비만과 탈모 예측

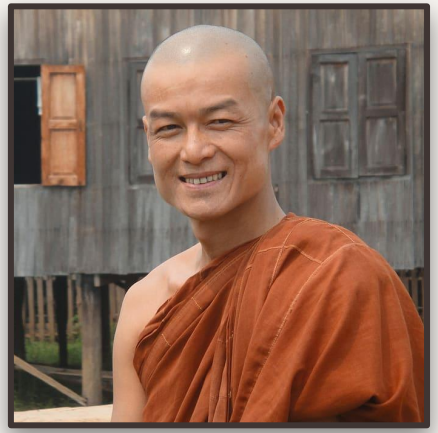
김동현
고우석
전진우
이윤서

두 가지 주제

비만

이윤서

전진우



탈모

고우석

김동현

01

비만

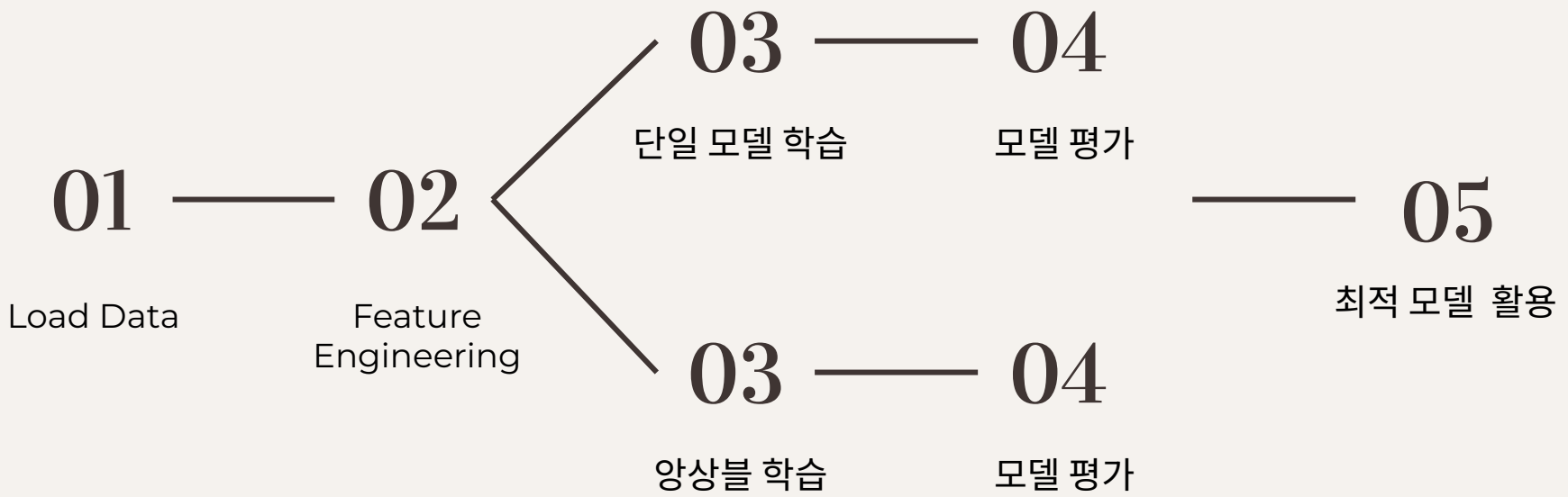
1-1 분류
1-2 회귀

01

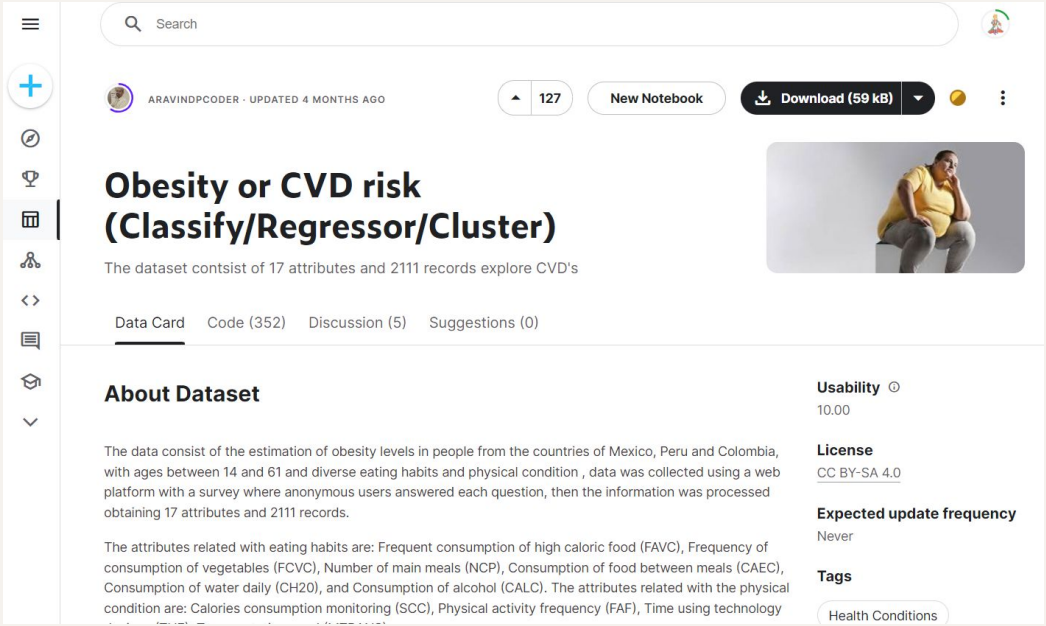
비만 : Classification

내 생활 습관으로 보는 비만 위험도는?

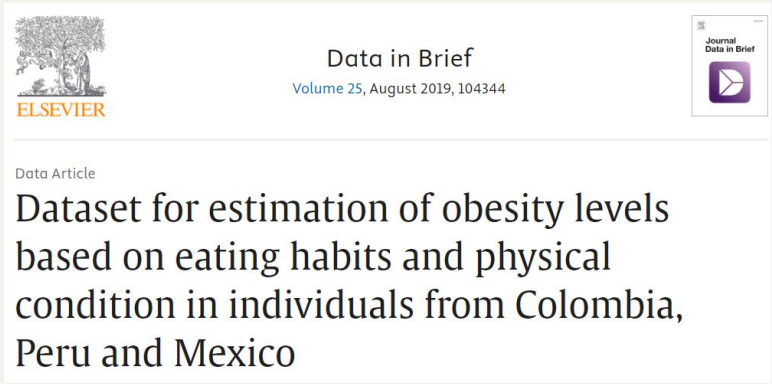
0. Index


01
Load Data
02
Feature Engineering
03
단일 모델 학습
04
모델 평가
03
앙상블 학습
04
모델 평가
05
최적 모델 활용

Data From:



Kaggle 비만 데이터



데이터 생성 정보

데이터 전처리

식습관 속성

- 고열량 식품 빈도(FAVC)
- 채소 빈도(FCVC)
- 하루 끼니 횟수(NCP)
- 간식(CAEC)
- 흡연 유무
- 물 섭취(CH20)
- 음주빈도(CALC)

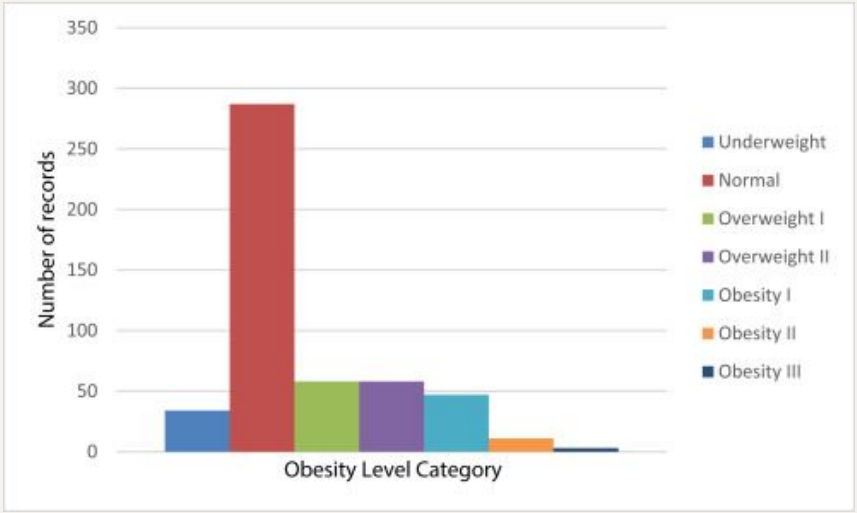
신체 상태 속성

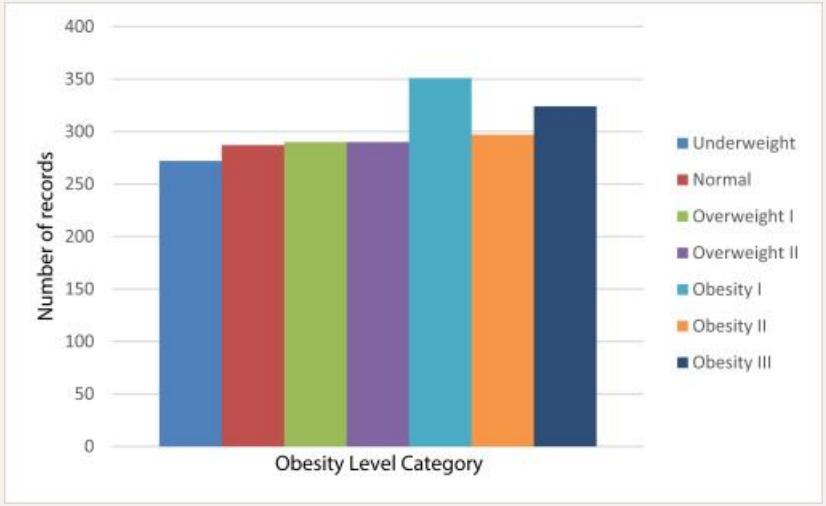
- 칼로리 소비(SCC)
- 활동량(FAF)
- 휴대폰 사용량(TUE)
- 이동수단(MTRANS)
- 가족력

기본 속성

- Gender : 성별
- Age : 나이
- Height : 키 m
- Weight : 몸무게 kg

데이터 전처리1. 클래스 균형을 위한 오버샘플링





Weka, SMOTE : 소수 클래스의 샘플을 합성하여 **클래스의 균형**을 맞춰줌

데이터 전처리2. 샘플링 노이즈 제거

업샘플링 전	업샘플링 후
좋음 3 보통 2 싫음 1	3 2.2548 1 2.4554
36.5 12.4 89.4	45.1648345 14.415612 36.5

적절한 round 처리 필요

데이터 전처리3. 범주형 데이터 Labeling

```python
# 순서형 데이터 정리
def self_OrdinalEncoder(DF, col_name, value_list):
    '''
    DF : 해당 데이터 프레임
    col_name : 대상 컬럼 명
    name_list : OrdinalEncoder를 위해 정렬한 value들의 list

    return : 인코딩된 컬럼 SR
    '''
    Encoding_dic[col_name] = tuple(zip(value_list, range(len(value_list))))
    SR = DF[col_name].replace(value_list, range(len(value_list)))
    return SR
```

순서 상관 있으면 -> Ordinal Encoding으로 순서 지정 후 넣기

순서 상관 없으면 -> 그냥 Ordinal Encoding / One Hot Encoding

데이터 전처리4. Regression을 위한 target 생성

$$Mass\ body\ index = \frac{Weight}{height*height}$$

After all calculation was made to obtain t
results were compared with the data prov

- Underweight Less than 18.5
- Normal 18.5 to 24.9
- Overweight 25.0 to 29.9
- Obesity I 30.0 to 34.9
- Obesity II 35.0 to 39.9
- Obesity III Higher than 40

→ 회귀 target

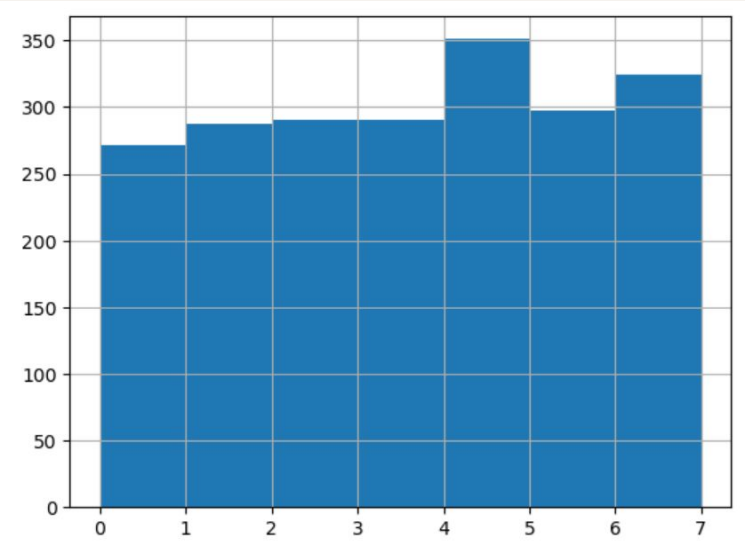
```
num_DF["BMI"] = num_DF.Weight / (num_DF.Height) ** 2
num_DF
```

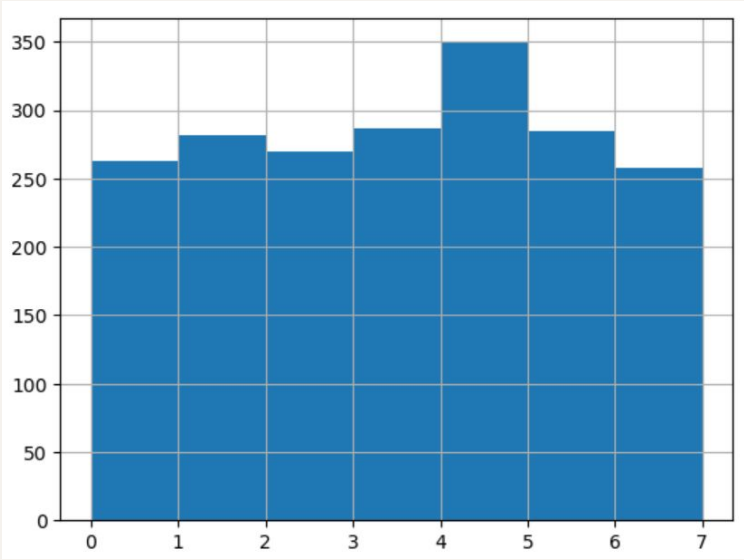
→ 분류 target

(기존 데이터에 있음)

NObeyesdad
Normal_Weight
Normal_Weight
Normal_Weight
Overweight_Level_I
Overweight_Level_II

데이터 전처리5. 중복 제거 및 클래스 균형 확인





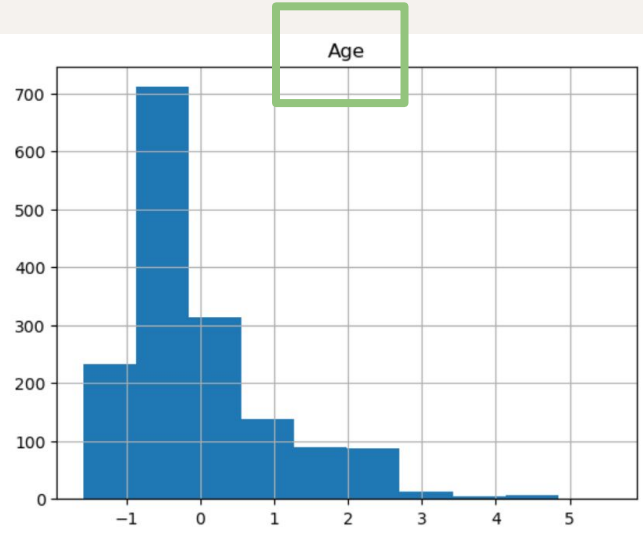
중복 제거 후에도 클래 균형이 크게 무너지지 않음

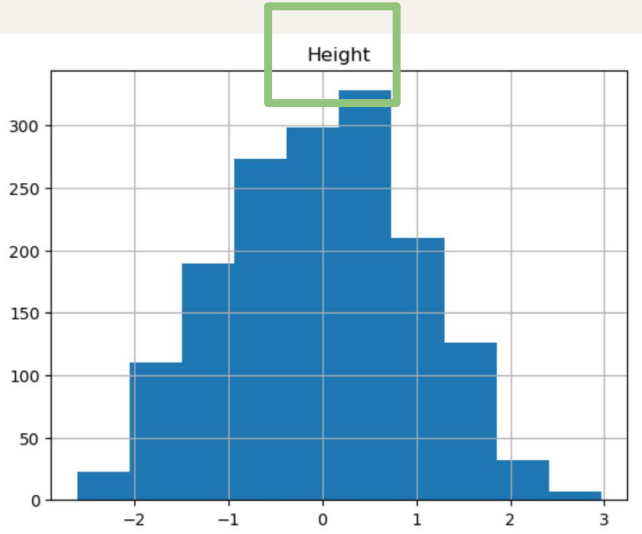
데이터 전처리5. 상관관계 확인 및 Scaling

$$\frac{Weight}{height*height}$$

상관관계 0.92 -> 타겟에 너무 직접적인 데이터 drop

Age	Height	Weight
21	1.62	64.0
21	1.52	56.0
23	1.80	77.0





단일 모델 선택

상관관계가 0.1 이하인 feature 포함

	train	test
KNeighborsClassifier()	0.835840	0.771930
LogisticRegression(max_iter=1000, random_state=42)	0.583960	0.571429
SVC(random_state=42)	0.726190	0.724311
DecisionTreeClassifier(max_depth=6, random_state=42)	0.656642	0.606516

포함X

train	test
0.810930	0.706030
0.564070	0.537688
0.694095	0.673367
0.653266	0.590452

-> 상관관계가 낮다고 무조건 불필요한 요소는 아닐 수도 있음

단일 모델 선택 - SVC 최적 파라미터

```
from sklearn.model_selection import GridSearchCV

n_CV = 5

params = { "kernel" : [ "poly", "linear", "rbf", "sigmoid"],
           "degree" : [2, 3, 4],
            "gamma" : ["scale", "auto"],
            "C" : [0.001, 0.01, 0.1, 1]
}

KNN_GS = GridSearchCV(SVC_model, param_grid=params, cv = n_CV)

result = KNN_GS.fit(Stand_X_train, y_train)
```

-> 과대적합

전 train 72.6 test 72.4

후 train 83.8 test 78.7

단일 모델 선택 - Logistic과 확률적경사하강법

```
Logistic_model = LogisticRegression(max_iter=1000, random_state=42)
```

```
from sklearn.linear_model import SGDClassifier # 확률적 경사하강법

SGD_model = SGDClassifier(loss="log_loss", random_state=42, max_iter = 10000)
# 한 번에 한 개 데이터만 랜덤 샘플링 통해 추출 후 Gradient 계산

SGD_model.fit(Stand_X_train, y_train)
```

model	train	test
LogisticRegression	0.584	0.571
SGDClassifier(loss=”log_loss”)	0.504	0.486

-> 데이터 전체를 학습시킬 여건이 된다면 확률적 경사하강법 보다 로지스틱 회귀가 좋음

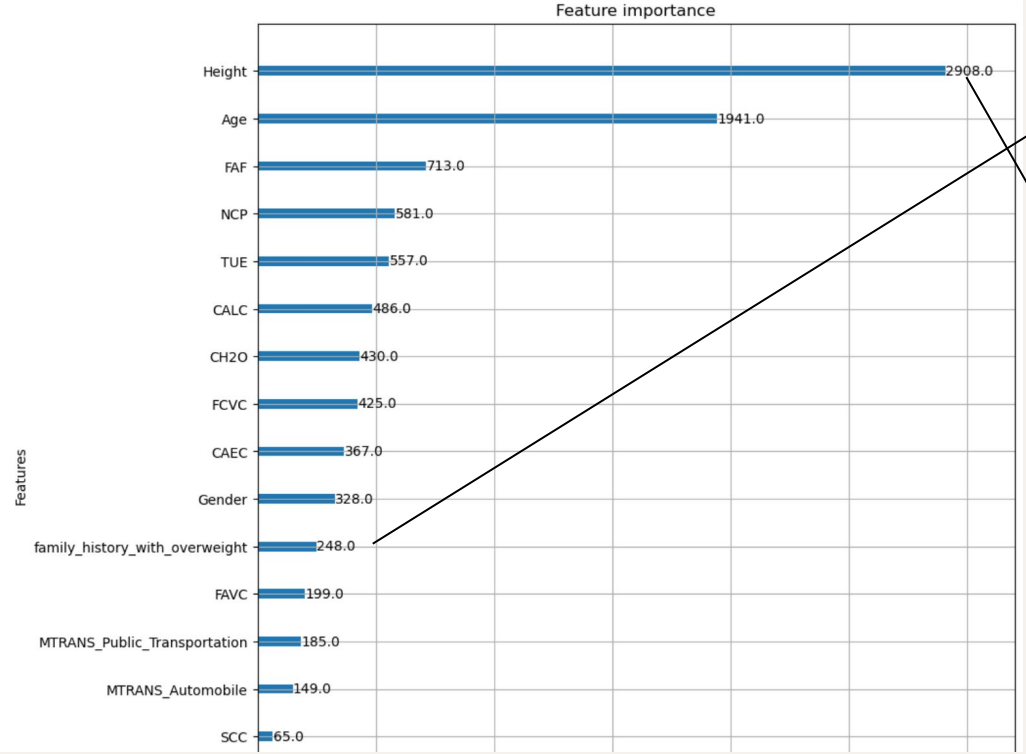
앙상블

gridCV best_model

모델	Train Score	Test Score	결과
RandomForestClassifier	0.719	0.709	최적
BaggingClassifier	0.793	0.764	과소
AdaBoostClassifier	0.937	0.810	과대
GradientBoostingClassifier	0.911	0.802	과대
XGBClassifier	0.999	0.842	과대

-> 단일 모델들 보다 앙상블 모델이 성능이 더 좋음을 알 수 있음

XGBClassifier 결과



corr

NObeyesdad	1.000000
family_history_with_overweight	0.500697
CAEC	0.341265
Age	0.284422
FAVC	0.242766
SCC	0.194944
FAF	0.178842
FCVC	0.157467
Height	0.152476
MTRANS_Walking	0.136622
CALC	0.136001
CH2O	0.108989
TUE	0.065581
MTRANS_Public_Transportation	0.064605
MTRANS_Motorbike	0.036100
MTRANS_Bike	0.035888
NCP	0.012468
Gender	0.011981
SMOKE	0.009518
MTRANS_Automobile	0.003184

새로운 데이터를 통한 예측

```
my_data = pd.DataFrame( data: [[0, -0.37025452, -1.18250493, 1, 1, 1, 2, 2, 0, 2, 0, 2, 0, 3, 0, 0, 0, 0, 1]],
                        columns=['Gender', 'Age', 'Height', 'family_history_with_overweight', 'FAVC',
   'FCVC', 'NCP', 'CAEC', 'SMOKE', 'CH2O', 'SCC', 'FAF', 'TUE', 'CALC',
   'MTRANS_Automobile', 'MTRANS_Bike', 'MTRANS_Motorbike',
   'MTRANS_Public_Transportation', 'MTRANS_Walking'])


# 입력된 정보에 해당하는 비만도 알려주기
obesity = model.predict(my_data)

proba = model.predict_proba(my_data)
print(f"{obesity_list[obesity[0]]}입니다.")
```

use_model

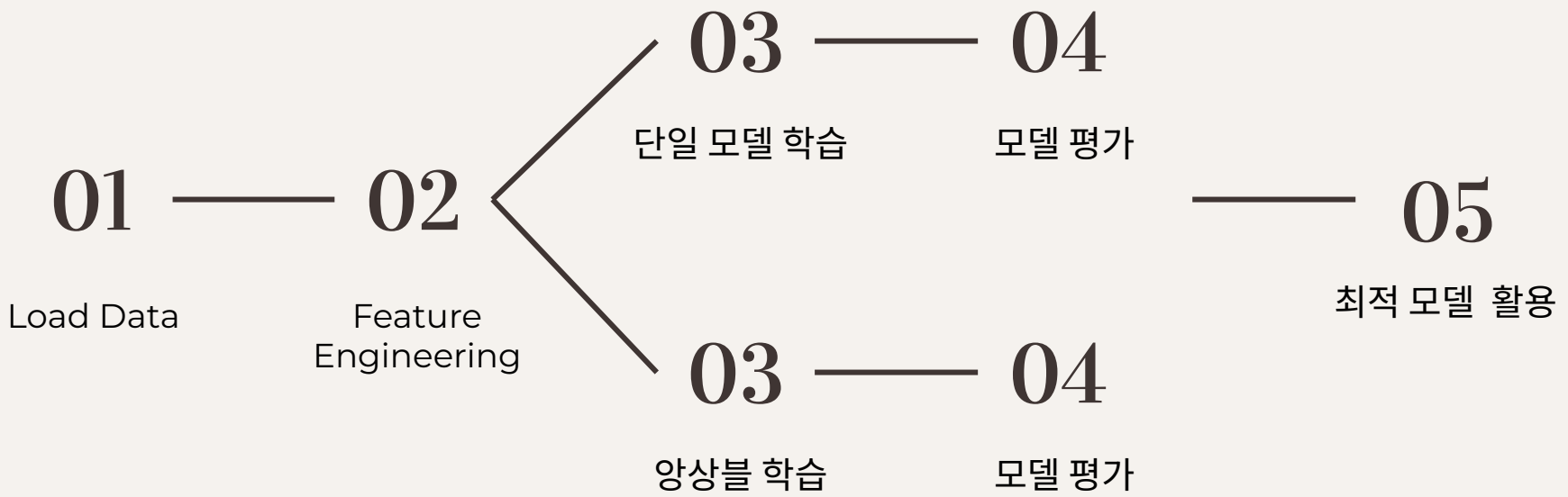
```
C:\Users\kdp\anaconda3\envs\EXAM_ML\python.exe C:\Users\kdp\PycharmProjects\EXAM_ML\project\use_model.py
Normal_Weight입니다.
```

02

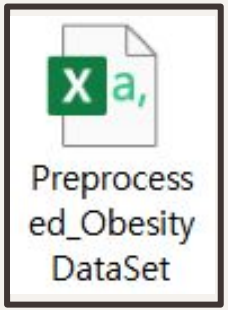
비만 : Regression

Insert a subtitle here if you need it

0. Index


01
Load Data
02
Feature Engineering
03
단일 모델 학습
04
모델 평가
03
앙상블 학습
04
모델 평가
05
최적 모델 활용

1. Prepare Data



갓.윤서님이
전처리한 .csv

Data columns (total 21 columns):

#	Column	Non-Null Count	Dtype
0	Gender	2111 non-null	int64
1	Age	2111 non-null	int64
2	Height	2111 non-null	float64
3	Weight	2111 non-null	float64
4	family_history_with_overweight	2111 non-null	int64
5	FAVC	2111 non-null	int64
6	FCVC	2111 non-null	int64
7	NCP	2111 non-null	int64
8	CAEC	2111 non-null	int64
9	SMOKE	2111 non-null	int64
10	CH2O	2111 non-null	int64
11	SCC	2111 non-null	int64
12	FAF	2111 non-null	int64
13	TUE	2111 non-null	int64
14	CALC	2111 non-null	int64
15	MTRANS_Automobile	2111 non-null	int64
16	MTRANS_Bike	2111 non-null	int64
17	MTRANS_Motorbike	2111 non-null	int64
18	MTRANS_Public_Transportation	2111 non-null	int64
19	MTRANS_Walking	2111 non-null	int64
20	NObeyesdad	2111 non-null	int64

dtypes: float64(2), int64(19)
memory usage: 346.5 KB

1. Prepare Data

01

결측치 = 없음

02

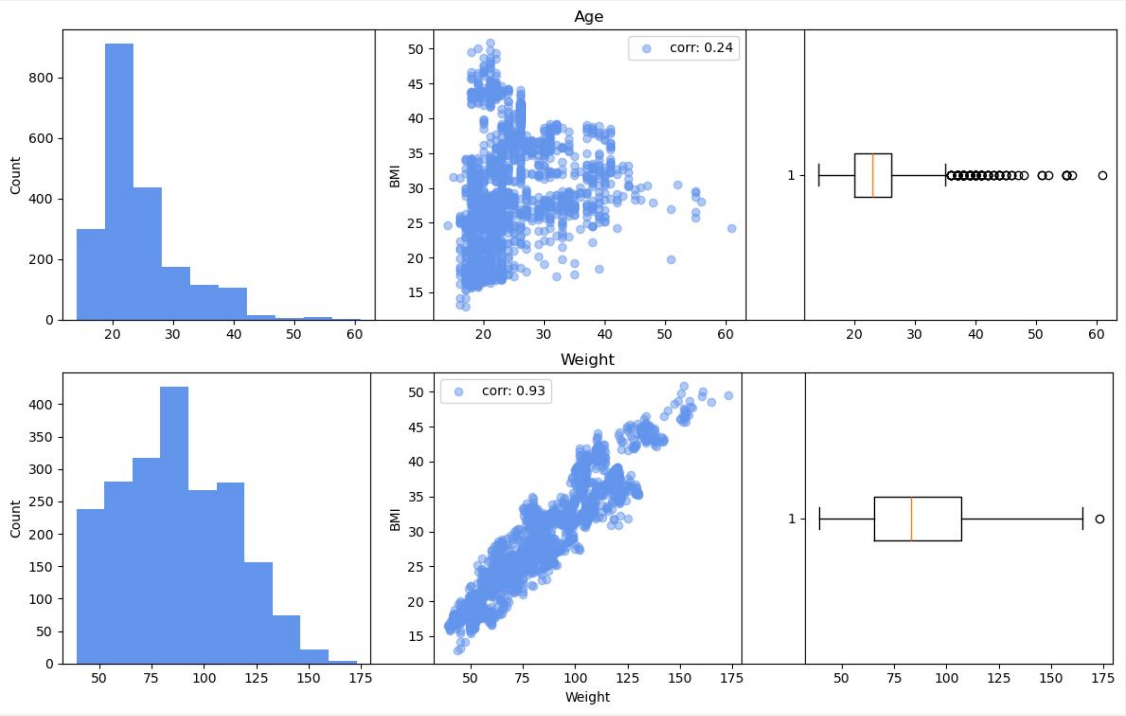
중복값 = 47

03

이상치

1. Prepare Data

이상치가 확인된 피처: 45세 이상 제외



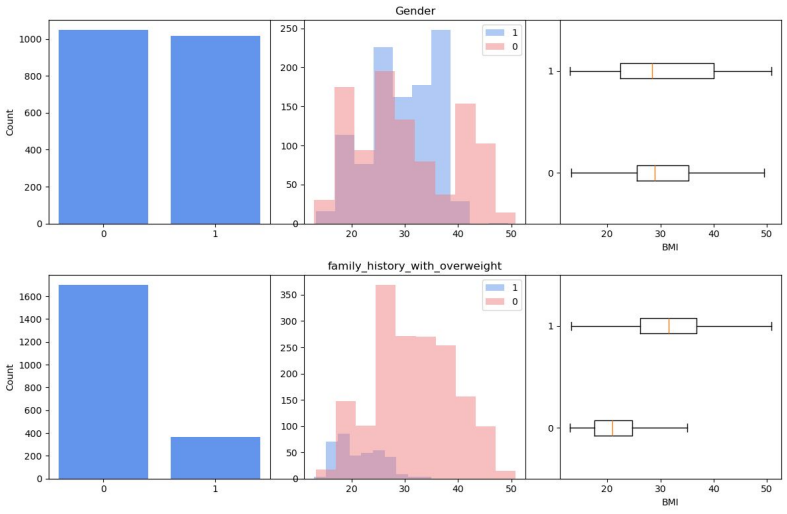
03

이상치

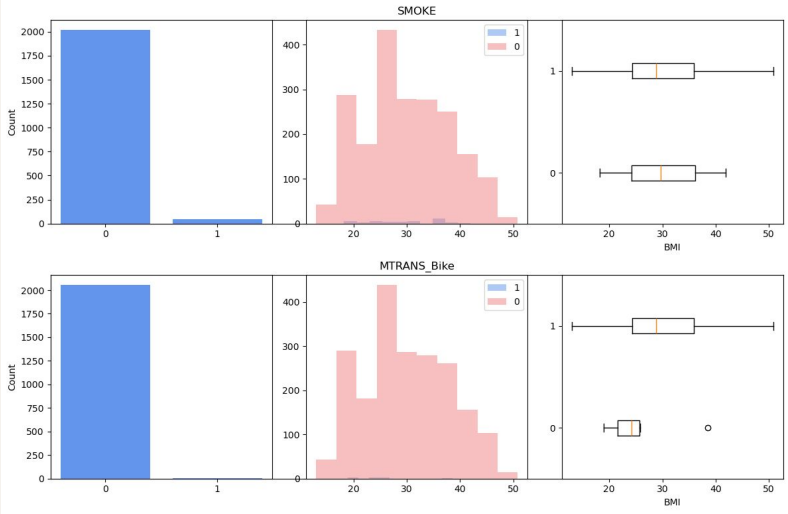
시각화 분석

2. Feature Selection

고른 분포의 피처

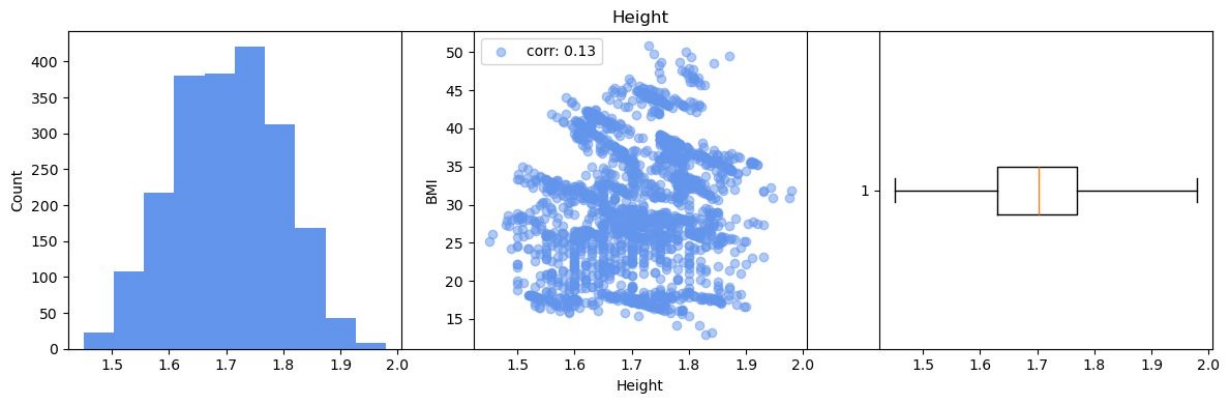


극심한 편향을 보이는 피처



2. Feature Selection

너무 낮은 상관 관계



2. Feature & Target Split

test_size=0.2, random_state=11

	성별	나이	키	가족력	고열량 섭취 빈도	채소 섭취 빈도	하루 섭취 끼니 수	간식 섭취 빈도	물 섭취량	활동량	스마트폰 사용량	음주 빈도	이동 수단 : 자가용	이동수단 : 대중교통	BMI
0	0	21	1.620	0	0	2	3	2	2	0	1	3	0	1	24.386526
1	0	21	1.520	0	0	3	3	2	3	3	0	2	0	1	24.238227
2	1	23	1.800	0	0	2	3	2	2	2	1	1	0	1	23.765432
3	1	27	1.800	1	0	3	3	2	2	2	0	1	0	0	26.851852
4	1	22	1.780	1	0	2	1	2	2	0	0	2	0	1	28.342381
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2059	0	21	1.711	0	1	3	3	2	2	2	1	2	0	1	44.884392
2060	0	22	1.749	0	1	3	3	2	2	1	1	2	0	1	43.707080
2061	0	23	1.752	0	1	3	3	2	2	1	1	2	0	1	43.557526
2062	0	24	1.739	0	1	3	3	2	3	1	1	2	0	1	44.078924
2063	0	24	1.739	0	1	3	3	2	3	1	1	2	0	1	44.145059

3-1. Model Selection

먼저 생각나는 모델

Model	KNN	Linear Regression	SVR
Train Score	0.85029	0.45401	0.72540
Test Score	0.72540	0.46332	0.58650

3-1. Model Selection : all_estimators( )

모듈 활용 :

```
# all estimators
from sklearn.utils import all_estimators

allAlgorithms = all_estimators(type_filter='regressor')
# print(allAlgorithms)
all_regressor = []
all_regressor_name = []
```

```
all_regressor_name =[
        'AdaBoostRegressor',
    'DecisionTreeRegressor',
    'ElasticNetCV',
    'ExtraTreeRegressor',
    'ExtraTreesRegressor',
    'GradientBoostingRegressor',
    'KNeighborsRegressor',
    'KernelRidge',
    'Lasso',
    'LassoCV',
    'LinearRegression',
    'LinearSVR',
    'RandomForestRegressor',
    'Ridge',
    'RidgeCV',
    'SGDRegressor',
    'SVR',
    'VotingRegressor']
```

3-1. Model Selection : all_estimators( )

ExtraTreesRegressor	RandomForest Regressor	DecisionTreeRegressor	ExtraTreeRegressor
0.875205	0.867078	0.773442	0.746733
KNeighborsRegressor	GradientBoosting Regressor	SVR	AdaBoostRegressor
0.725404	0.710392	0.58651	0.538573
ElasticNetCV	KernelRidge	Ridge	LassoCV
0.467844	0.466905	0.465284	0.46413
RidgeCV	LinearRegression	SGDRegressor	LinearSVR
0.463563	0.463321	0.453672	0.452019

3-2. Model Evaluation : Extra Trees

과.적.합

	ExtraTrees( )	range(1, 150)	GridSearchCV
Parameters	-	random_state=i -> 11	criterion='absolute_error', max_depth=20, n_estimators=150
Train Score	0.9999	0.9999	0.9988
Test Score	0.8737	0.8763	0.8766

3-3. Model Evaluation : RandomForest

과.적.합.2

	RandomForest( )	range(1, 150)	GridSearchCV
Parameters	-	random_state=i -> 4	criterion="poisson", max_depth=20, max_features="sqrt", n_estimators=150
Train Score	0.9797	0.9802	0.9821
Test Score	0.8658	0.8753	0.8757

3-3. Model Evaluation : RandomForest

최종 선택 : RandomForest( )

	RandomForest( )	range(1, 150)	GridSearchCV
Train Score	0.9797	0.9802	0.9827
Test Score	0.8658	0.8753	0.8764
	ExtraTrees( )	range(1, 150)	GridSearchCV
Train Score	0.9999	0.9999	0.9988
Test Score	0.8737	0.8763	0.8766

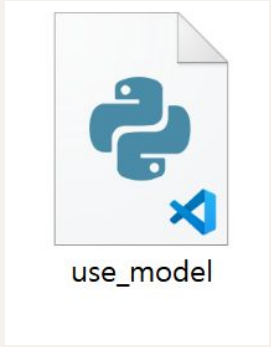
3-4. Model Evaluation : Boosting

	XGBRegressor( )	RandomForest( )
Parameters	Eta = 0.15	
Train Score	0.9705	0.9821
Test Score	0.8505	0.8757

4. Save Model

```python
# rfbs 저장
import pickle
pickle.dump(rfbs, open('rfbs.pkl', 'wb'))
```

```python
# rfbs 저장
import pickle
pickle.dump(rfbs, open('rfbs.pkl', 'wb'))
```



5. Prediction

성별	나이	키	가족력	고열량 섭취 수	채소 섭취	하루 섭취 끼니 수
1	25	1.71	0	0	2	3
간식 섭취 빈도	물 섭취량	활동량	스마트폰 사용량	음주 빈도	이동수단: 자가용	이동수단: 대중교통
3	3	3	1	2	0	1

03

탈모팀-고우석

꼭, 그렇게 빠져야만
하는가?

“머리카락이 빠지는 것이 아니다.
내가 전진하고 있는 것이다.”

–손정의(소프트 뱅크 창업주)

1. 데이터 소개

11 rows | 100000 rows × 11 columns | Static Output

	total_protein	total_keratine	hair_texture	vitamin	manganese	iron	calcium	body_water_content	stress_level	liver_data	hair_fall
0	312	100	14	249	87	55	333	44	41	368	4
1	52	207	3	425	387	1	182	26	65	41	1
2	170	197	11	140	199	91	414	30	54	90	4
3	256	334	19	358	120	3	35	48	45	65	2
4	309	185	58	207	329	301	345	23	90	346	4
...	...	...	...	...	...	...	...	...	...	...	...
99995	440	20	63	209	473	260	12	86	7	281	5
99996	311	233	37	352	194	159	391	57	76	30	5
99997	27	210	10	185	376	278	96	40	49	110	3
99998	450	18	35	250	287	157	81	31	2	156	1
99999	464	487	15	440	110	164	356	33	93	12	0

Hair-loss 데이터 셋(두발에 존재하는 여러 영양소들과 그에 따른 탈모율 정보가 들어 있음)

출처:kaggle

1. 데이터 소개

1-100 7757 rows × 14 columns

	age	gender	job_role	province	salary	is_married	is_hereditary	weight	height	shampoo	is_smoker
0	27.0	female	Government Employee	Bengkulu	7.957453e+06	1.0	0.0	54.315053	170.428542	Pantone	1
1	53.0	female	Government Employee	Bandung	7.633003e+06	1.0	0.0	72.873404	165.530097	Pantone	0
2	37.0	female	Employee	Bandung	6.637625e+06	1.0	0.0	46.321533	154.599388	Moonsilk	0
3	36.0	female	Jobless	Palu	3.624871e+06	1.0	0.0	51.539781	167.340481	Deadbuoy	1
4	38.0	male	Unspecified	Palangkaraya	6.031808e+06	1.0	0.0	60.726909	165.514773	Merpati	1
5	55.0	female	Government Employee	Palangkaraya	9.213032e+06	1.0	1.0	54.287045	179.235145	Pantone	0
6	40.0	female	Unspecified	Serang	1.068256e+07	1.0	0.0	54.824881	177.431122	Shoulder & Head	0
7	47.0	male	Employee	Banda Aceh	4.508321e+06	1.0	0.0	74.795152	170.540938	Shoulder & Head	1
8	41.0	male	Unspecified	Palu	9.846426e+06	1.0	0.0	55.049547	157.192078	Pantone	0
9	46.0	female	Unspecified	Palembang	9.257426e+06	1.0	0.0	52.764656	174.397170	Shoulder & Head	0
10	34.0	male	Government Employee	Kupang	4.765812e+06	1.0	1.0	43.657207	151.078647	Moonsilk	0
11	40.0	male	Employee	Sofifi	1.862776e+07	1.0	0.0	81.140441	167.492096	Shoulder & Head	0
12	41.0	male	Employee	Ambon	1.939622e+07	1.0	0.0	47.959075	152.098351	Deadbuoy	1
13	26.0	male	Employee	Tanjungselor	1.036350e+07	1.0	1.0	54.047139	175.790306	Pantone	1
14	45.0	male	Government Employee	Tanjung Pinang	1.211718e+07	1.0	1.0	54.548239	167.855936	Moonsilk	1
15	21.0	male	Employee	Banjarmasin	6.867878e+06	1.0	0.0	45.084456	151.949667	Merpati	1
16	33.0	male	Employee	Denpasar	4.466446e+06	1.0	0.0	49.239343	165.955076	Deadbuoy	1
17	54.0	male	Employee	Mamuju	1.147456e+07	1.0	0.0	69.381971	175.132923	Pantone	1
18	35.0	female	Employee	Kupang	1.209364e+07	1.0	0.0	54.247992	153.990308	Merpati	0
19	38.0	male	Jobless	Banda Aceh	1.087815e+07	1.0	0.0	53.103794	187.817838	Deadbuoy	0
20	38.0	female	Employee	Kupang	6.786868e+06	1.0	1.0	65.393980	169.192650	Shoulder & Head	1
21	49.0	male	Government Employee	Makassar	1.020936e+07	1.0	0.0	55.953868	175.460518	Merpati	0
22	49.0	male	Employee	Pangkalpinang	7.061549e+06	1.0	0.0	62.676346	176.575678	Deadbuoy	1
23	39.0	male	Government Employee	Yogyakarta	9.326527e+06	1.0	1.0	59.770940	179.201384	Pantone	1

bald_probability 데이터 셋(14개의 컬럼들은 각각 나이, 성별, 직업,출신지역,월급여, 결혼 여부(0,1), 유전,몸무게,키,사용 샴푸,흡연 여부(0,1), 교육수준, 스트레스 수준(1-10,)등을 나타냄

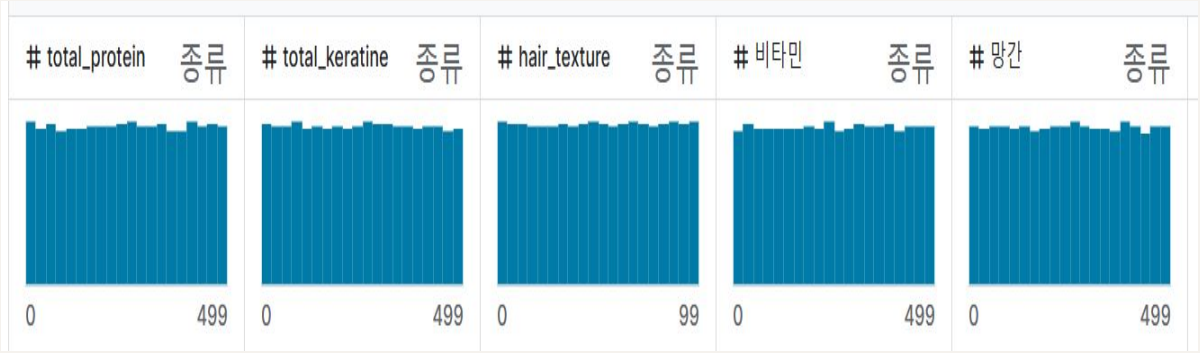
출처:kaggle

2. Hair loss 의 경우

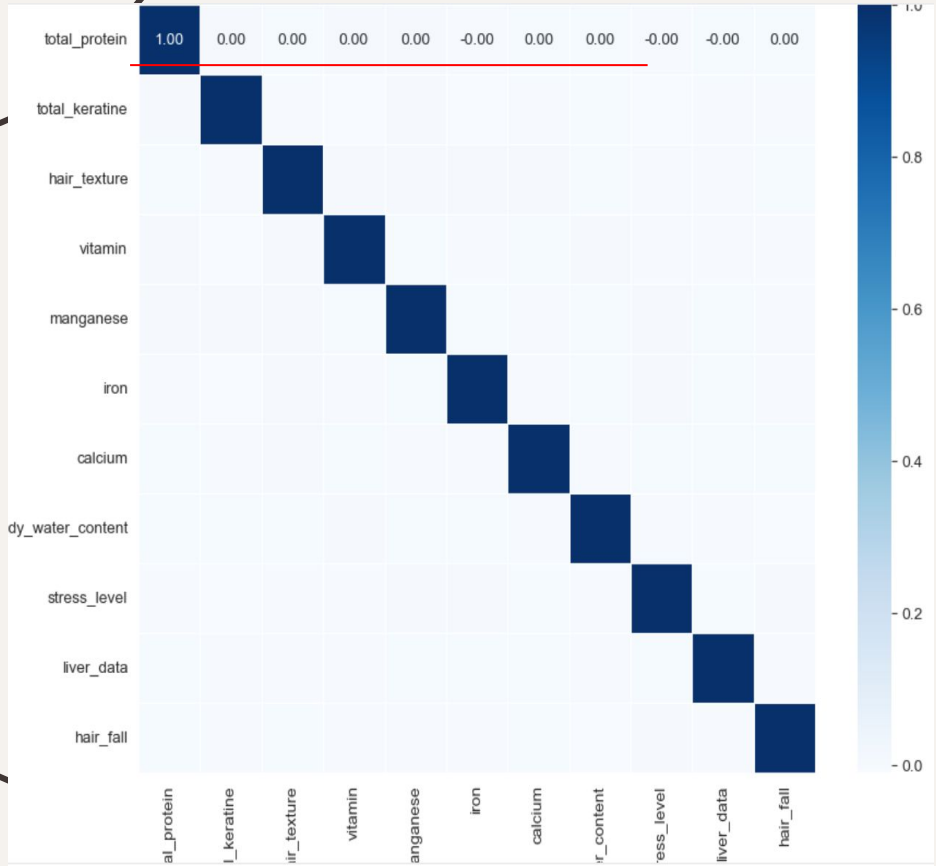
```
1 hairlossDF.isnull().sum()
Executed at 2024.03.10 22:26:34 in 31ms
```

1-10 Length: 11, dtype: int64

	<unnamed>
total_protein	0
total_keratine	0
hair_texture	0
vitamin	0
manganese	0
iron	0
calcium	0
body_water_content	0
stress_level	0
liver_data	0



먼저 hair loss data의 경우, 결측치나 두드러지는 값 없이 고르게 분포 하기는 하지만...



Whoa!

상관관계가 제대로 되는게 하나도 없음을 볼 수 있었고....

1. 데이터 준비

```
from sklearn.ensemble import RandomForestClassifier
```
Executed at 2024.03.10 21:50:32 in 85ms

```
en_model=RandomForestClassifier(random_state=55)
en_model.fit(X_train,y_train)
```
Executed at 2024.03.10 21:51:55 in 1m 23s 546ms

RandomForestClassifier

RandomForestClassifier(random_state=55)

```
en_predictions=en_model.predict(X_test)
```
Executed at 2024.03.10 21:51:57 in 2s 101ms

```
en_score=en_model.score(X_test,y_test)
```
Executed at 2024.03.10 21:52:00 in 2s 119ms

```
en_score
```
Executed at 2024.03.10 21:52:00 in 39ms

```
0.16165
```

처참한 스코어도 볼 수 있었다.

당신의 탈모, 머리속 영양과는 큰 관련이 없다...

다행입니다.

이제 머리에 좋다는 영양제는

전진하는 당신의 탈모를 막을 수 없어요

2. bald_probability 의 경우

```
bproDF.corr(numeric_only=True)
```

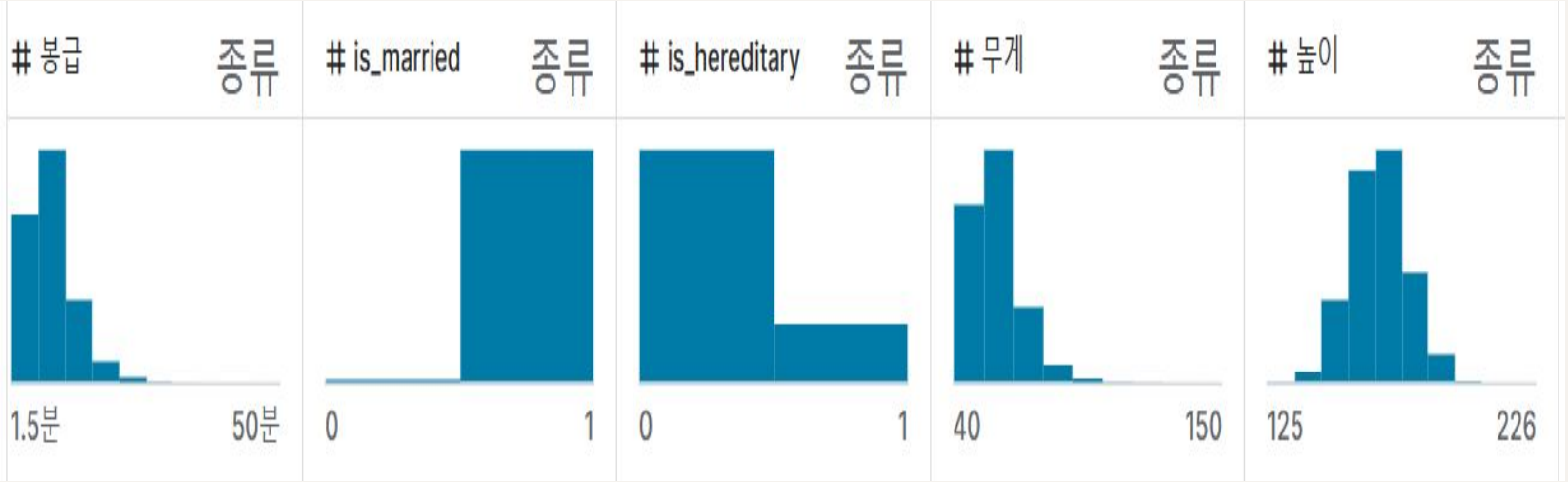
Executed at 2024.03.10 22:04:29 in 637ms

10 rows | 10 rows × 10 columns

	age	salary	is_married	is_hereditary	weight	height	is_smoker	stress	bald_prob
age	1.000000	0.011377	0.335560	-0.012923	0.003425	-0.005570	0.008551	-0.027253	0.351014
salary	0.011377	1.000000	-0.012801	-0.003772	0.009982	0.014072	-0.005680	0.014862	0.058547
is_married	0.335560	-0.012801	1.000000	-0.006400	0.007821	0.010043	-0.001654	-0.016736	0.121769
is_hereditary	-0.012923	-0.003772	-0.006400	1.000000	0.006798	0.003418	-0.014366	-0.003837	0.437712
weight	0.003425	0.009982	0.007821	0.006798	1.000000	0.389833	-0.001583	-0.012058	-0.000390
height	-0.005570	0.014072	0.010043	0.003418	0.389833	1.000000	-0.001122	0.005670	0.005765
is_smoker	0.008551	-0.005680	-0.001654	-0.014366	-0.001583	-0.001122	1.000000	-0.013973	0.261299
stress	-0.027253	0.014862	-0.016736	-0.003837	-0.012058	0.005670	-0.013973	1.000000	0.287893
bald_prob	0.351014	0.058547	0.121769	0.437712	-0.000390	0.005765	0.261299	0.287893	1.000000
BMI	0.008295	0.002315	0.006717	0.005687	0.922347	0.022041	-0.001942	-0.015880	-0.001859

이 데이터 셋은 , 아까만큼 무참한 상관관계를 보여주지는 않지만...

2. bald_probability 의 경우



?????? 범주형이 너무.....

2. bald_probability 의 경우

```
bproDF.isna().sum()
```
Executed at 2024.03.10 22:04:27 in 82ms

1-10 Length: 14, dtype: int64

	<unnamed>
age	85
gender	75
job_role	1300
province	86
salary	73
is_married	70
is_hereditary	88
weight	56
height	74
shampoo	59

결측치가 저렇게 뜬 이때 뭔가 잘못된걸 느꼈다.

2. bald_probability 의 경우

```
for i in bproDF[bproDF['gender'].isnull()].index:
    if bproDF['height'][i] >= 158:
        bproDF.loc[i, 'gender'] = 'male'
    else:
        bproDF.loc[i, 'gender'] = 'female'

print(bproDF)
```

피쳐를 채우기 위해 신장(height) 데이터를 통해 성별의 빈 값에 임의로 집어넣거나..
(인도네시아 남성 평균 신장=158cm를 기준으로 이상을 남성으로, 미만을 여성으로..)

2. bald_probability 의 경우

```
bproDF['age_range'] = bproDF['age'].apply(lambda x: '0+' if x < 10
                                    else '10+' if 10 <= x < 20
                                    else '20+' if 20 <= x < 30
                                    else '30+' if 30 <= x < 40
                                    else '40+' if 40 <= x < 50
                                    else '50+' if 50 <= x < 60
                                    else '60+' if 60 <= x < 70
                                    else '70+' if 70 <= x < 80
                                    else '80+')
```

연령별로 나이를 구분해보기도 하고..

2. bald_probability 의 경우

```
broDF['salary_range'] = broDF['salary'].apply(lambda x: 'low' if x < 3070000
                                                else 'middle' if 3070000 <= x < 8000000
                                                else 'high')
```

인도네시아 평균 월급여를 기준으로 3계급으로 나누기도 했지만..

2. bald_probability 의 경우

피쳐 대신 내 머리만 빠질 뿐이었기에...

2. bald_probability 의 경우

```
mean_salary=bproDF['salary'].mean()
bproDF['salary'].fillna(mean_salary,inplace=True)
Executed at 2024.03.10 22:04:28 in 423ms

mean_married=bproDF['is_married'].mean()
bproDF['is_married'].fillna(round(mean_married),inplace=True)
Executed at 2024.03.10 22:04:28 in 437ms

mean_hereditary=bproDF['is_hereditary'].mean()
bproDF['is_hereditary'].fillna(round(mean_hereditary),inplace=True)
Executed at 2024.03.10 22:04:28 in 447ms

mean_weight=bproDF['weight'].mean()
bproDF['weight'].fillna(mean_weight,inplace=True)
Executed at 2024.03.10 22:04:28 in 468ms

mean_height=bproDF['height'].mean()
bproDF['height'].fillna(mean_height,inplace=True)
Executed at 2024.03.10 22:04:28 in 455ms

mean_smoke=bproDF['is_smoker'].mean()
bproDF['is_smoker'].fillna(round(mean_smoke),inplace=True)
Executed at 2024.03.10 22:04:28 in 466ms

mean_stress=bproDF['stress'].mean()
bproDF['stress'].fillna(round(mean_stress),inplace=True)
Executed at 2024.03.10 22:04:28 in 479ms
```

Add Code Cell Ac

```
mode_edu=bproDF['education'].mode().values[0]
bproDF['education'].fillna(mode_edu,inplace=True)
Executed at 2024.03.10 22:04:28 in 288ms

mode_shampoo=bproDF['shampoo'].mode().values[0]
bproDF['shampoo'].fillna(mode_shampoo,inplace=True)
Executed at 2024.03.10 22:04:28 in 296ms

#텅 빈 값 알아서 넣어주기
Executed at 2024.03.10 22:04:28 in 407ms

bproDF['job_role'].fillna('Unspecified',inplace=True)
Executed at 2024.03.10 22:04:28 in 421ms

Executed at 2024.03.10 22:04:28 in 410ms

bproDF['province'].fillna('Others',inplace=True)
Executed at 2024.03.10 22:04:28 in 423ms
```

방해가 되는 null값들을 임의로 mean이나 mode를 사용해 바꿀수 밖에 없었다.

2. bald_probability 의 경우

```
select=['age_range', 'stress_range', 'is_married', 'is_smoker','family_hereditary']
```

Executed at 2024.03.10 22:04:29 in 593ms

```
dummies=pd.get_dummies(bproDF[select])
```

Executed at 2024.03.10 22:04:29 in 593ms

```
feature=dummies
target=bproDF['probability']
```

Executed at 2024.03.10 22:04:29 in 582ms

위에서 빨간 줄로 그었던, 상대적으로 corr이 높았던 값들을 피쳐로 설정하여

2. bald_probability 의 경우

[1-1] LogisticRegression

```
LR_model=LogisticRegression()

LR_model.fit(X_train_sc,y_train)
```

Executed at 2024.03.10 22:04:29 in 535ms

LogisticRegression
LogisticRegression()

```
#LR 모델의 스코어 내기
LR_train_score=LR_model.score(X_train_sc,y_train)
LR_test_score=LR_model.score(X_test,y_test)
```

Executed at 2024.03.10 22:04:29 in 671ms

```
print(f'Training score: {LR_train_score}', f'Test score: {LR_test_score}')
```

Executed at 2024.03.10 22:04:29 in 643ms

```
Training score: 0.7756451612903226 Test score: 0.7812903225806451
```

과소적합으로 보임

Logisticregression 모델에 핏하여 스코어를 산출해보고

2. bald_probability 의 경우

```
print(f'[LogisticRegression] Train score: {LR_model.score(X_train_sc,y_train)},Test score: {LR_model.score(X_test,y_test)}')

print(f'[DecisionTreeClassifier] Train score: {dt_model.score(X_train_sc,y_train)},Test score: {dt_model.score(X_test,y_test)}')

print(f'[RandomForestClassifier] Train score: {svc_model.score(X_train_sc,y_train)},Test score: {svc_model.score(X_test,y_test)}')
```

Executed at 2024.03.10 23:45:51 in 2s 86ms

```
[LogisticRegression] Train score: 0.7756451612903226,Test score: 0.7812903225806451
[DecisionTreeClassifier] Train score: 0.7775806451612903,Test score: 0.7793548387096774
[RandomForestClassifier] Train score: 0.7774193548387097,Test score: 0.7806451612903226
```

같은 데이터를 다른 모델로 피팅 해본다.

3. 나름의 연관성- 유전(heredity)

corr=0.437812

source=3대가 빡빡이.png

3. 나름의 연관성- 스트레스(stress)

corr=0.287893

source=여기서 투블럭?.png

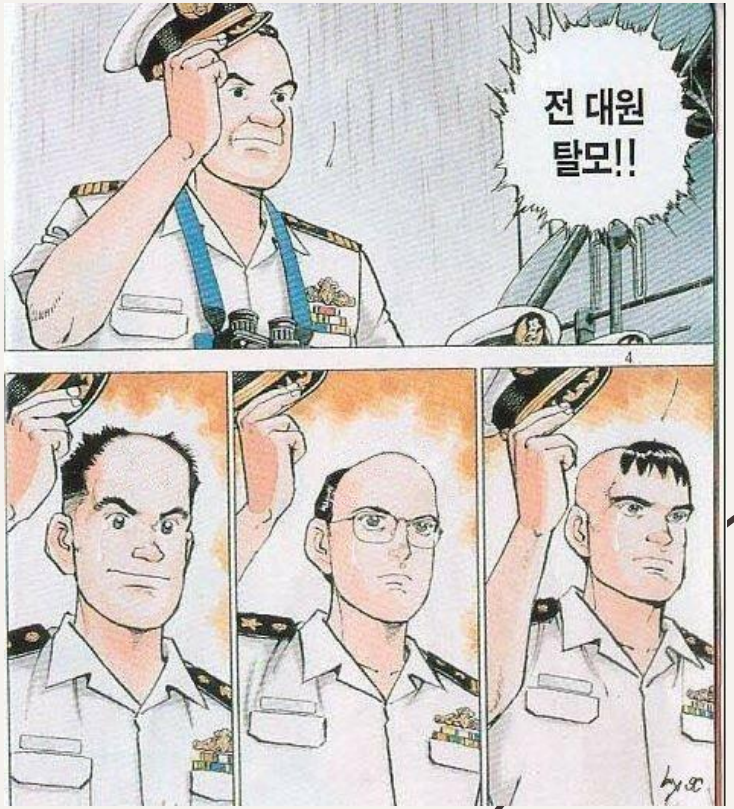
3. 나름의 연관성- 흡연(smoking)

corr=0.261299

source=담배피는 대머리.png

다음에 더 해야 할 것

- 한국 탈모 데이터를 조사해서 모델을 만들어 보기
- 다양한 사람들의 두피 정보를 이미지 파일로 받아 딥러닝 도전해 보기
- 만족할 만한 정교한 모델 만들기



04

탈모팀 - 김동현

Insert a subtitle here if you need it

2. bald_probability 의 경우

```
result_predict=classification_report(y_test,y_pred)

print(f'결과표:\n{result_predict}')
```

Executed at 2024.03.10 23:45:38 in 126ms

```
결과표:
              precision    recall  f1-score   support

        High       0.80      0.88      0.84      1003
         Low       0.73      0.60      0.66       547

    accuracy                           0.78      1550
   macro avg       0.77      0.74      0.75      1550
weighted avg       0.78      0.78      0.78      1550
```

재현율,정확도 등의 값들을 산출하는 리포트를 출력하였다.

데이터 전처리

```
# 중복 데이터 82개 삭제
baldDF = baldDF.drop_duplicates()
baldDF.shape
```

✓ 0.0s

```
(7835, 14)
```

```
# bald_prob가 결측된 데이터 삭제
baldDF = baldDF.dropna(subset = ['bald_prob'])
baldDF.shape
```

✓ 0.0s

```
(7757, 14)
```

```
# 중복 데이터 82개 삭제
baldDF = baldDF.drop_duplicates()
baldDF.shape
```

✓ 0.0s

```
(7835, 14)
```

```
# bald_prob가 결측된 데이터 삭제
baldDF = baldDF.dropna(subset = ['bald_prob'])
baldDF.shape
```

✓ 0.0s

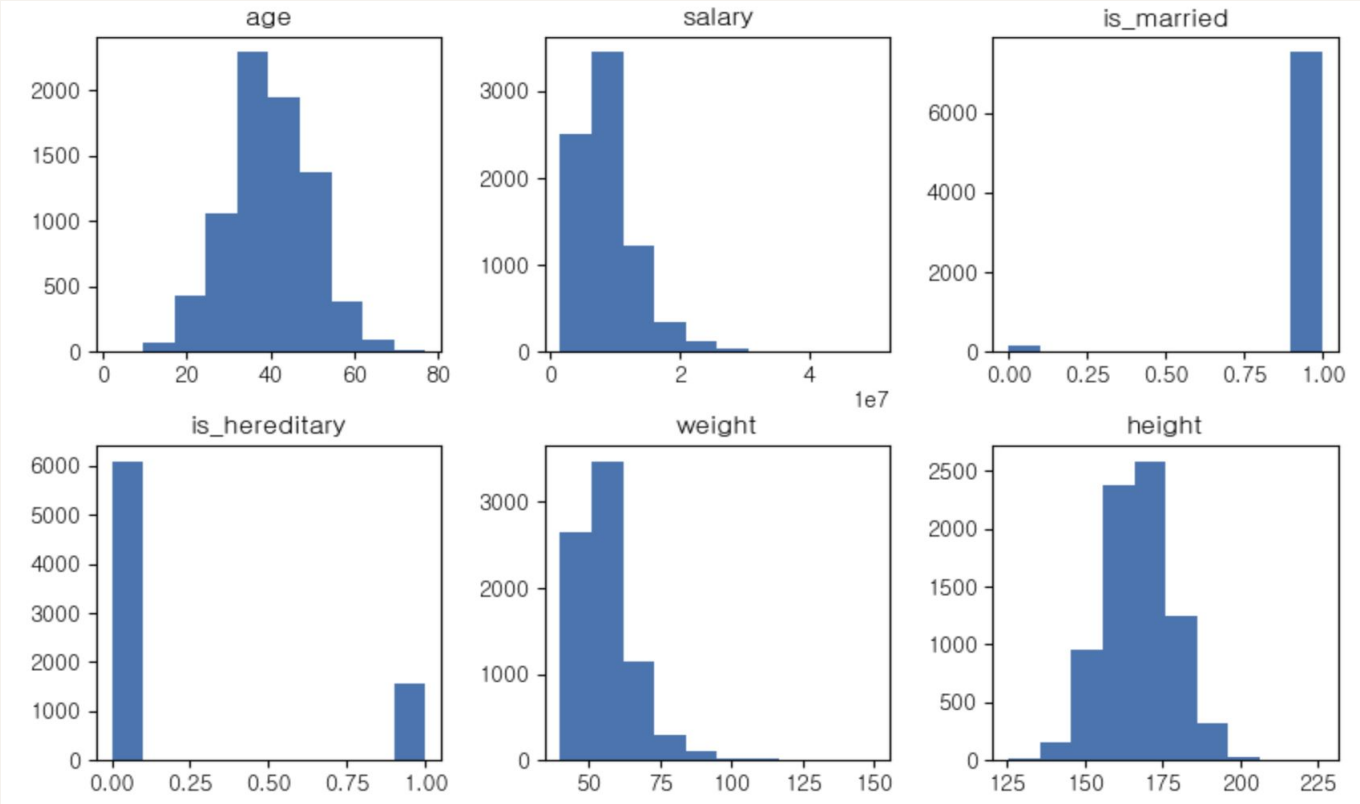
```
(7757, 14)
```

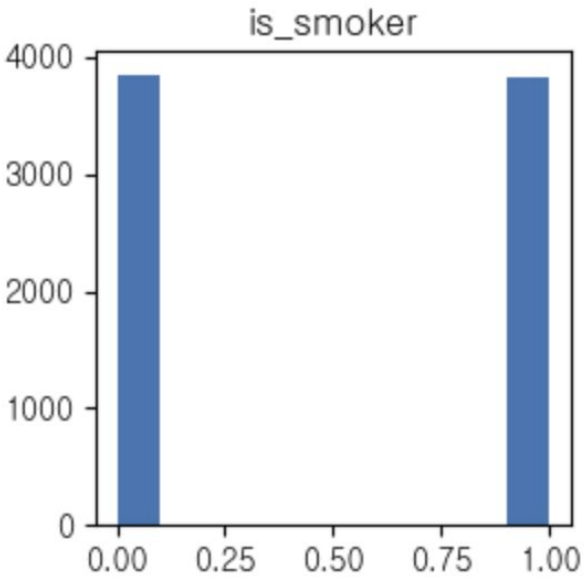
```
baldDF.nunique()
```

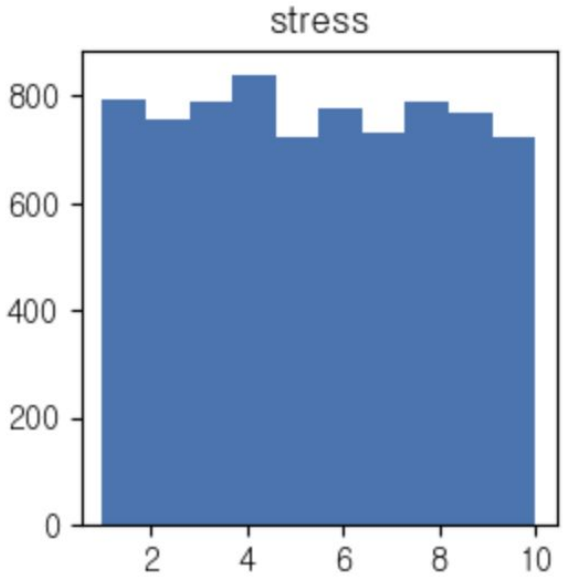
✓ 0.0s

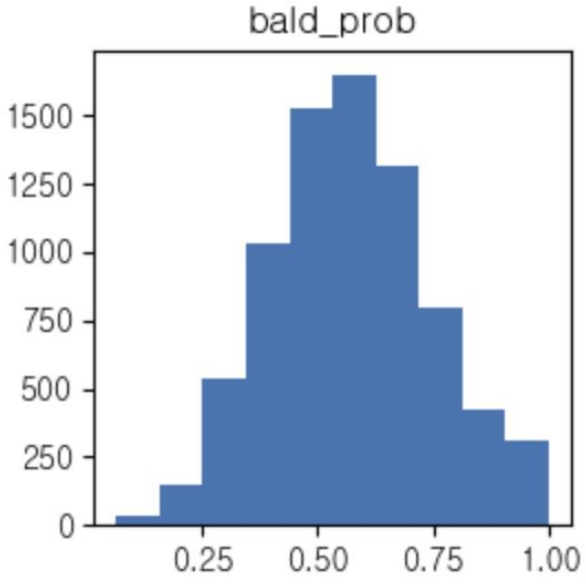
```
age              70
gender            2
job_role          3
province         34
salary         7686
is_married        2
is_hereditary     2
weight         7701
height         7685
shampoo           5
is_smoker         2
education         6
stress           10
bald_prob      7659
dtype: int64
```

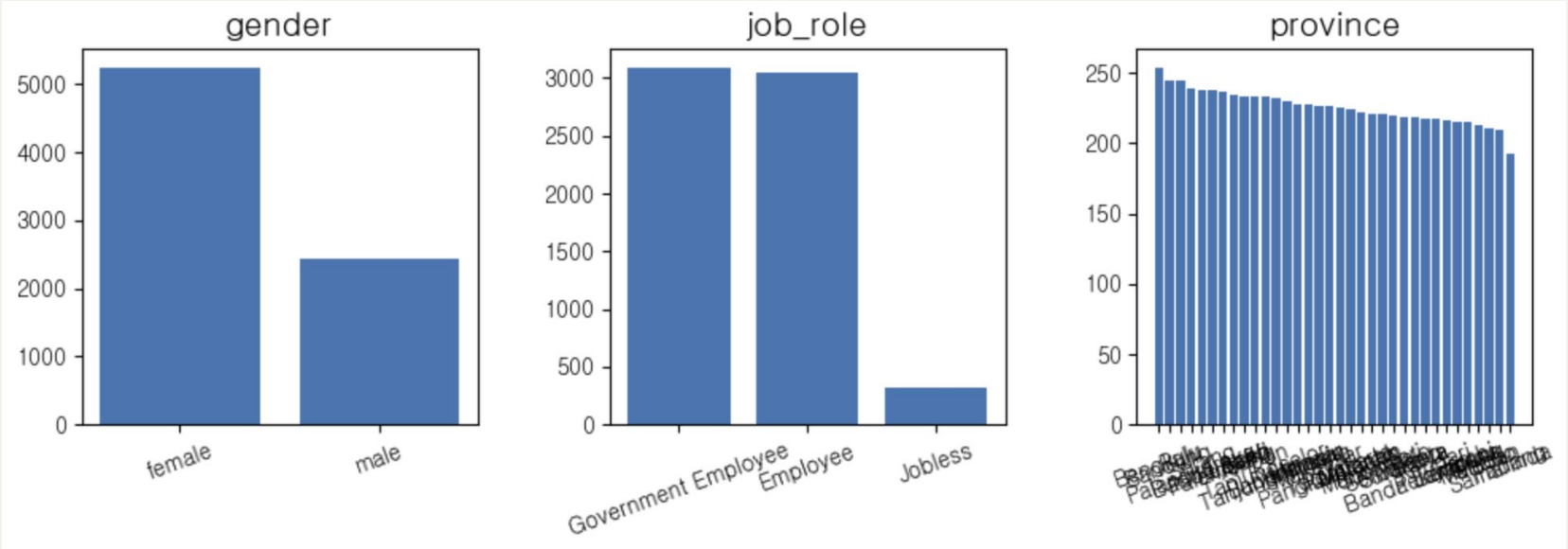
데이터 분포

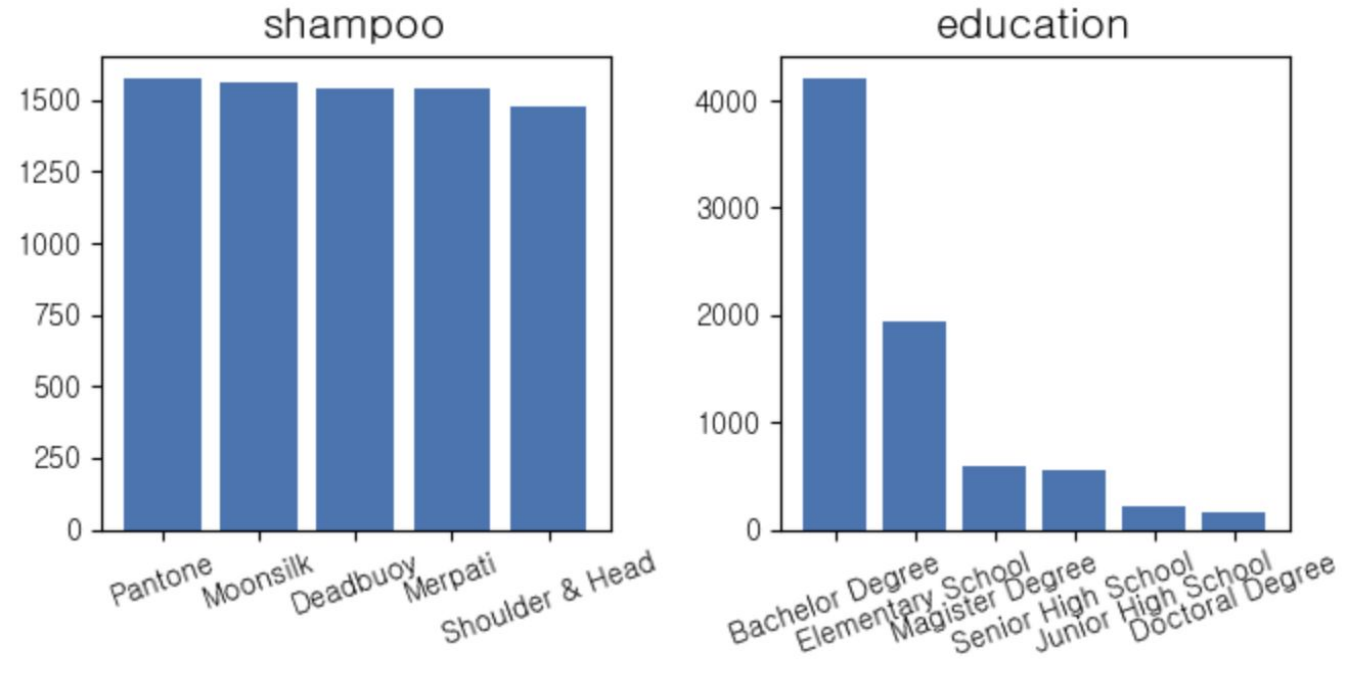

age
0
500
1000
1500
2000
0
20
40
60
80
salary
0
1000
2000
3000
0
2
4
1e7
is_married
0
2000
4000
6000
0.00
0.25
0.50
0.75
1.00
is_hereditary
0
1000
2000
3000
4000
5000
6000
0.00
0.25
0.50
0.75
1.00
weight
0
1000
2000
3000
50
75
100
125
150
height
0
500
1000
1500
2000
2500
125
150
175
200
225


is_smoker
4000
3000
2000
1000
0
0.00
0.25
0.50
0.75
1.00


stress
800
600
400
200
0
2
4
6
8
10


bald_prob
1500
1250
1000
750
500
250
0
0.25
0.50
0.75
1.00


gender
5000
4000
3000
2000
1000
0
female
male
job_role
3000
2500
2000
1500
1000
500
0
Government Employee
Employee
Jobless
province
250
200
150
100
50
0


shampoo
1500
1250
1000
750
500
250
0
Pantone
Moonsilk
Deadbuoy
Merpati
Shoulder & Head
education
4000
3000
2000
1000
0
Bachelor Degree
Elementary School
Magister Degree
Senior High School
Junior High School
Doctoral Degree

이상치 처리

```python
# Z-score 기반 이상치 탐지 함수

def find_outlier(column_list, scale = 3, df = baldDF):
    for col in column_list:
        mean_value = df[col].mean()
        std_value = df[col].std()

        base =scale
        z_data = (df[col] - mean_value) / std_value
        mask = z_data.abs() > base
        print(f'{col} 특성의 이상치 개수 : {z_data[mask].count()}개\n')
```

✓ 0.0s

```python
find_outlier(['age', 'height'])
```

✓ 0.0s

```
age 특성의 이상치 개수 : 17개

height 특성의 이상치 개수 : 23개
```

이상치 처리

```python
# 사분위수 기반 이상치 탐지 함수
def find_outlier2(column_list, scale = 1.5, df = baldDF):
    for col in column_list:
        q1 = df[col].quantile(0.25)
        q3 = df[col].quantile(0.75)
        iqr = q3 - q1

        # 이상치로 판단할 기준이 되는 값
        lower = q1 - scale*iqr
        upper = q3 + scale*iqr
        mask = (df[col] < lower) | (df[col] > upper)

        print(f'{col} 특성의 이상치 개수 : {df[col][mask].count()}개\n')
```

✓ 0.0s

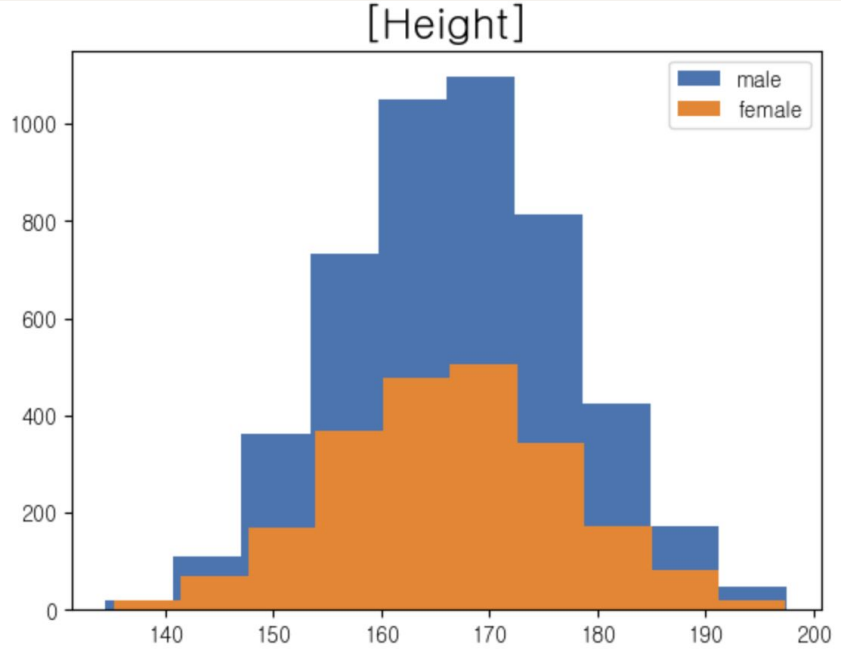
```python
# 사분위수 기반
find_outlier2(['salary', 'weight'])
```

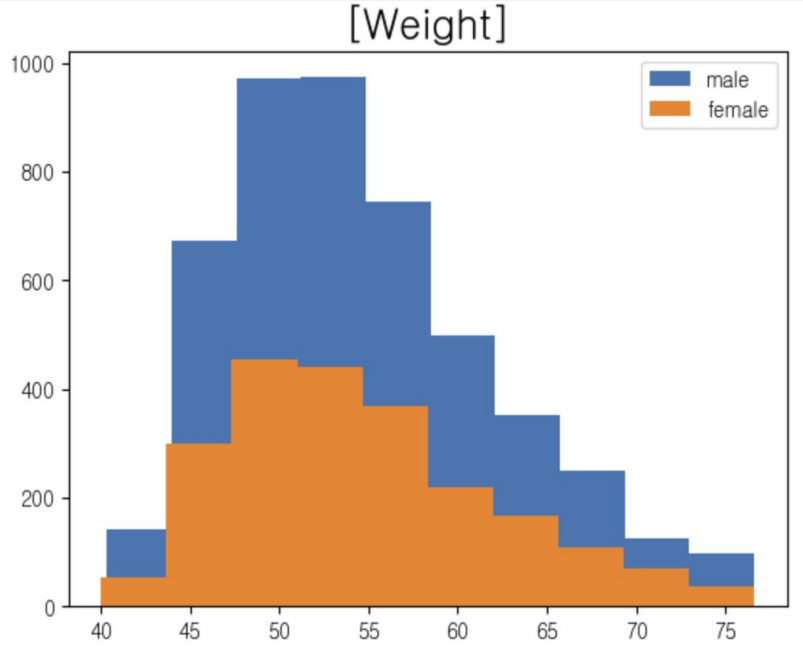
✓ 0.0s

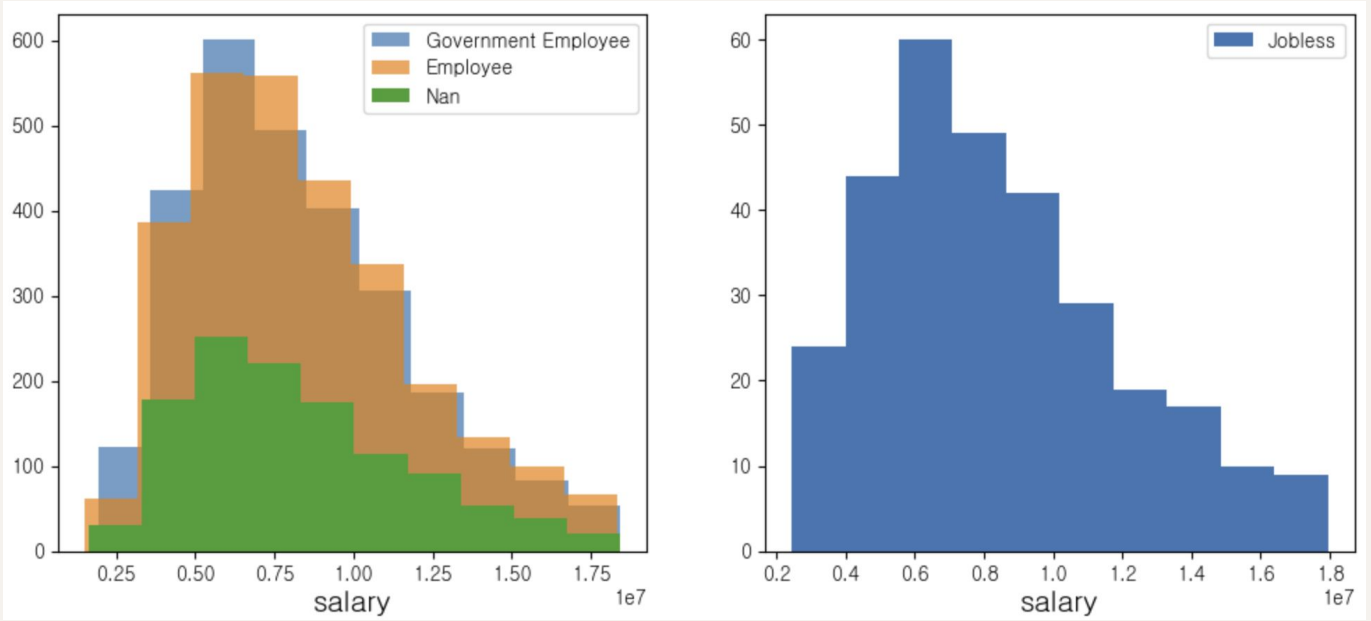
```
salary 특성의 이상치 개수 : 277개

weight 특성의 이상치 개수 : 276개
```

피처당 결측치 수

```
age                80
gender             68
job_role         1192
province           81
salary             68
is_hereditary      82
weight             54
height             62
shampoo            55
is_smoker          64
education          65
stress             55
bald_prob           0
dtype: int64
```

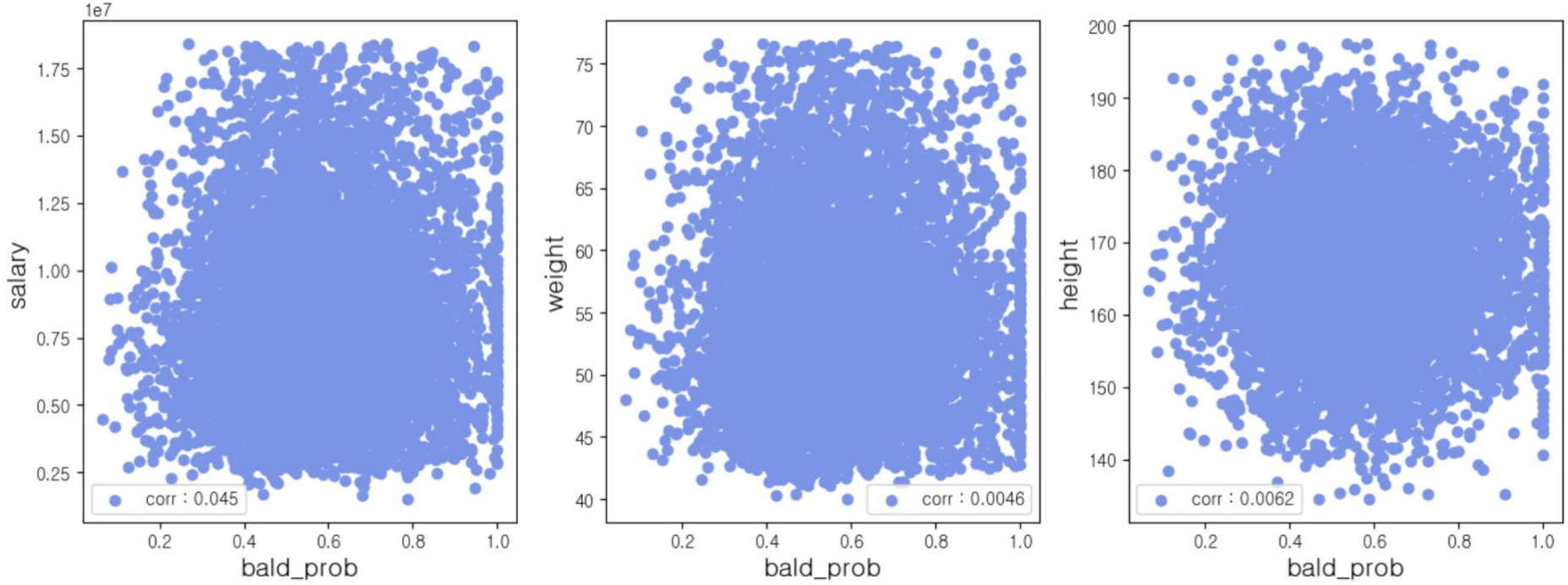

[Height]
male
female
1000
800
600
400
200
0
140
150
160
170
180
190
200


[Weight]
male
female
1000
800
600
400
200
0
40
45
50
55
60
65
70
75




```
# 결측값
baldDF['job_role'] = baldDF['job_role'].fillna('etc')
```

✓ 0.0s


1e7
salary
bald_prob
corr : 0.045
weight
bald_prob
corr : 0.0046
height
bald_prob
corr : 0.0062

최빈값 사용

- age, gender, province, is_hereditary, shampoo, is_smoker, education, stress 결측치 처리

```
simp = SimpleImputer(missing_values = np.nan, strategy = 'most_frequent')
df = baldDF.copy()
df = df[['age', 'gender', 'province', 'is_hereditary', 'shampoo', 'is_smoker', 'education', 'stress']]
df = pd.DataFrame(simp.fit_transform(df), columns = df.columns, index = df.index)
df.isna().sum()
```

✓ 0.0s

```
age              0
gender           0
province         0
is_hereditary    0
shampoo          0
is_smoker        0
education        0
stress           0
dtype: int64
```

```
# 각 피처의 결측값을 최빈값으로 대체
for col in ['age', 'gender', 'province', 'is_hereditary', 'shampoo', 'is_smoker', 'education', 'stress']:
    baldDF[col] = df[col].values
```

라벨 인코딩

```
# 범주형 데이터 피처 중 실수형은 정수형으로 변환
baldDF = baldDF.astype({'age' : int, 'is_hereditary' : int, 'is_smoker' : int, 'stress' : int})
```

0.0s

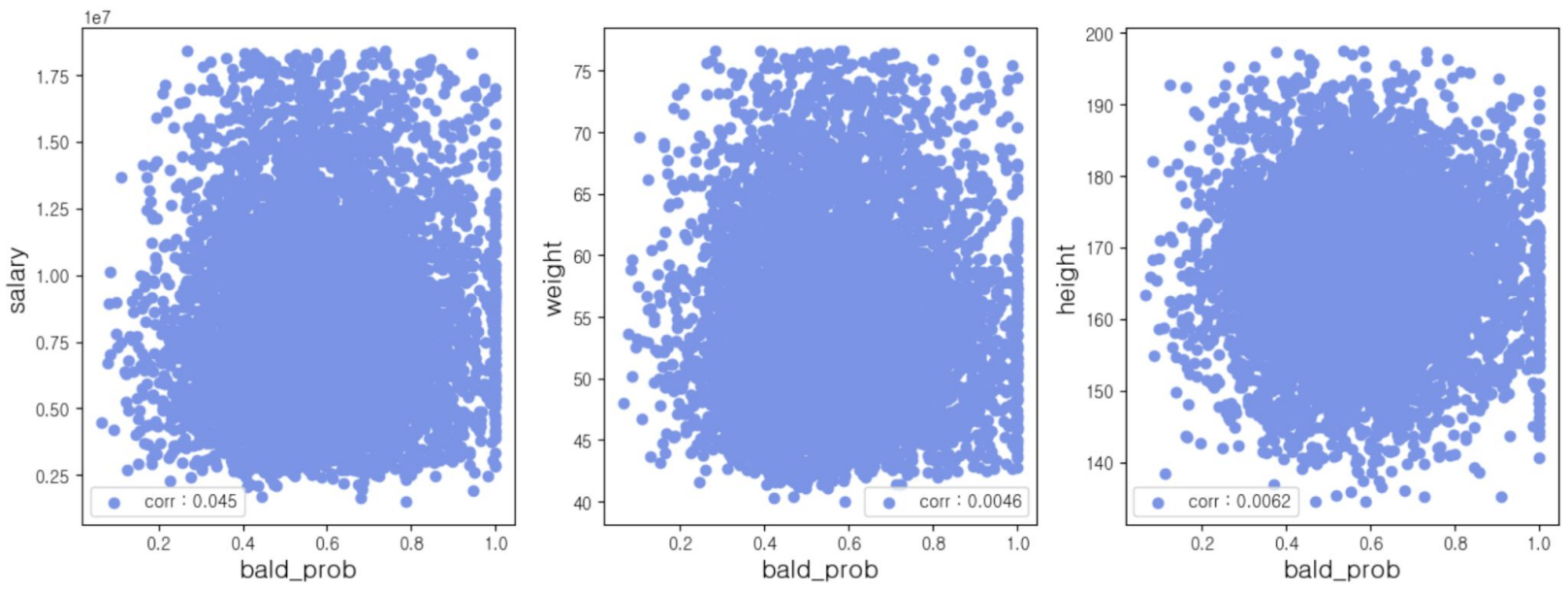
```
# gender 피처 데이터는 0, 1로 변환
baldDF['gender'] = baldDF['gender'].replace({'female' : 0, 'male' : 1})
```

0.0s

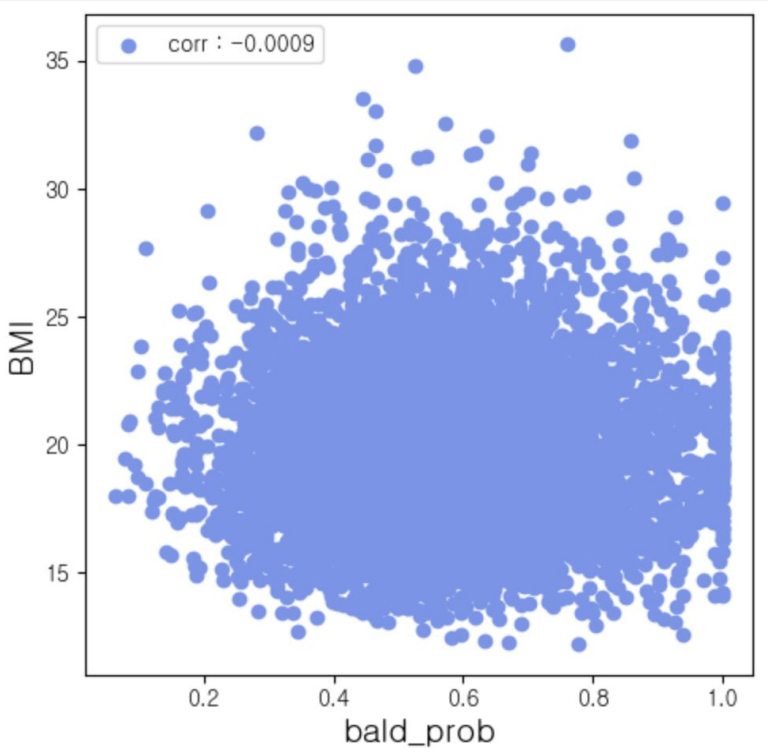
```
# education 피처 데이터는 학위 순으로 1 ~ 6으로 배정

baldDF['education'] = baldDF['education'].replace({'Elementary School' : 1,
                                                    'Junior High School' : 2,
                                                    'Senior High School' :3,
                                                    'Bachelor Degree' : 4,
                                                    'Magister Degree' : 5,
                                                    'Doctoral Degree' : 6})
```

```python
# 남은 범주형 데이터 라벨 인코딩
for col in ['job_role', 'province', 'shampoo']:
    le = LabelEncoder()
    baldDF[col] = le.fit_transform(baldDF[col])
```

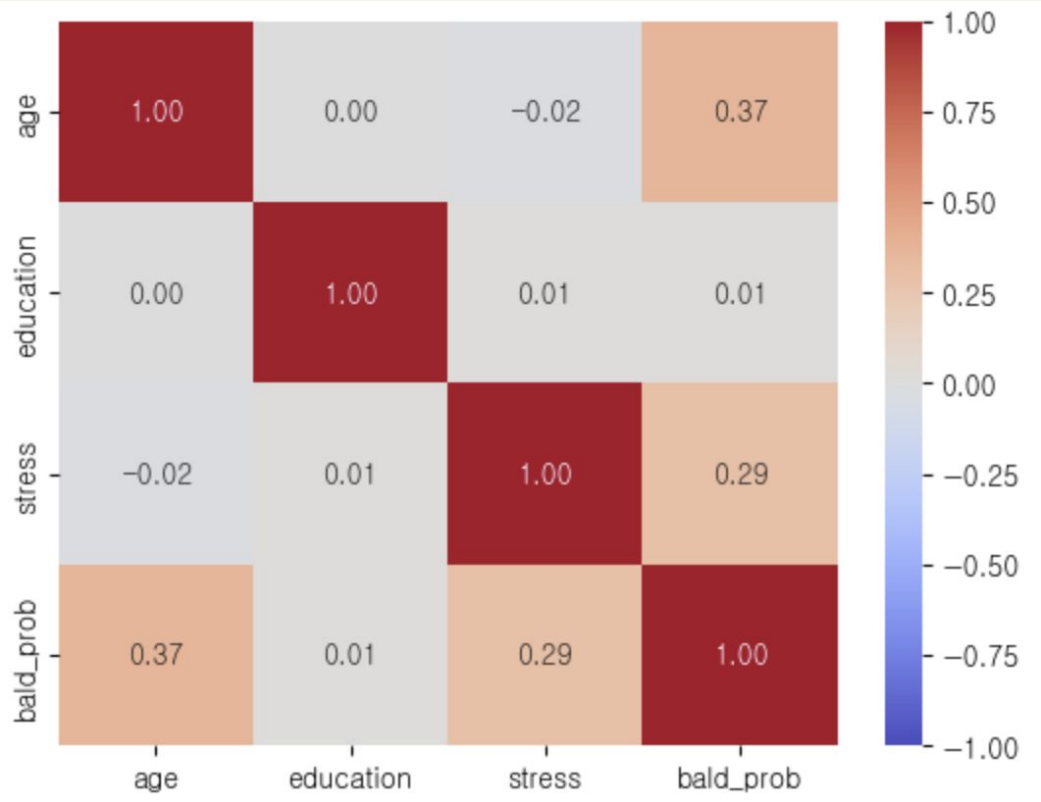
연속형 데이터


1e7
1.75
1.50
1.25
1.00
0.75
0.50
0.25
salary
corr : 0.045
0.2
0.4
0.6
0.8
1.0
bald_prob
75
70
65
60
55
50
45
40
weight
corr : 0.0046
bald_prob
200
190
180
170
160
150
140
height
corr : 0.0062
bald_prob

BMI와의 관계



순서형 데이터


age
education
stress
bald_prob
1.00
0.00
-0.02
0.37
0.00
1.00
0.01
0.01
-0.02
0.01
1.00
0.29
0.37
0.01
0.29
1.00
age
education
stress
bald_prob
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

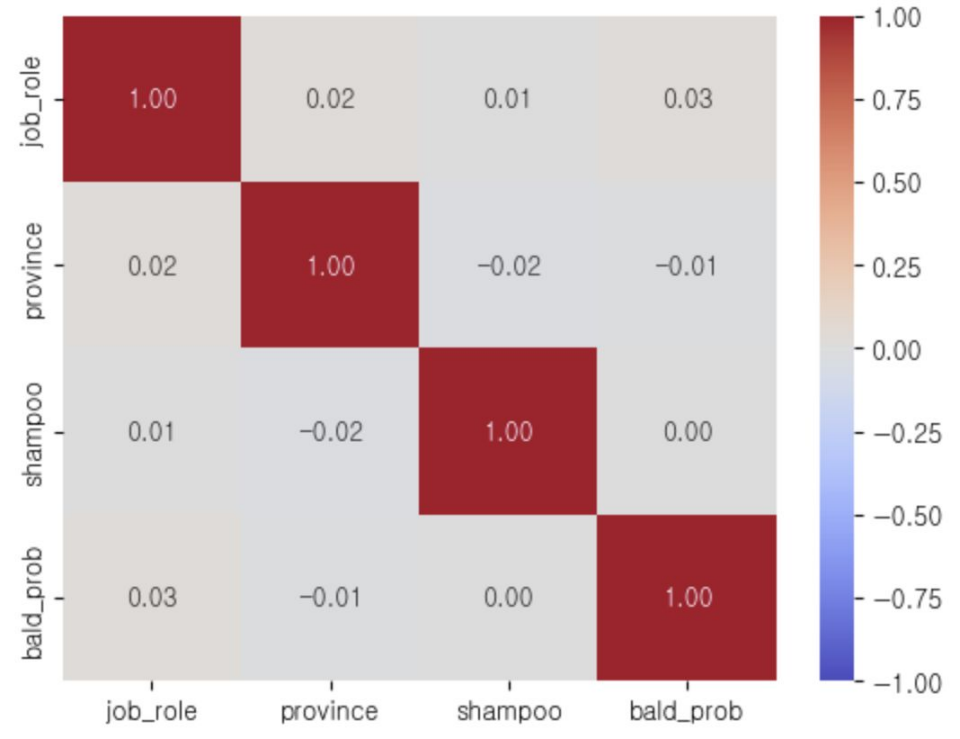
이진 범주형 데이터

gender과 bald_prob의 Biserial coef는 0.26

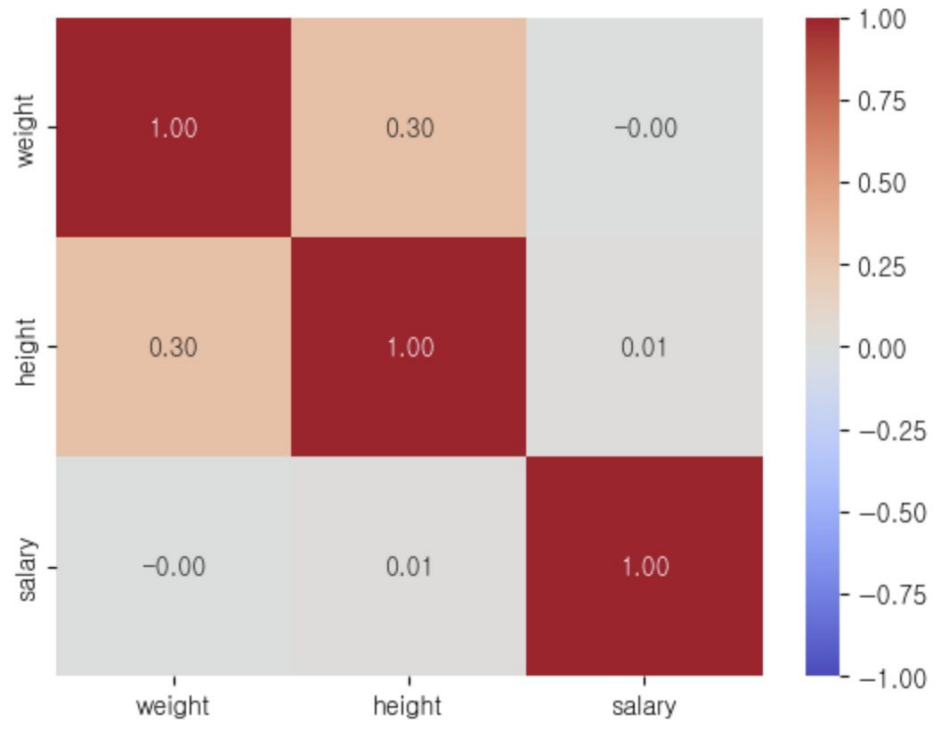
is_hereditary과 bald_prob의 Biserial coef는 0.43

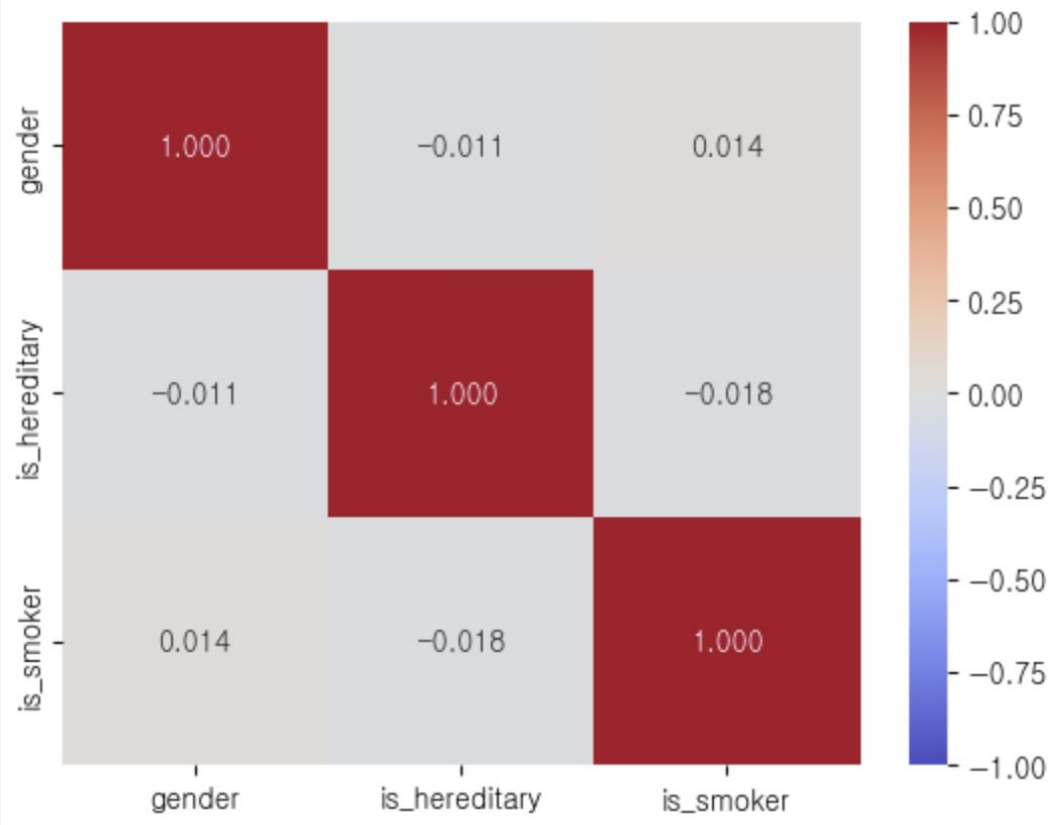
is_smoker과 bald_prob의 Biserial coef는 0.26

다중 범주형 데이터


job_role
province
shampoo
bald_prob
1.00
0.02
0.01
0.03
0.02
1.00
−0.02
−0.01
0.01
−0.02
1.00
0.00
0.03
−0.01
0.00
1.00
job_role
province
shampoo
bald_prob
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

다중 공선성 확인


weight
height
salary
1.00
0.30
-0.00
0.30
1.00
0.01
-0.00
0.01
1.00
weight
height
salary
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


gender
is_hereditary
is_smoker
1.000
-0.011
0.014
-0.011
1.000
-0.018
0.014
-0.018
1.000
gender
is_hereditary
is_smoker
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


age
job_role
province
shampoo
education
stress
1.000 0.025 0.000 0.000 0.000 0.000
0.025 1.000 0.021 0.000 0.000 0.000
0.000 0.021 1.000 0.029 0.015 0.000
0.000 0.000 0.029 1.000 0.000 0.000
0.000 0.000 0.015 0.000 1.000 0.012
0.000 0.000 0.000 0.000 0.012 1.000
age
job_role
province
shampoo
education
stress
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

학습 진행

```
xgb_model = XGBRegressor()
n_cv = 5
params = {'learning_rate' : [0.05, 0.1, 0.2],
          'max_depth' : [5, 6, 7, 8],
          'colsample_bytree' : [0.5, 1.0]
          }

# 하이퍼파라미터 조합된 모델 생성 후 교차검증 데이터셋으로 학습
# 진행해주는 인스턴스
gs = GridSearchCV(xgb_model, param_grid = params, cv = n_cv)
```

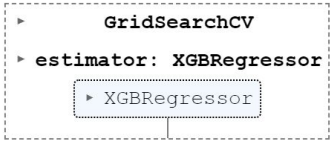
✓ 0.0s

```
# 조합된 모델들의 학습 진행
result = gs.fit(X_train, y_train)
```

✓ 31.5s

```
# 결과 확인
result
```

✓ 0.0s




```
# 조합된 모델 중 최고의 성능을 내는 하이퍼파라미터 값
gs.best_params_
```

✓ 0.0s

```
{'colsample_bytree': 0.5, 'learning_rate': 0.1, 'max_depth': 5}
```

```
# 조합된 모델 중 최고의 성능을 내는 모델 인스턴스
my_best_model = gs.best_estimator_
```

✓ 0.0s

```
# 조합된 모델 중 최고 점수
gs.best_score_
```

✓ 0.0s

```
0.8380826480566185
```

```python
import pandas as pd
import numpy as np

model = load(model_file)

data = input('알아서 입력하세요 : ')
if len(data):
    data_list = list(map(float, data.split(',')))  # 넣을 데이터에 컬럼명이 없으면 경고 메세지 뜬다.
    feature_name = ['age', 'education', 'gender', 'height', 'is_hereditary', 'is_smoker',
       'job_role', 'salary', 'stress', 'weight']
    df = pd.DataFrame(dict(zip(feature_name, data_list)), index = [0])
    print(df)

    # 모델의 predict(2D)
    pre_prob = model.predict(df)
    #proba = np.round(np.max(model.predict_proba(df)[0]) * 100, 1)
    print(f'해당 데이터를 가진 사람이 hair loss일 확률은 {np.round(pre_prob*100, 2)}% 입니다.')
else:
    print('입력된 정보가 없습니다.')
```

✓ 32.4s

```
    age  education  gender  height  is_hereditary  is_smoker  job_role  \
0  33.0        5.0     1.0   180.0            0.0        0.0       3.0

       salary  stress  weight
0  9448807.05     5.0    73.0
해당 데이터를 가진 사람이 hair loss일 확률은 [36.2]% 입니다.
```

< 담당 역할 >

	이윤서	전진우	고우석	김동현
준비	자료 조사	깃 관리	주제 선정	주제 선정
주제	1. 비만		2. 탈모	
분석	분류	회귀	회귀	회귀