# Alchemist 3기 아이디어톤

df\_pig 조

#### 1. 데이터 확인

①pd.read\_csv로 데이터를 불러오고 head, info, describe()를 통해 데이터의 형태를 확인

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC
0	Female	21.000000	1.620000	64.000000	yes	no	2.0	3.0	Sometimes
1	Female	21.000000	1.520000	56.000000	yes	no	3.0	3.0	Sometimes
2	Male	23.000000	1.800000	77.000000	yes	no	2.0	3.0	Sometimes
3	Male	27.000000	1.800000	87.000000	no	no	3.0	3.0	Sometimes
4	Male	22.000000	1.780000	89.800000	no	no	2.0	1.0	Sometimes
2106	Female	20.976842	1.710730	131.408528	yes	yes	3.0	3.0	Sometimes
2107	Female	21.982942	1.748584	133.742943	yes	yes	3.0	3.0	Sometimes
2108	Female	22.524036	1.752206	133.689352	yes	yes	3.0	3.0	Sometimes
2109	Female	24.361936	1.739450	133.346641	yes	yes	3.0	3.0	Sometimes
2110	Female	23.664709	1.738836	133.472641	yes	yes	3.0	3.0	Sometimes

RangeIndex: 2111 entries, 0 to 2110 Data columns (total 17 columns): Non-Null Count Dtype Column object Gender 2111 non-null Age 2111 non-null float64 Height 2111 non-null float64 Weight 2111 non-null float64 family history with overweight 2111 non-null object FAVC object 2111 non-null FCVC 2111 non-null float64 NCP 2111 non-null float64 CAEC 2111 non-null object SMOKE 2111 non-null object 2111 non-null float64 10 CH20 object 11 SCC 2111 non-null 12 FAF 2111 non-null float64 13 TUE 2111 non-null float64 14 CALC 2111 non-null object 15 MTRANS 2111 non-null object 2111 non-null object 16 NObeyesdad dtypes: float64(8), object(9)

<class 'pandas.core.frame.DataFrame'>

------ 200 Ft ME

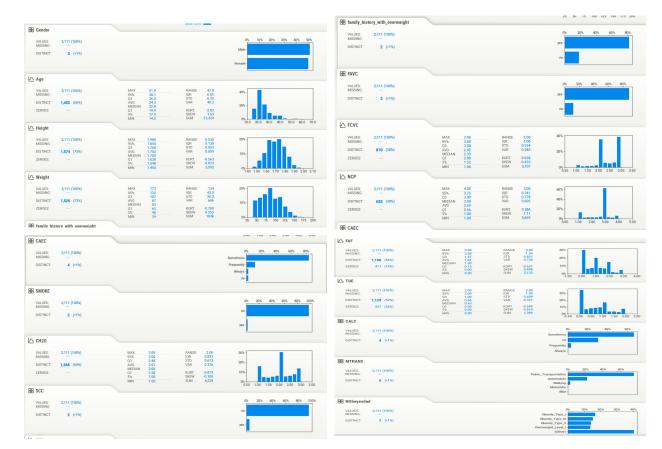
2111 rows × 17 columns

pig.describe()

	Age	Height	Weight	FCVC	NCP	CH2O	FAF	TUE
count	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000
mean	24.312600	1.701677	86.586058	2.419043	2.685628	2.008011	1.010298	0.657866
std	6.345968	0.093305	26.191172	0.533927	0.778039	0.612953	0.850592	0.608927
min	14.000000	1.450000	39.000000	1.000000	1.000000	1.000000	0.000000	0.000000
25%	19.947192	1.630000	65.473343	2.000000	2.658738	1.584812	0.124505	0.000000
50%	22.777890	1.700499	83.000000	2.385502	3.000000	2.000000	1.000000	0.625350
75%	26.000000	1.768464	107.430682	3.000000	3.000000	2.477420	1.666678	1.000000
max	61.000000	1.980000	173.000000	3.000000	4.000000	3.000000	3.000000	2,000000

## ②데이터에서 null값은 존재하지 않고 일부 칼럼이 비정형 데이터임을 확인

# ③sweetviz를 통한 데이터 시각화



## ④smote로 인한 문제 확인

```
pig[(['Age', 'FCVC', 'NCP', 'TUE'])].value counts()
          FCVC
                         TUE
Age
                NCP
21.000000
          2.0
                1.000000
                         0.000000
                                     18
18.000000 2.0
               3.000000 0.000000
                                     16
23.000000 2.0
               3.000000 1.000000
                                     15
21.000000 2.0
               3.000000
                         0.000000
                                     13
                          1.000000
                                     12
21.282238 3.0
               3.000000
                         0.849236
21.274628 3.0
               3.489918 0.128394
21.243142 3.0
               1.726260 0.000000
21.238416
         3.0
               3.000000 0.890527
                                      1
61.000000 3.0
               3.000000 1.000000
Name: count, Length: 1784, dtype: int64
```

#### 2. 데이터 전처리

1. 실수형 데이터 스케일링 (StandardScaler)

```
#실수형 데이터인 피쳐 세계 정규화
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
float_list=['Height','Weight','CH2O']
#타겠라 인코딩
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
pig['NObeyesdad'] = label_encoder.fit_transform(pig['NObeyesdad'])
#실수형 데이터만 선택한 데이터프레일
pig_float=pig[float_list]
```

#### 데이터 전처리

2. smote로 인해 변환된 함수 round() 적용

```
#smote로 인해 변환된 함수 반을림 적용.
pig.Age = pig.Age.round(); pig.FCVC = pig.FCVC.round(); pig.NCP = pig.NCP.round(); pig.TUE = pig.TUE.round()
```

3. 피처와 타겟 구분

```
#폐처와 타켓 분리
X_features = pig.drop(['NObeyesdad'], axis=1, inplace=False)
y_target = pig['NObeyesdad']
```

4. 비정형 데이터에 원핫인코딩 적용

X\_features\_ohe = pd.get\_dummies(X\_features, columns=['Gender','family\_history\_with\_overweight','FAVC','CAEC','SMOKE','SCC','CALC','MTRANS'], dtype=int)
X features ohe

# 데이터 전처리

	Age	FCVC	NCP	FAF	TUE	Height	Weight	CH20	Gender_Female	Gender_Male	 SCC_yes	CALC_Always	CALC_Frequently
0	21.0	2.0	3.0	0.000000	1.0	-0.875589	-0.862558	-0.013073	1	0	 0	0	0
1	21.0	3.0	3.0	3.000000	0.0	-1.947599	-1.168077	1.618759	1	0	 1	0	0
2	23.0	2.0	3.0	2.000000	1.0	1.054029	-0.366090	-0.013073	0	1	 0	0	1
3	27.0	3.0	3.0	2.000000	0.0	1.054029	0.015808	-0.013073	0	1	 0	0	1
4	22.0	2.0	1.0	0.000000	0.0	0.839627	0.122740	-0.013073	0	1	 0	0	0

## 상관도 분석

target	1.000000
Weight	0.387643
CAEC_Sometimes	0.316962
family history with overweight yes	0.313667
Age	0.235660
CALC_Sometimes	0.114104
CH20	0.108868
CAEC_no	0.066715
SCC_no	0.050679
CALC_Frequently	0.047318
FAVC_yes	0.044582
MTRANS_Automobile	0.041170
Height	0.038986
Gender_Male	0.024908
SMOKE_no	0.023256
FCVC	0.012068
MTRANS_Public_Transportation	-0.003748
MTRANS_Bike	-0.017351
CALC_Always	-0.022484
SMOKE_yes	-0.023256
Gender_Female	-0.024908
MTRANS_Motorbike	-0.034293
FAVC_no	-0.044582
SCC_yes	-0.050679
TUE	-0.059050
MTRANS_Walking	-0.073823
NCP	-0.085367
CAEC_A1ways	-0.099028
FAF	-0.129564
CALC_no	-0.134716
family_history_with_overweight_no	-0.313667
CAEC_Frequently	-0.351827
Name: target, dtype: float64	

Age - 1000504815.290260.04620492121060630911274062900112.0200063001000500.063.00	1.00
FCVC0.005 1 0 TORIZ 907 70 94/080 47726 - 2/602 92 99 TOR TO 00 94 15/69 200 047 607 607 507 500 9960 702 200 704 100 201 2	
NCP-0.0061 10 TB0 TB2-0 0.08 TUT 07 2016 0600 060 00 04 400 04 40 00 07 06 50 17 0.09 05 00 20 08 08 50 08 50	
FAF -0, 15:02, 1, 110, 06, 29:0 5 ft-7, 19, 09:0 57:0 7-10, 0 f02:30, 300 90;400, 00 f07:0 7:0 f02:0 67:0 f02:0 67:0 1, 13	
THE 40.2907/0130611 08800000000/2028/02/06808 20 0080400 0080600 201 04 26 302 062 105 4020 620 106:05 9	- 0.75
Height-0.02.68428-2905 10.46.2 1 67.60 2528.100 804602-0409-6585551-80100008-0510.0207.0209.039	
Weight -0.20 180-0.05 08916 1 0.20 1851 0 30 -0 27 27 00 0538 40 -0 102 62 6 20-2 60 20 02 86 2591 40 20 20 59 0 39	
CH2O-D. CM68D CO TITO COLD D. 2 1 D. 10, 10, 150 SD CO COLD SC COLD SC COLD DO COLD COLD COLD COLD COLD COLD	The second
Gender_Female-0.049360720190077-0.16.1 1 1 1 0.10006965909320723694576-0.00229588924049570453692925	- 0.50
Gender_Male 9.048967001900(X3) 16.11-1 11-0.10-0 (0568-9-19-02073963734646.10-01-02-39-97387347405-04-45-0408-9-25	
amily_history_with_overweight_no -0.2/192.8/665702525-0.19.10.11.5.0.20.2070320.3/4.090070-07/1919.0102.8/253.8/990.96.05.947/31	
mily_history_with_overweight_yes_0_2000206305025250.0.150.10.151.10.20.20.0332730-0900.07171-0.1000.92.0280.00-90.55.06596.201	1
FAVC_no-0.00600006401.0601082.70.0000006020.2 1 31 05016.1990005050119000078071041200508.08010900045	- 0.25
FAVC.,ves 0.06303096.40 D6318.2 749905645212 1 1,0851819.0336709119019907967.0.020898202562008045	
CAEC ANNOYO-0, 93 TO 82 92 TO TO HAR STRIPE BY 90 90 TO 1885 DE 11, 0 888 60 DESTARDA CIBIER O CRIB TIA GUICURO 43 TECA CO 20 TO 1885 DE 1886 DE STARDA CIBIERO A CRIB TIA GUICURO 43 TECA CO 20 TO 1885 DE 1886 DE STARDA CIBIERO A CRIB TIA GUICURO 43 TECA CO 20 TECA	
CAEC_Frequently -0.12(0500,03899890,30,30,10410,10220,30718-0805,10,0,03800,40,10,1006,04618-0.09,0020070694235	
CAEC_Sometimes 0.0.00592404040718.40083070723438.1019.20.811 0.906205218.0609.907.0.0070.00207.562032	- 0.00
CAEC_no-0.0629890.024-0109611 150586319.1908.4892.5673 100:0401464561845635164864901.0560.267	
SMOKE, NO-D. 40910 1987/01/08885 812:03.1945/1060/200 705/080-04174/05/20 11 -1 00:00/48/32/20 (11:06:30 11)HE 012:00/201 0/23	
SMOKE_yes 0.0900700700100986592603.80659300.00.865050090505200-11 1 .0040480820.0030560120089502907023	
SDC_no 0.120750.478 031 0.2.000 10.0.1219.19 1908110.1606993041 1 004860420204392 BARO.04951	0.25
SCC_yes -0.0207331-0740-031-002008 10.0.10.19.191-0.0000110.000690484-1 10.004081042020482 04:00.049051	
CALC_Always-0, 90, 93-8 4530 0 4530 0 4530 0 4530 0 22 0 22 0 10 10 10 10 10 10 10 10 10 10 10 10 1	
CALC_Frequently 0.0830 20 0555 84600 90 903 50 50202 9570 508 96 89 90 50 10-0 00 60 50 11 26 02 990 50 30 80 9 47	
CALC_Sometimes-0.0010000408210 2606288038925251047040-0610 1008300 90.3942942-512 10.00.07.90.10189806.11 1	0.50
CALC.,no -0.42936970206213:2592823242833314.02013:0.4705963083020202044;1,931,0420.6341633913	
MTRANS_Automobile -0.30.070600030500800.40600-040090905060002990-0-000020-05030480-0800794 10-000305 08-0041	
MTRANS_BikeO-042842942947020490294605057050058620804302003809080080242040140.00.0450 10 00489909517	17902
MTRANS_Motorbike 0.020070281.662010.000.000.000.000.000.000.000.000.000	0.75
MTRANS_Public_Transportation - 0.007/060494.09593816.006959299994783625022729300103828999001.0491 10/28037	
MTRANS_Walking-0.054010540105930000E0390396706748.0006796701040000000000000000000000000000000000	
target Q 24002085030593839-01 026353131 0450519335320679230296795 0224710 0394790 0394797 1	
	1.00
Age REVOC NOT PER PET TO THE PET	
W N N N N N N N N N N N N N N N N N N N	

#### 소프트 보팅과 하드 보팅을 통한 학습

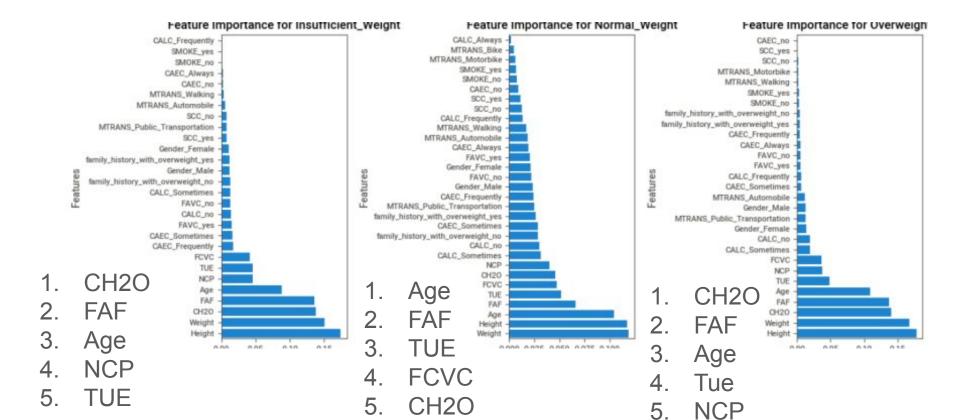
```
from sklearn.ensemble import VotingClassifier
from sklearn.linear model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model selection import train test split
from sklearn.metrics import accuracy score
# 데이터 준비
X = X features ohe # ゴオ
y = pig['NObeyesdad'] # 타켓
# 훈련 데이터와 테스트 데이터 분리
X train, X test, y train, y test = train test split(X, y, test size=0.2, random state=42)
# 개별 모델 정의
dt clf = DecisionTreeClassifier(random state=156)
log reg = LogisticRegression(max iter=1000, random state=156)
rf clf = RandomForestClassifier(random state=156, max depth=20, max features='sqrt', min samples leaf=1, min samples split=5, n estimators=200
gb model = GradientBoostingClassifier(learning rate= 0.05, max depth= 3, min samples split= 2, n estimators= 200, subsample= 0.8, random state
# 하드 보팅 분류기
hard voting clf = VotingClassifier(estimators=[
   ('log_reg', log_reg),
   ('decision tree', dt clf),
   ('rf clf', rf clf),
   ('gb model', gb model)
1, voting='hard') # 하드 보팅
# 소프트 보팅 부류기
soft voting clf = VotingClassifier(estimators=[
   ('log reg', log reg),
   ('decision tree', dt clf),
   ('rf clf', rf clf),
   ('gb model', gb model)
], voting='soft') # 소프트 보팅
```

## 모델 훈련, 예측, 평가

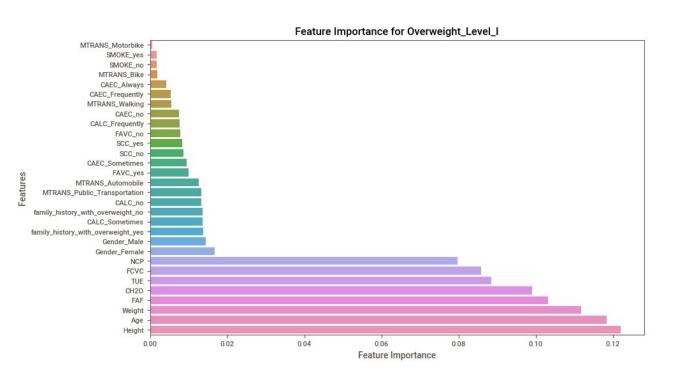
```
# 모델 훈련
hard voting clf.fit(X train, y train)
soft voting clf.fit(X train, y train)
# 예측
y_pred_hard = hard_voting_clf.predict(X_test)
y pred soft = soft voting clf.predict(X test)
# 정확도 평가
accuracy hard = accuracy score(y test, y pred hard)
accuracy soft = accuracy score(y test, y pred soft)
print(f"Hard Voting Accuracy: {accuracy hard:.4f}")
print(f"Soft Voting Accuracy: {accuracy soft:.4f}")
```

Hard Voting Accuracy: 0.9598 Soft Voting Accuracy: 0.9669

## 저체중 그룹, 정상체중 그룹, 과체중 1 그룹



## 과체중 그룹-Overweight\_Level\_I

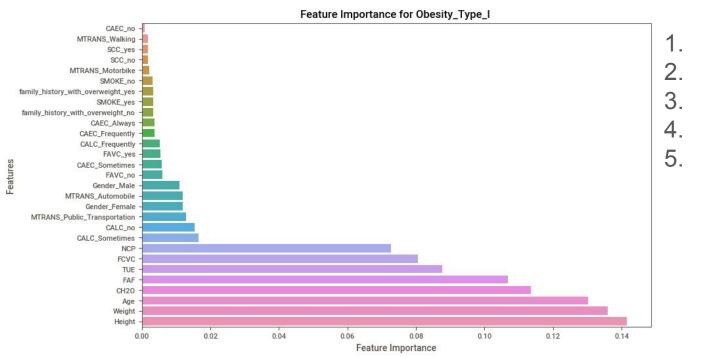


- 1. FAF
- 2. CH2O
- 3. TUE
- 4. FCVC
- 5. NCP

중요 인자

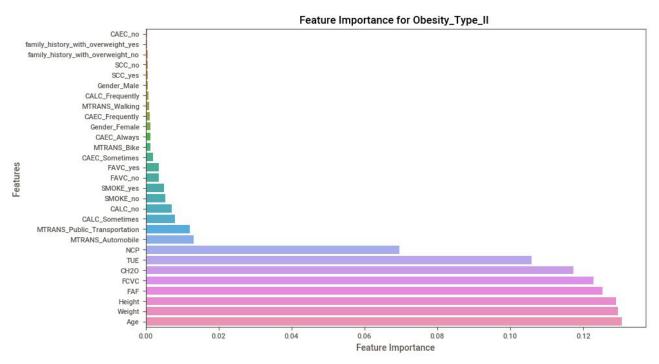
- FAF: 운동량
- CH2O: 물섭취량
- TUE: 전자기기 사용
- FCVC: 채소섭취량
- NCP: 하루에 먹는 끼니 수

# 비만 그룹-Obesity\_Type\_I



- 1. CH2O
- 2. FAF
- 3. TUE
- 4. FCVC
- 5. NCP

# 비만 그룹-Obesity\_Type\_II



- 1. FAF
- 2. FCVC
- 3. CH2O
- 4. TUE
- 5. NCP

# 비만 그룹-Obesity\_Type\_III

