수 있는 데이터 형태로 변환할 수 있게된다.


2. 정규화 : 연속형 데이터를 특정 범위 내로 스케일 조정하는 방법이다.
주로 데이터의 범위를 0과 1사이로 제한할 때 사용할 수 있으며, 데이터의 상대적인 크기와 중요성을 유지할 수 있다.
효과 : 현재 데이터는 각 컬럼별 수치 범위가 다양하기 때문에 정규화를 통해 데이터를 0과 1사이로 맞추는 정규화를 진행할 수 있다.

```
y = df['NObeyesdad']
X = df.drop('NObeyesdad', axis = 1)
X.columns
```

 Index(['id', 'Gender', 'Age', 'Height', 'Weight',
 'family_history_with_overweight', 'FAVC', 'FCVC', 'NCP', 'CAEC',
 'SMOKE', 'CH2O', 'SCC', 'FAF', 'TUE', 'CALC', 'MTRANS', 'BMI'],
 dtype='object')

```
X = X.drop('id',axis = 1)
```

```
cat_df = X.select_dtypes(include = 'object').columns
X = pd.get_dummies(X,columns = cat_df)
X
```




	Age	Height	Weight	NCP	BMI	Gender_female	Gender_male	family_history_with_overweight_no	family_history_with_ov
0	27	1.800000	87	3	26.851852	False	True		True
1	26	1.850000	105	3	30.679328	False	True		False
2	36	1.800000	99	3	30.555556	False	True		True
3	29	1.530000	78	3	33.320518	True	False		True
4	23	1.650000	70	3	25.711662	True	False		False
...
1257	21	1.710730	131	3	44.901475	True	False		False
1258	22	1.748584	134	3	43.741923	True	False		False
1259	23	1.752206	134	3	43.543817	True	False		False
1260	24	1.739450	133	3	44.071535	True	False		False
1261	24	1.738836	133	3	44.144338	True	False		False

1262 rows × 40 columns

1-3 피쳐 엔지니어링을 통해 파생변수 1개를 생성하고 그 이유를 말하라

```
#


import numpy as np
df['Age_BMI_ratio'] = df['Age']/df['BMI']
df['Age_BMI_ratio'] = df['Age_BMI_ratio'].replace([np.inf, -np.inf], 1)
print('주어진 데이터를 토대로 age와 BMI 두개의 특성을 이용해서 Age_BMI_ratio라는 새로운 특성을 추가하였다')
print('이 새로운 특성은 Age를 BMI로 나눈 값이다. 이를 추가한 이유는 BMI대비 나이가 높을수록 비만정도가 높다는 가설을 확인하기 위함이다.')
```

 주어진 데이터를 토대로 age와 BMI 두개의 특성을 이용해서 Age_BMI_ratio라는 새로운 특성을 추가하였다
이 새로운 특성은 Age를 BMI로 나눈 값이다. 이를 추가한 이유는 BMI대비 나이가 높을수록 비만정도가 높다는 가설을 확인하기 위함이다.

2-1 앙상블을 제외한 분류 모델 3가지 구축 및 결과 비교 및 설명하라

```
print('''
1. 결정트리.
2. SVM
3. KNN을 사용하여 분류해보았다.
''')
```

```
y.replace('overweight_level_i', 0,inplace = True)
y.replace('obesity_type_i', 1,inplace = True)
y.replace('obesity_type_ii', 2,inplace = True)
y.replace('obesity_type_iii', 3,inplace = True)
y
```

 1. 결정트리.
2. SVM
3. KNN을 사용하여 분류해보았다.

```
0      0
1      1
2      1
3      1
4      0
...
1257   3
1258   3
1259   3
1260   3
1261   3
Name: NObeyesdad, Length: 1262, dtype: int64
```

```

print('결정트리')
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X_train,X_test,y_train,y_test = train_test_split(X,y, test_size = 0.2, random_state=10)
model_dic = DecisionTreeClassifier()
model_dic.fit(X_train,y_train)
y_pred_decision = model_dic.predict(X_test)
accuracy = accuracy_score(y_test, y_pred_decision)
print('결정트리 정확도 : ',accuracy)

print('KNN ')
from sklearn.neighbors import KNeighborsClassifier
model_knn = KNeighborsClassifier(n_neighbors=3)
model_knn.fit(X_train,y_train)
y_pred_knn = model_knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred_knn)
print('knn 정확도 : ', accuracy)

from sklearn.svm import SVC
print('SVM ')
model_svm = SVC()
model_svm.fit(X_train,y_train)
y_pred_svm = model_svm.predict(X_test)
accuracy = accuracy_score(y_test, y_pred_svm)
print('svm 정확도 : ', accuracy)

```

```

결정트리
결정트리 정확도 : 0.9881422924901185
KNN
knn 정확도 : 0.9920948616600791
SVM
svm 정확도 : 0.8893280632411067

```

print('knn의 정확도가 0.99가 나와서 가장 높은 정확도를 보였고 그 다음으로 결정트리의 정확도가 0.98로 높았으며, svm이 세 모델 중에서 성능은 가장 낮은 것'
 print('따라서 knn모델이 가장 높은 정확도를 가지고 있으므로 가장 좋은 모델이라고 판단할 수 있다.')

knn의 정확도가 0.99가 나와서 가장 높은 정확도를 보였고 그 다음으로 결정트리의 정확도가 0.98로 높았으며, svm이 세 모델 중에서 성능은 가장 낮은 것'
 따라서 knn모델이 가장 높은 정확도를 가지고 있으므로 가장 좋은 모델이라고 판단할 수 있다.

2-2 2-1에서 사용한 모델 중 하나를 골라 그리드 서치를 통해서 파라미터 튜닝 및 분류 모델 성능 평가 (precision ,recall)

```

from sklearn.model_selection import GridSearchCV

param_grid = {'max_depth' : [None, 10,20,30],
              'min_samples_split' : [2,4]}
dic_gird_search = GridSearchCV(estimator = model_dic, param_grid= param_grid, cv = 5, scoring = 'accuracy')
dic_gird_search.fit(X_train, y_train)

pred = dic_gird_search.predict(X_test)
from sklearn.metrics import classification_report
print(classification_report(y_test,pred))

```

```

precision    recall  f1-score   support

0           1.00      1.00      1.00         60
1           1.00      1.00      1.00         68
2           1.00      1.00      1.00         52
3           1.00      1.00      1.00         73

accuracy          1.00      1.00      1.00        253
macro avg          1.00      1.00      1.00        253
weighted avg          1.00      1.00      1.00        253

```

2-3 2-1의 3가지 모델을 soft voting을 이용하여 모델링 한 결과와 2-2과 비교하라

```
from sklearn.ensemble import VotingClassifier
```

```
model_svm = SVC(probability= True)
ensemble_model = VotingClassifier(estimators = [('knn', model_knn), ('svm', model_svm), ('dic' , model_dic)], voting = 'soft')
```

```
ensemble_model.fit(X_train, y_train)
y_pred = ensemble_model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print('앙상블(보팅) 정확도 : ', accuracy)
```

↻ 앙상블(보팅) 정확도 : 0.9960474308300395

```
print('2-2의 모델과 소프트 보팅을 사용한 2-3의 결과물을 비교한 결과 2-2의 모델이 더 좋은 성능을 내는 것을 확인할 수 있다.')
```

↻ 2-2의 모델과 소프트 보팅을 사용한 2-3의 결과물을 비교한 결과 2-2의 모델이 더 좋은 성능을 내는 것을 확인할 수 있다.

3-2 하나의 모델을 선정하여 Drop-Column Importance 방법으로 특성의 중요도(feature importance)를 산출하여 제시

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

```
model = DecisionTreeClassifier(random_state = 10)
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
baseline = accuracy_score(y_test, y_pred)
```

```
drop_importances = []
for col in X_train.columns:
    X_train_drop = X_train.drop(columns = [col])
    X_test_drop = X_test.drop(columns = [col])
```

```
model_drop = DecisionTreeClassifier(random_state = 10)
model_drop.fit(X_train_drop,y_train)
```

```
y_pred_drop = model_drop.predict(X_test_drop)
drop_accuracy = accuracy_score(y_test, y_pred_drop)
```

```
drop_importances.append((col, baseline - drop_accuracy))
```

```
print(drop_importances)
importance_df = pd.DataFrame(drop_importances, columns = ['Feature','Importance'])
```

↻ [('Age', 0.011857707509881465), ('Height', 0.0), ('Weight', 0.0), ('NCP', 0.0039525691699604515), ('BMI', 0.007905138339921014), ('Gender_female

importance_df



	Feature	Importance
0	Age	0.011858
1	Height	0.000000
2	Weight	0.000000
3	NCP	0.003953
4	BMI	0.007905
5	Gender_female	0.000000
6	Gender_male	0.000000
7	family_history_with_overweight_no	0.000000
8	family_history_with_overweight_yes	0.000000
9	FAVC_no	-0.003953
10	FAVC_yes	-0.003953
11	FCVC_always	0.000000
12	FCVC_never	0.000000
13	FCVC_sometimes	0.000000
14	CAEC_always	0.000000
15	CAEC_frequently	0.000000
16	CAEC_no	0.003953
17	CAEC_sometimes	0.000000
18	SMOKE_no	0.000000
19	SMOKE_yes	0.000000
20	CH2O_between 1 and 2 l	-0.003953
21	CH2O_less than a liter	-0.003953
22	CH2O_more than 2 l	0.007905
23	SCC_no	-0.003953
24	SCC_yes	-0.003953
25	FAF_0	-0.003953
26	FAF_1 to 2	0.003953
27	FAF_2 to 4	0.003953
28	FAF_4 to 5	-0.003953
29	TUE_0 to 2	0.003953
30	TUE_3 to 5	0.003953

데이터 설명

- 데이터 출처 : 자체제작
- 데이터 링크 : https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/31/adp_31_2_v2.csv
- 데이터 설명 : 중고등학생 건강검진 데이터

4-1 아래의 기준으로 전처리를 하여 걱정 체중 여부 컬럼을 생성하고 걱정 체중여부에 대한 빈도표를 만들어라

bmi 계산 - $Bmi = \text{몸무게(kg)} / (\text{키(m)})^2$

만나이 구하기 - 건강검진을 받았던 날을 기준으로 생년월일과 일수 차이가 16년 364일 이하인 경우 만 16세 그 이상의 경우 만 17로 분류하라 - 윤년 등은 고려하지 않는다. 햇수로 16년 + 일수로 364일이 기준이다

걱정 체중 여부 (BMI가 아래 구간에 들어올 경우)

17세 남자 : 21.03이상 23.21 미만

17세 여자 : 20.03이상 22.39 미만

16세 남자 : 21.18이상 23.45 미만

16세 여자 : 19.61이상 21.74 미만

```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/31/adp_31_2_v2.csv')
df.head()
```

	ID	키	weight	생년월일	건강검진일	공학여부	채소섭취정도	아침식사여부	일주일운동시간	수면시간	성별
0	ID_4135	169.01	65.47	20041003	2020_11_15	1	2	1	4.4	8.3	남성
1	ID_3289	181.62	69.36	19970725	2014_11_20	0	3	0	4.4	6.9	남성
2	ID_1847	160.89	65.12	20020921	2020_01_28	1	1	1	1.7	9.6	여성
3	ID_4785	162.21	62.28	20020106	2018_09_27	1	4	0	5.1	6.8	남성
4	ID_5693	159.13	54.11	19980708	2015_03_03	0	4	1	0.3	8.5	여성

```
df['생년월일포맷변환'] = pd.to_datetime(df['생년월일'], format = "%Y%m%d")
df['측정일자포맷변환'] = pd.to_datetime(df['건강검진일'], format = "%Y_%m_%d")

df['만나이'] = (df['측정일자포맷변환'] - df['생년월일포맷변환']).dt.days // 365
df
```

	ID	키	weight	생년월일	건강검진일	공학여부	채소섭취정도	아침식사여부	일주일운동시간	수면시간	성별	생년월일포맷변환	측정일자포맷변환	만나이
0	ID_4135	169.01	65.47	20041003	2020_11_15	1	2	1	4.4	8.3	남성	2004-10-03	2020-11-15	16
1	ID_3289	181.62	69.36	19970725	2014_11_20	0	3	0	4.4	6.9	남성	1997-07-25	2014-11-20	17
2	ID_1847	160.89	65.12	20020921	2020_01_28	1	1	1	1.7	9.6	여성	2002-09-21	2020-01-28	17
3	ID_4785	162.21	62.28	20020106	2018_09_27	1	4	0	5.1	6.8	남성	2002-01-06	2018-09-27	16
4	ID_5693	159.13	54.11	19980708	2015_03_03	0	4	1	0.3	8.5	여성	1998-07-08	2015-03-03	16
...
6796	ID_6443	156.04	55.36	20030429	2020_04_23	1	4	1	7.8	9.3	여성	2003-04-29	2020-04-23	16
6797	ID_3606	182.46	67.55	20050528	2022_08_03	1	4	0	2.9	8.9	남성	2005-05-28	2022-08-03	17

```
df.만나이.unique()

array([16, 17])

# bmi 계산 - Bmi = 몸무게(kg)/((키(m)**2)
df['BMI'] = df['weight']/((df['키']/100)**2)
df.head()
```

	ID	키	weight	생년월일	건강검진일	공학여부	채소섭취정도	아침식사여부	일주일운동시간	수면시간	성별	생년월일포맷변환	측정일자포맷변환	만나이	BMI
0	ID_4135	169.01	65.47	20041003	2020_11_15	1	2	1	4.4	8.3	남성	2004-10-03	2020-11-15	16	22.920154
1	ID_3289	181.62	69.36	19970725	2014_11_20	0	3	0	4.4	6.9	남성	1997-07-25	2014-11-20	17	21.027214
2	ID_1847	160.89	65.12	20020921	2020_01_28	1	1	1	1.7	9.6	여성	2002-09-21	2020-01-28	17	25.156852

```
...
적정 체중 여부 (BMI가 아래 구간에 들어올 경우)
16세 남자 : 21.18이상 23.45 미만
16세 여자 : 19.61이상 21.74 미만
17세 남자 : 21.03이상 23.21 미만
17세 여자 : 20.03이상 22.39 미만
...
```

```
df.loc[(df['성별'] == '남성') & (df['만나이'] == 16) & (df['BMI']>=21.18) & (df['BMI']<23.45), '적정체중여부'] = '적절'
df.loc[(df['성별'] == '여성') & (df['만나이'] == 16) & (df['BMI']>=19.61) & (df['BMI']<21.74), '적정체중여부'] = '적절'
df.loc[(df['성별'] == '남성') & (df['만나이'] == 17) & (df['BMI']>=21.03) & (df['BMI']<23.21), '적정체중여부'] = '적절'
df.loc[(df['성별'] == '여성') & (df['만나이'] == 17) & (df['BMI']>=20.03) & (df['BMI']<22.39), '적정체중여부'] = '적절'

df.loc[df['적정체중여부'] != '적절', '적정체중여부'] = '부적절'

df.head()
```

↻

	ID	키	weight	생년월일	건강검진일	공학여부	채소섭취정도	아침식사여부	일주일운동시간	수면시간	성별	생년월일포맷변환	측정일자포맷변환	나이	BMI	적정체중여부
0	ID_4135	169.01	65.47	20041003	2020_11_15	1	2	1	4.4	8.3	남성	2004-10-03	2020-11-15	16	22.920154	적절
1	ID_3289	181.62	69.36	19970725	2014_11_20	0	3	0	4.4	6.9	남성	1997-07-25	2014-11-20	17	21.027214	부적절
2	ID_1847	160.89	65.12	20020921	2020_01_28	1	1	1	1.7	9.6	여성	2002-09-21	2020-01-28	17	25.156852	부적절

```
df['적정체중여부'].value_counts()
```

↻

```
적정체중여부
적절      3993
부적절    2808
Name: count, dtype: int64
```

✓ 4-2 4-1에서 구한 적정 체중 여부와 나머지 컬럼(공학여부, 아침식사여부, 일주일운동시간, 채소섭취정도, 수면시간, 성별) 이 독립적인지 통계적으로 확인하라

```
from scipy.stats import chi2_contingency

columns_to_test = ['공학여부', '아침식사여부', '일주일운동시간', '채소섭취정도', '수면시간', '성별']

result = []
for column in columns_to_test:
    contingency_table = pd.crosstab(df['적정체중여부'], df[column])
    chi2, p, _, _ = chi2_contingency(contingency_table)
    result.append([chi2,p])

result_df = pd.DataFrame(result, columns = ['chi', 'p_value'], index = columns_to_test)
result_df
```

↻

	chi	p_value
공학여부	4.542368	0.033066
아침식사여부	0.152151	0.696488
일주일운동시간	115.745222	0.119871
채소섭취정도	10.143194	0.038083
수면시간	46.992431	0.554877
성별	0.002345	0.961379

```
print('''
검정 결과 p-value값이 0.05이상이면 해당 변수는 적정체중여부와 독립적이고
0.05이하이면 독립적이지 않다고 판단할 수 있다.

따라서 결과를 확인해보면 공학여부, 채소섭취정도는 독립적이지 않은 변수이며
나머지 컬럼, 즉 아침식사여부, 일주일운동시간, 수면시간, 성별은 적정 체중여부와 독립적인 것을 확인할 수 있었다.
''')
```

↻

```
검정 결과 p-value값이 0.05이상이면 해당 변수는 적정체중여부와 독립적이고
0.05이하이면 독립적이지 않다고 판단할 수 있다.

따라서 결과를 확인해보면 공학여부, 채소섭취정도는 독립적이지 않은 변수이며
나머지 컬럼, 즉 아침식사여부, 일주일운동시간, 수면시간, 성별은 적정 체중여부와 독립적인 것을 확인할 수 있었다.
```

✓ 4-3 4-2에서 유의한 변수들만 가지고 적정 체중 여부를 예측하는 모델을 구현하고 성능 평가 및 해석을 하라 (로지스틱회귀 ,xgb)

로지스틱회귀 모델은 오즈비를 구하라 xgb의 경우 각 피쳐중요도를 확인하고 예측에 영향을 가장 미치는 변수를 확인하라

```

from sklearn.linear_model import LogisticRegression
import xgboost as xgb
import numpy as np

df.loc[df['적정체중여부'] == '적절', '적정체중여부'] = 1
df.loc[df['적정체중여부'] == '부적절', '적정체중여부'] = 0
X = pd.get_dummies(df[['공학여부', '채소섭취정도']])
y = df['적정체중여부']
y = y.astype(int)

logistic_model = LogisticRegression()
logistic_model.fit(X,y)

xgboost_model = xgb.XGBClassifier()
xgboost_model.fit(X,y)

logistic_coefs = logistic_model.coef_[0]
xgboost_importance = xgboost_model.feature_importances_

print('로지스틱 모델 오즈비 : ', np.exp(logistic_coefs))
print('xgboost 모델 피쳐 중요도 : ',xgboost_importance )

↗ 로지스틱 모델 오즈비 : [1.11204771 1.00557371]
xgboost 모델 피쳐 중요도 : [0.7267853 0.27321473]

print(''''
로지스틱 모델의 오즈비는 공학여부의 경우 1.11204771 , 채소섭취정도는 1.00557371로 측정되었고
XGBoost 모델의 피쳐중요도는 공학여부의 경우 0.7267853, 채소섭취정도는 0.27321473가 나왔다.
xgboost의 경우 예측에 가장 많이 영향을 미치는 컬럼은 공학여부로 확인할 수 있었다.
''')
)

↗ 로지스틱 모델의 오즈비는 공학여부의 경우 1.11204771 , 채소섭취정도는 1.00557371로 측정되었고
XGBoost 모델의 피쳐중요도는 공학여부의 경우 0.7267853, 채소섭취정도는 0.27321473가 나왔다.
xgboost의 경우 예측에 가장 많이 영향을 미치는 컬럼은 공학여부로 확인할 수 있었다.

```

✓ 4-4 4-3 두 모델의 roc-auc 그래프를 하나의 그래프에 겹쳐 그려라

```

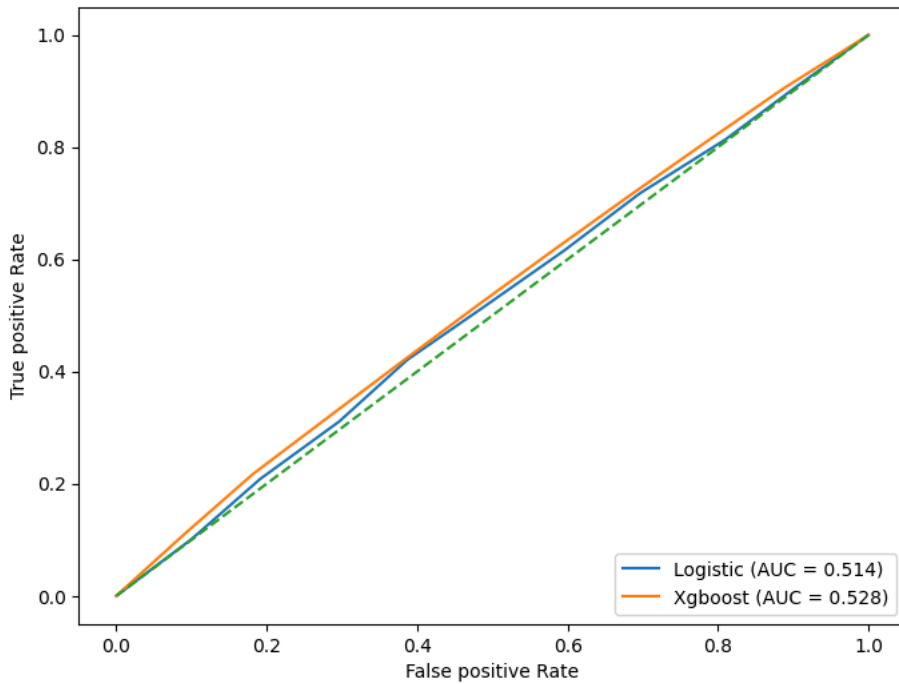
from sklearn.metrics import roc_auc_score, roc_curve
import matplotlib.pyplot as plt

y_logistic_prob = logistic_model.predict_proba(X)[:,-1]
fpr_logistic, tpr_logistic, _ = roc_curve(y,y_logistic_prob)
roc_auc_logistic = roc_auc_score(y,y_logistic_prob)

y_xgboost_prob = xgboost_model.predict_proba(X)[:,-1]
fpr_xgboost, tpr_xgboost, _ = roc_curve(y,y_xgboost_prob)
roc_auc_xgboost = roc_auc_score(y,y_xgboost_prob)

plt.figure(figsize = (8,6))
plt.plot(fpr_logistic, tpr_logistic , label = f'Logistic (AUC = {roc_auc_logistic:.3f})')
plt.plot(fpr_xgboost, tpr_xgboost , label = f'Xgboost (AUC = {roc_auc_xgboost:.3f})')
plt.plot([0,1],[0,1], linestyle = '--')
plt.xlabel('False positive Rate')
plt.ylabel('True positive Rate')
plt.legend(loc = 'lower right')
plt.show()

```



✓ 통계 (40점)

5. 평균이 θ 이고 분산이 100인 정규분포의 사전분포가 (100, 256)일때 120의 값을 가지는 데이터가 있을 경우 사후평균은? (5점)

무게가 $N(\theta, 100)$ 인 정규분포에서, 사전분포는 $N(100, 256)$ 이다. 제품의 무게가 120kg일 때, 사후분포의 평균을 구하라 (단, 소수점 3째자리까지 구하라)

```
mu_prior = 100
sigma_prior_squared = 256
```

```
sigma_data_squared = 100
x_observed = 120
```

```
mu_posterior = (sigma_data_squared * mu_prior + sigma_prior_squared * x_observed) / (sigma_data_squared + sigma_prior_squared)
round(mu_posterior, 3)
```

114.382

데이터 설명

- 데이터 출처 : <https://www.kaggle.com/datasets/yasserh/advertising-sales-dataset> 후처리
- 데이터 링크 : https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/31/adp_31_5_advertising.csv
- 데이터 설명 : TV, Radio, Newspaper에 각각 광고비(달러)를 다르게 했을때 매출액 (Sales, 밀리언달러)를 나타내는 데이터
- 종속변수 : Sales

✓ 6-1 회귀 모델링 후 유의하지 않는 변수 파악 (15점)

```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/31/adp_31_5_advertising.csv')
df.head()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype
---  ---
0    TV          200 non-null    float64
1    Radio       200 non-null    float64
2    Newspaper   200 non-null    float64
3    Sales       200 non-null    float64
```

```
dtypes: float64(4)
memory usage: 6.4 KB

y = df['Sales']
X = df[['TV', 'Radio', 'Newspaper']]

import statsmodels.api as sm
X = sm.add_constant(X)
model = sm.OLS(y,X).fit()

print(model.summary())
print('Newspaper 변수가 유의하지 않음')
```

↻

OLS Regression Results

Dep. Variable:	Sales	R-squared:	0.903
Model:	OLS	Adj. R-squared:	0.901
Method:	Least Squares	F-statistic:	605.4
Date:	Sun, 26 May 2024	Prob (F-statistic):	8.13e-99
Time:	13:37:07	Log-Likelihood:	-383.34
No. Observations:	200	AIC:	774.7
Df Residuals:	196	BIC:	787.9
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
const	4.6251	0.308	15.041	0.000	4.019	5.232
TV	0.0544	0.001	39.592	0.000	0.052	0.057
Radio	0.1070	0.008	12.604	0.000	0.090	0.124
Newspaper	0.0003	0.006	0.058	0.954	-0.011	0.012

Omnibus:	16.081	Durbin-Watson:	2.251
Prob(Omnibus):	0.000	Jarque-Bera (JB):	27.655
Skew:	-0.431	Prob(JB):	9.88e-07
Kurtosis:	4.605	Cond. No.	454.

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Newspaper 변수가 유의하지 않음

✓ 6-2 변수 선택시 먼저 제거 될 변수 및 근거 제시

```
print('')
pvalue 값이 0.05이하일 때 유의미한 변수로 진단할 수 있다
현재 Newspaper를 제외한 나머지 변수들은 유의미한 것을 알 수 있고, 같은 이유로 Newspaper가 변수 선택시 먼저 제거해야함을 알 수 있습니다
''')
```

↻

pvalue 값이 0.05이하일 때 유의미한 변수로 진단할 수 있다

현재 Newspaper를 제외한 나머지 변수들은 유의미한 것을 알 수 있고, 같은 이유로 Newspaper가 변수 선택시 먼저 제거해야함을 알 수 있습니다

✓ 6-3 VIF를 통한 다중공선성 진단

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif = pd.DataFrame()
vif['변수'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

print(vif)
print('일반적으로 VIF값이 100이 넘어가면 다중공선성 문제가 있다고 판단한다. 그런데 결과를 보면 모든 변수가 다중 공선성이 없는 것을 확인할 수 있다.')
```

↻

	변수	VIF
0	const	6.848900
1	TV	1.004611
2	Radio	1.144952
3	Newspaper	1.145187

일반적으로 VIF값이 100이 넘어가면 다중공선성 문제가 있다고 판단한다. 그런데 결과를 보면 모든 변수가 다중 공선성이 없는 것을 확인할 수 있다.

✓ 7 (15점)

- 데이터 링크 : https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/31/adp_31_7.csv
- 데이터 설명 : A,B,C,D,E 영업사원의 각 계약 성사 유무 (1:계약, 0:미계약) 를 나타낸 데이터이다. 영업사원의 평균 계약 성사 건수는 같은지 통계 검정하라

```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/Datamanim/datarepo/main/adp/31/adp_31_7.csv')
df.head()
```