Import Libraries

In [159…

```
import pandas as pd
import seaborn as sns
import numpy as np
import random
import os
import datetime

from autogluon.tabular import TabularPredictor

from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier

# 한글 그래프
import matplotlib.pyplot as plt
import platform
if platform.system() == 'Darwin': # Mac
        plt.rc('font', family='AppleGothic')
elif platform.system() == 'Windows': # Window
        plt.rc('font', family='Malgun Gothic')
elif platform.system() == 'Linux': # 리눅스 (Colab)
        #!wget
"https://www.wfonts.com/download/data/2016/06/13/malgun-
gothic/malgun.ttf"
        #!mv malgun.ttf /usr/share/fonts/truetype/
        #import matplotlib.font_manager as fm
        #fm._rebuild()
        plt.rc('font', family='Malgun Gothic')
plt.rcParams['axes.unicode_minus'] = False #한글 폰트 사용시 마이
너스 폰트 깨짐 해결
import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

In [160…

```
def seed_everything(seed):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)
```

```
seed_everything(42) # Seed 고정
```

Data Explanation

일반적으로 흡연 여부와 관련성이 높을 것으로 예상되는 피처

1. 시력
2. 충치
3. 공복 혈당
4. 혈압
5. 중성 지방

-

흡연 여부와 영향이 없는 것으로 예상되는 피처

1. ID
2. 나이
3. 키(cm)
4. 몸무게(kg)
5. BMI (체질량 지수)

Load Dataset

In [162…

```
# test 데이터를 분석에 활용하는 것은 Data Leakage에 해당하므로 train 데이터만 사용
train = pd.read_csv('data/train.csv')
train.head(3)
```

Out[162]:

	ID	나이	키(cm)	몸무게(kg)	BMI	시력	충치	공복혈당	혈압	중성지방	혈청크레아티닌	콜레스테롤	고밀도지단백	저밀도지단백	헤모글로빈	요단백
0	TRAIN_0000	35	170	70	24.22	1.10	1	98	40	80	1.3	211	75	120	15.9	1
1	TRAIN_0001	40	150	55	24.44	1.00	0	173	39	104	0.6	251	46	184	11.8	1
2	TRAIN_0002	60	170	50	17.30	0.75	0	96	40	61	0.8	144	43	89	15.3	1

In [163…

```
test = pd.read_csv('data/test.csv')
```

Explore Data

기술통계량 확인

In [164…
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7000 entries, 0 to 6999
Data columns (total 18 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   ID         7000 non-null   object 
 1   나이         7000 non-null   int64  
 2   키(cm)      7000 non-null   int64  
 3   몸무게(kg)    7000 non-null   int64  
 4   BMI        7000 non-null   float64
 5   시력         7000 non-null   float64
 6   충치         7000 non-null   int64  
 7   공복 혈당      7000 non-null   int64  
 8   혈압         7000 non-null   int64  
 9   중성 지방      7000 non-null   int64  
 10  혈청 크레아티닌   7000 non-null   float64
 11  콜레스테롤      7000 non-null   int64  
 12  고밀도지단백     7000 non-null   int64  
 13  저밀도지단백     7000 non-null   int64  
 14  헤모글로빈      7000 non-null   float64
 15  요 단백       7000 non-null   int64  
 16  간 효소율      7000 non-null   float64
 17  label      7000 non-null   int64  
dtypes: float64(5), int64(12), object(1)
memory usage: 984.5+ KB
```

In [166…
```
train.describe()
```

Out[166]:

	나이	키(cm)	몸무게(kg)	BMI	시력	충치	
count	7000.000000	7000.000000	7000.000000	7000.000000	7000.000000	7000.000000	7
mean	43.973571	164.781429	65.932857	24.144423	1.011650	0.227429	
std	12.063793	9.170213	12.978702	3.501945	0.427828	0.419202	
min	20.000000	135.000000	30.000000	14.270000	0.100000	0.000000	
25%	35.000000	160.000000	55.000000	21.600000	0.800000	0.000000	
50%	40.000000	165.000000	65.000000	23.880000	1.000000	0.000000	
75%	50.000000	170.000000	75.000000	26.120000	1.200000	0.000000	
max	85.000000	190.000000	130.000000	42.450000	9.900000	1.000000	

1. 나이 : 적절하게 분포되어 있음.
2. 키 : 적절하게 분포되어 있음.
3. 몸무게 : 적절하게 분포되어 있음.
4. BMI : 적절하게 분포되어 있음.
5. 시력 : max가 9.9 ---------> 데이터에 불순물 존재

6. 충치 : 있거나 (1) 없거나 (0) 적절함.
7. 공복 혈당 : max가 386 ----------> 도메인 지식으로 확인 필요
8. 혈압 : 적절하게 분포되어 있음.
9. 중성 지방 : 잘 모르겠는데 max가 999인 게 좀 이상함 -----> 확인 필요
10. 혈청 크레아티닌 : 10이면 대체 얼마나..? ---> 확인 필요
11. 콜레스테롤 : 적절해 보임
12. 고밀도지단백 : 적절해 보임
13. 저밀도지단백 : 적절해 보임
14. 헤모글로빈 : 적절해 보임
15. 요단백 : ------> 이상치 있어 보임
16. 간 효소율 : -------> 이상치 있어 보임

- 전반적으로 건강 데이터인데도 불구하고 지나치게 높은 이상치가 포함되어 있어서 확인이 필요함.
- 스케일링이 좀 필요함.

In [167…
```
# 흡연자 데이터 확인
train[train['label']==1].describe()
```

Out[167]:

	나이	키(cm)	몸무게(kg)	BMI	시력	충치	
count	2571.000000	2571.000000	2571.000000	2571.000000	2571.000000	2571.000000	25
mean	41.207701	169.531311	71.238818	24.725531	1.052820	0.281602	1(
std	11.280386	6.616517	12.014572	3.532278	0.387819	0.449868	
min	20.000000	145.000000	40.000000	15.570000	0.100000	0.000000	6
25%	35.000000	165.000000	65.000000	22.490000	0.850000	0.000000	9
50%	40.000000	170.000000	70.000000	24.220000	1.050000	0.000000	9
75%	50.000000	175.000000	80.000000	27.060000	1.200000	1.000000	1(
max	80.000000	190.000000	125.000000	40.820000	5.950000	1.000000	38

In [168…
```
# 비흡연자 데이터 확인
train[train['label']==0].describe()
```

Out[168]:

	나이	키(cm)	몸무게(kg)	BMI	시력	충치
count	4429.000000	4429.000000	4429.000000	4429.000000	4429.000000	4429.000000
mean	45.579138	162.024159	62.852788	23.807094	0.987751	0.195981
std	12.213191	9.316991	12.515393	3.439864	0.447732	0.396999
min	20.000000	135.000000	30.000000	14.270000	0.100000	0.000000
25%	40.000000	155.000000	55.000000	21.480000	0.750000	0.000000
50%	45.000000	160.000000	60.000000	23.440000	1.000000	0.000000
75%	55.000000	170.000000	70.000000	25.710000	1.200000	0.000000
max	85.000000	190.000000	130.000000	42.450000	9.900000	1.000000

In [169…
```
# 결측치 확인
train.isnull().sum().sum()
```

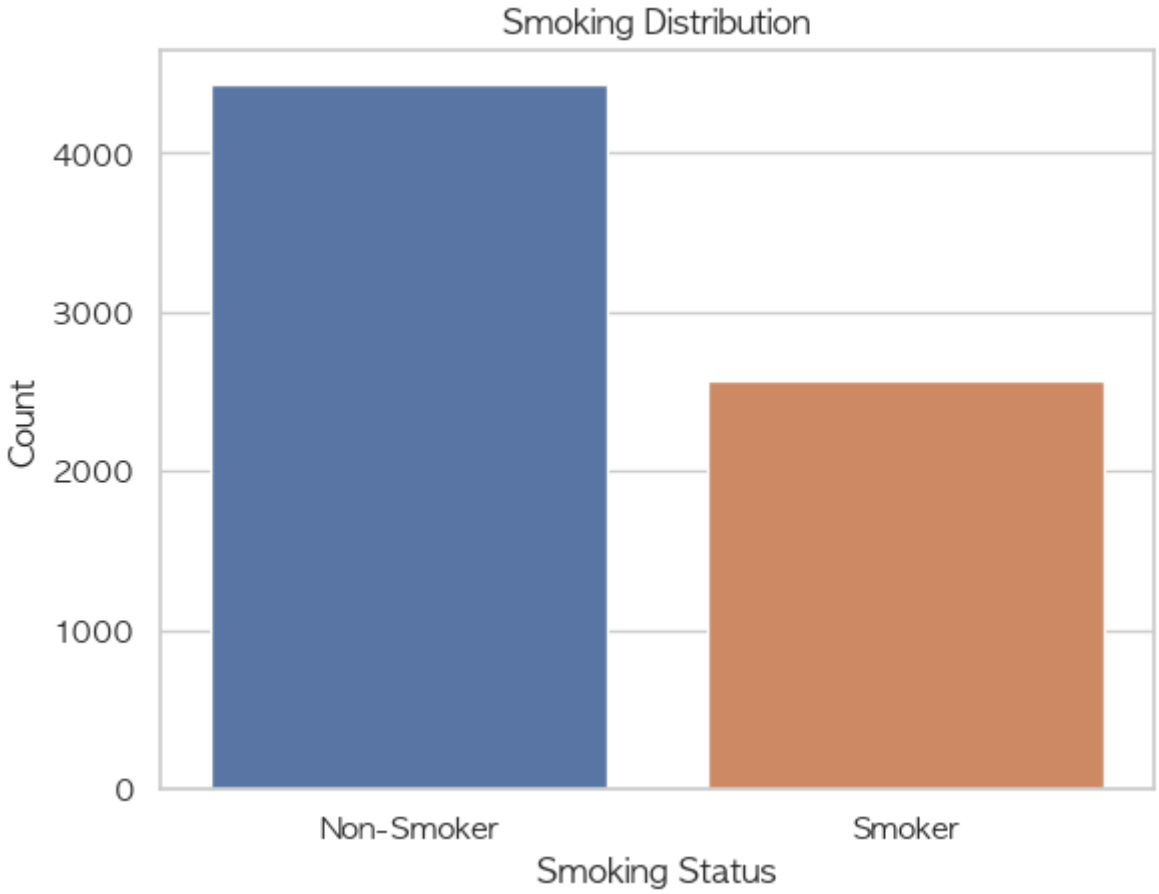
Out[169]:
```
0
```

데이터 시각화

종속변수 분포 확인

In [170…
```
# 데이터프레임이 train이라면 다음과 같이 사용
sns.countplot(x='label', data=train)

plt.title('Smoking Distribution')
plt.xlabel('Smoking Status')
plt.ylabel('Count')
plt.xticks(ticks=[0,1], labels=['Non-Smoker', 'Smoker'])
plt.show()
```



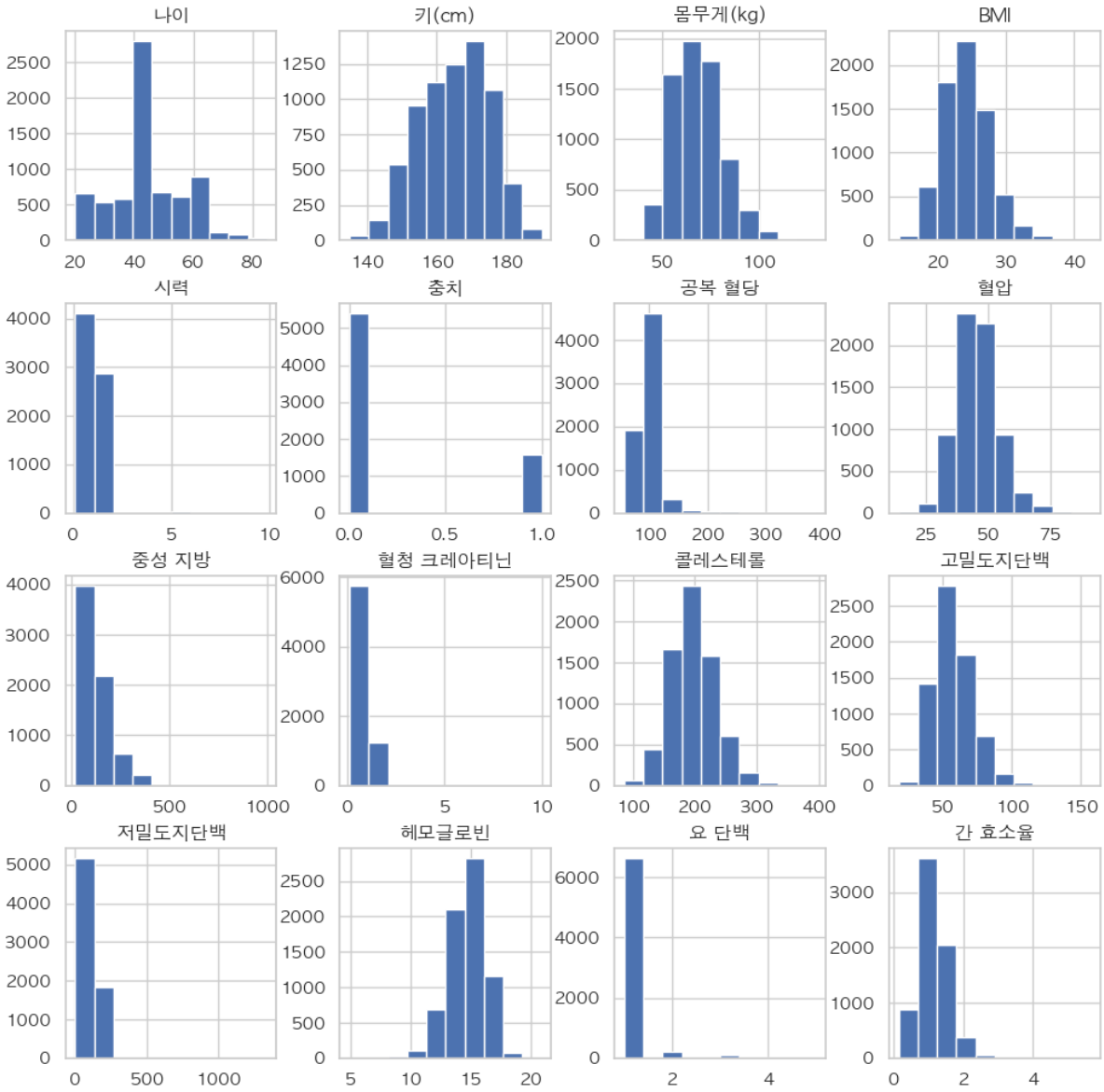
label(target) 피처의 값은 [0 : 비흡연, 1 : 흡연] 으로 인코딩되어 있습니다.
비흡연자가 흡연자보다 약 두 배 가량 많은 것을 확인할 수 있습니다.

전체 데이터 분포 확인

위에서 확인한 smoking(target) 변수를 제외하고 각 피처의 분포를 확인하겠습니다.

In [171…

```
except_target = train.drop('label', axis = 1)
except_target.hist(figsize = (12,12))
plt.show()
```

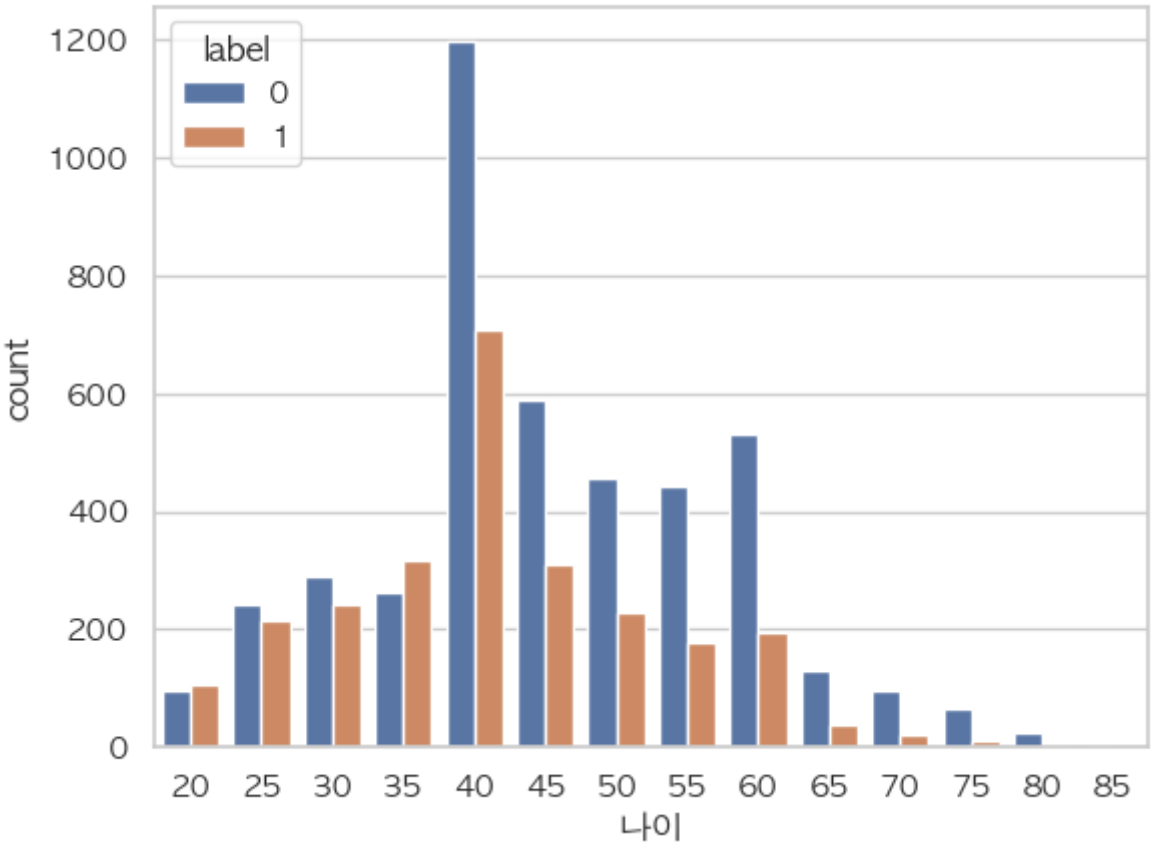


dental caries(충치 개수)는 0 또는 1의 값만을 가집니다.
하지만 ALT(알라닌 아미노 전이효소), Gtp(글루타밀 전이효소) 등은 아주 큰 값을 갖는 데이터가 일부 존재하기 때문에 x축이 길어진 것을 볼 수 있습니다.

연령별 흡연율 확인

In [172…

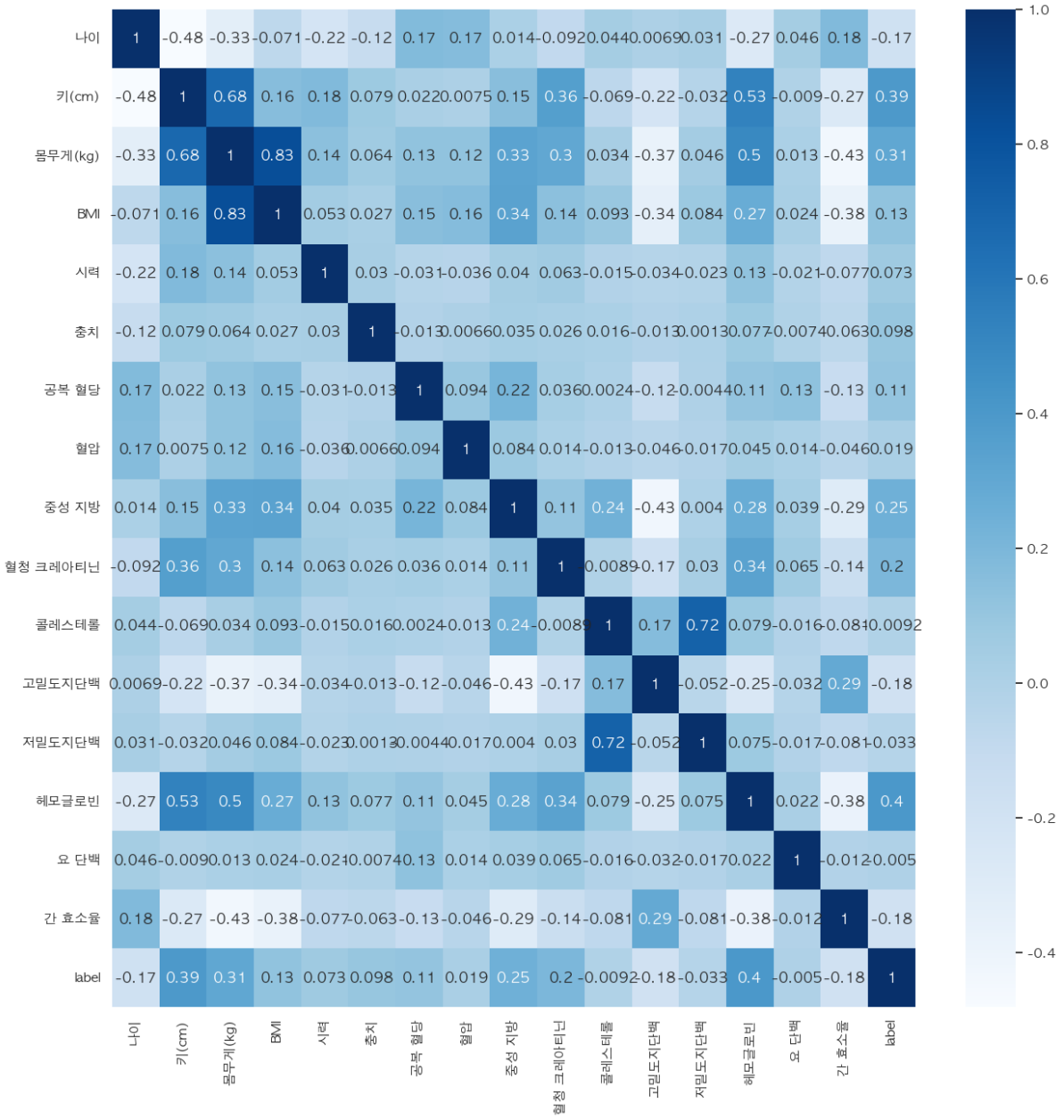
```
sns.countplot(x = '나이', hue = 'label', data = train)
plt.show()
```



상관관계 확인

In [173…

```
plt.figure(figsize = (15,15), dpi = 100)
sns.heatmap(train.corr(), annot = True, cmap = 'Blues')
plt.show()
```



데이터 전처리

이상치 처리

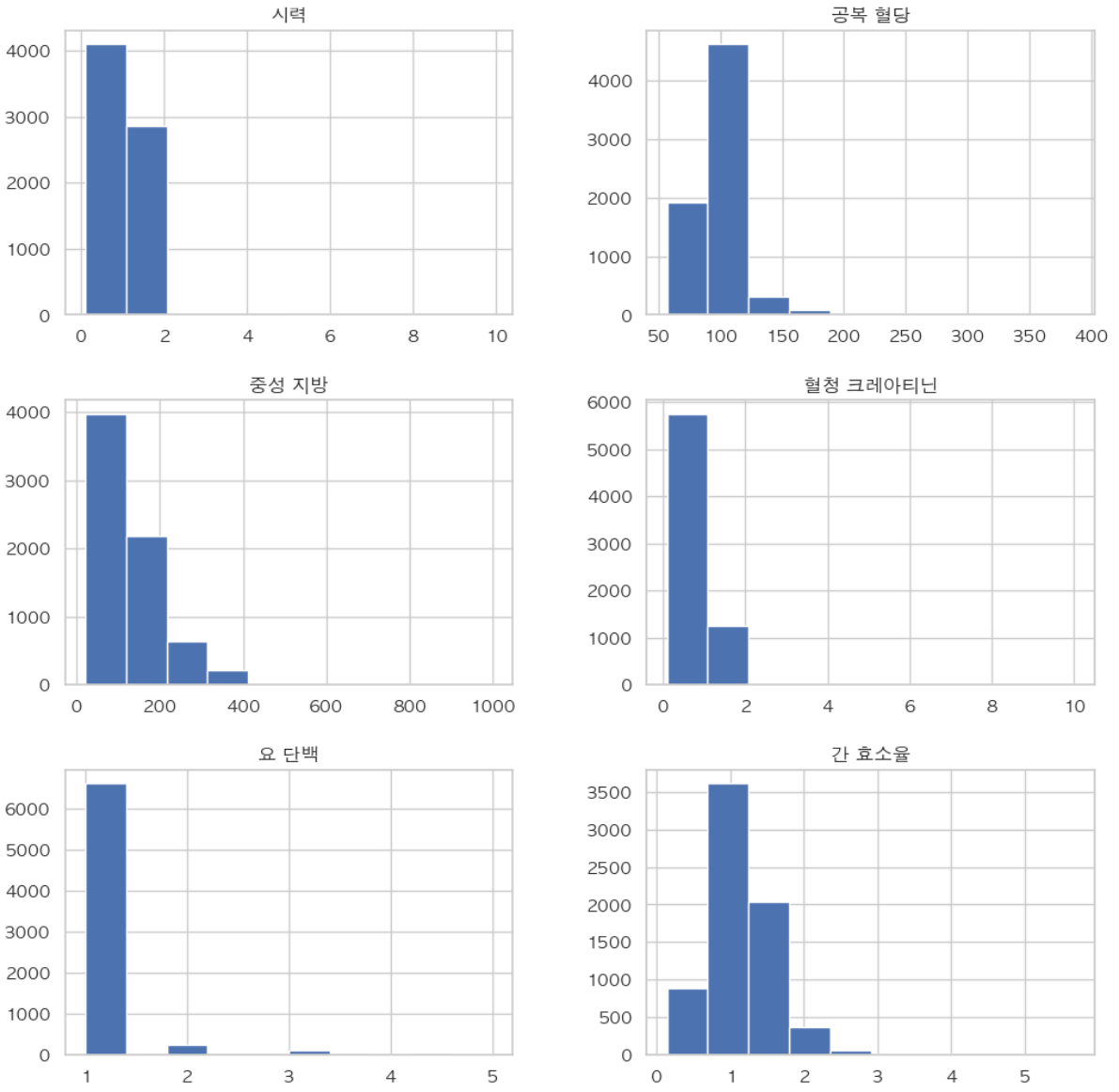
시력, 공복 혈당, 중성 지방, 혈청 크레아티닌, 요단백, 간 효소율

In [174…
```
train.columns
```

Out[174]:
```
Index(['ID', '나이', '키(cm)', '몸무게(kg)', 'BMI', '시력', '충치', '공복 혈당',
'혈압',
       '중성 지방', '혈청 크레아티닌', '콜레스테롤', '고밀도지단백', '저밀도지단백', '헤모글
로빈', '요 단백',
       '간 효소율', 'label'],
      dtype='object')
```

In [175…
```
train[['시력', '공복 혈당', '중성 지방', '혈청 크레아티닌', '요 단백',
'간 효소율']].hist(figsize=(12,12))
```

```
plt.show()
```

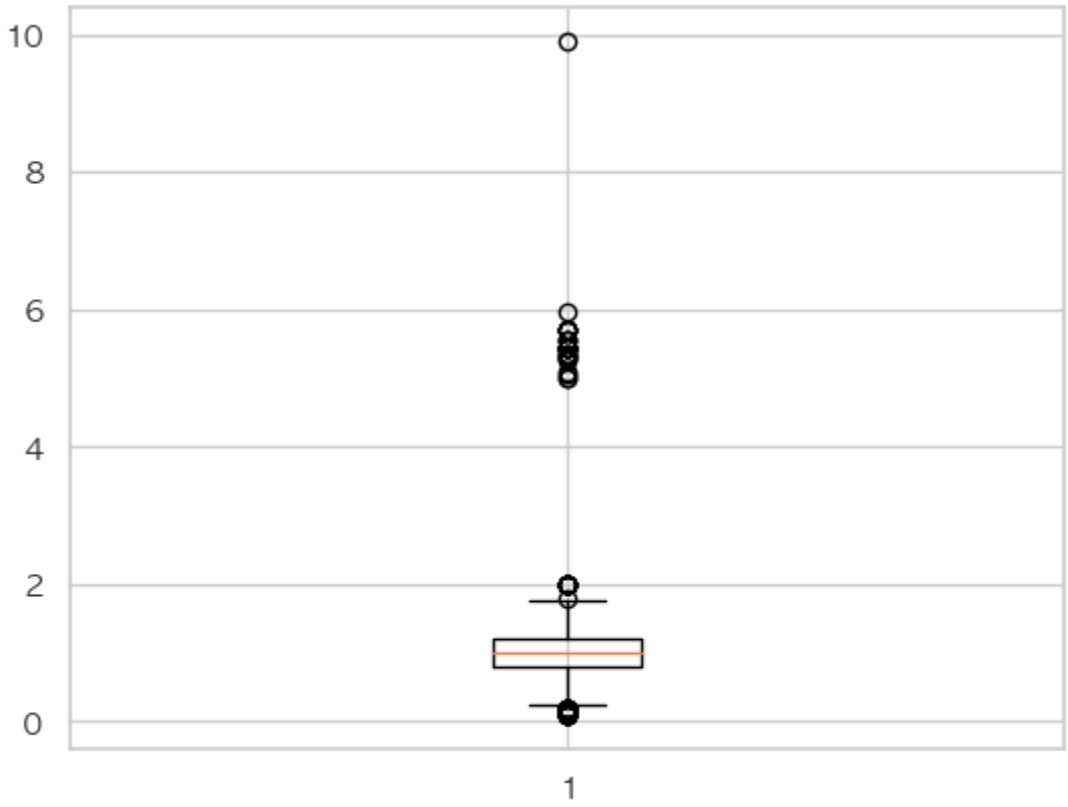


In [176…

```
plt.boxplot(train['시력'])
```

Out[176]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7fb9a1f9a1c0>,
  <matplotlib.lines.Line2D at 0x7fb9a1f9a490>],
 'caps': [<matplotlib.lines.Line2D at 0x7fb9a1f9a760>,
  <matplotlib.lines.Line2D at 0x7fb9a1f9aa30>],
 'boxes': [<matplotlib.lines.Line2D at 0x7fb9a1f8dfd0>],
 'medians': [<matplotlib.lines.Line2D at 0x7fb9a1f9ad00>],
 'fliers': [<matplotlib.lines.Line2D at 0x7fb9a1f9afd0>],
 'means': []}
```



In [179…
```
###
```

로그 변환

- 로그 변환은 산점도를 그려 보고 해야 한다. 그냥 한다고 해서 모델 성능이 무조건 좋아지지는 않는다.
- 해야 하면 train data에만 적용할 것.

In [178…
```
###
```

구간화

BMI < 18.5: 저체중 18.5 <= BMI < 24.9: 정상체중 25 <= BMI < 29.9: 과체중 30 <= BMI: 비만

In [ ]:
```
# from sklearn.preprocessing import OneHotEncoder

# 학습 데이터로 OneHotEncoder 학습
# encoder = OneHotEncoder()
# encoder.fit(X_train)  # X_train은 학습 데이터의 피쳐들로 이루어진 데이터프레임

# 학습된 인코더를 학습 데이터와 테스트 데이터에 적용
# X_train_encoded = encoder.transform(X_train)
```

```
# X_test_encoded = encoder.transform(X_test)  # X_test는 테스트 데이터의 피쳐들로 이루어진 데이터프레임
```

컬럼 생성

In [181…

```
###
```

컬럼 삭제

In [182…

```
train.drop(['ID'], axis=1, inplace=True)
test.drop(['ID'], axis=1, inplace=True)
```

In [183…

```
train.head(3)
```

Out[183]:

	나이	키 (cm)	몸무게 (kg)	BMI	시력	충치	공복혈당	혈압	중성지방	혈청크레아티닌	콜레스테롤	고밀도지단백	저밀도지단백	헤모글로빈	요단백	간효소율	label
0	35	170	70	24.22	1.10	1	98	40	80	1.3	211	75	120	15.9	1	1.53	1
1	40	150	55	24.44	1.00	0	173	39	104	0.6	251	46	184	11.8	1	1.45	0
2	60	170	50	17.30	0.75	0	96	40	61	0.8	144	43	89	15.3	1	1.04	0

스케일링 (정규화)

1. 표준화(Standardization):

- 상황: 표준화는 데이터의 분포가 정규 분포를 따르는 것을 가정할 때 주로 사용됩니다.
- 메커니즘: 표준화는 평균을 0, 표준편차를 1로 만들어 데이터를 표준 정규 분포로 변환합니다.
- 사용 사례: 일반적으로 모델들이 평균이 0이고 표준편차가 1인 데이터에 잘 동작하므로, 선형 회귀, 로지스틱 회귀, 신경망 등 다양한 모델에서 사용됩니다. SVM과 같은 일부 모델은 표준화된 데이터를 선호합니다.

1. 정규화(Normalization):

- 상황: 피쳐들의 범위가 다른 상황에서 사용됩니다. 예를 들어 피쳐 간의 단위가 다를 때 사용됩니다.
- 메커니즘: 정규화는 데이터를 최소값과 최대값 사이의 범위로 변환하여 0에서 1 사이의 값을 가지도록 만듭니다.
- 사용 사례: 정규화는 k-최근접 이웃 분류, 신경망에서의 이미지 처리와 같이 피쳐의 스케일이 다를 때 주로 사용됩니다. 최소-최대 정규화는 SVM과 같은 일부 모델에서도 사용될 수 있습니다.

In [129…

```
# Normalization
```

In [ ]:
```
# Min-max scaling
```

PCA

In [ ]:
```
###
```

데이터 분할

In [121…
```
# 데이터를 학습용과 검증용 나눕니다.
train_data, valid_data = train_test_split(train,
test_size=0.2, random_state=42)
```

모델 학습

In [122…
```
random_seed = 42
```

In [123…
```
# AutoGluon을 활용하여 모델 학습
predictor = TabularPredictor(label='label').fit(train_data,
presets='best_quality')
```

```
No path specified. Models will be saved in: "AutogluonModels/ag-20230817_140
254/"
Presets specified: ['best_quality']
Stack configuration (auto_stack=True): num_stack_levels=0, num_bag_folds=8,
num_bag_sets=1
Beginning AutoGluon training ...
AutoGluon will save models to "AutogluonModels/ag-20230817_140254/"
AutoGluon Version:  0.8.2
Python Version:     3.9.13
Operating System:   Darwin
Platform Machine:   x86_64
Platform Version:   Darwin Kernel Version 22.5.0: Thu Jun  8 22:22:19 PDT 20
23; root:xnu-8796.121.3~7/RELEASE_ARM64_T8103
Disk Space Avail:   211.10 GB / 494.38 GB (42.7%)
Train Data Rows:    5600
Train Data Columns: 16
Label Column: label
Preprocessing data ...
AutoGluon infers your prediction problem is: 'binary' (because only two uniq
ue label-values observed).
        2 unique label values:  [1, 0]
        If 'binary' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_ty
pe as one of: ['binary', 'multiclass', 'regression'])
Selected class <--> label mapping:  class 1 = 1, class 0 = 0
Using Feature Generators to preprocess the data ...
Fitting AutoMLPipelineFeatureGenerator...
        Available Memory:                    840.9 MB
        Train Data (Original)  Memory Usage: 0.72 MB (0.1% of available memo
ry)
        Inferring data type of each feature based on column values. Set feat
ure_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator...
                        Note: Converting 1 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator...
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator...
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator...
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator...
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) :  5 | ['BMI', '시력', '혈청 크레아티닌', '헤모글로
빈', '간 효소율']
                ('int', [])   : 11 | ['나이', '키(cm)', '몸무게(kg)', '충치', '공
복 혈당', ...]
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     :  5 | ['BMI', '시력', '혈청 크레아티닌', '헤모
글로빈', '간 효소율']
                ('int', [])       : 10 | ['나이', '키(cm)', '몸무게(kg)', '공복
혈당', '혈압', ...]
                ('int', ['bool']) :  1 | ['충치']
        0.0s = Fit runtime
```

```
        16 features in original data used to generate 16 features in process
ed data.
        Train Data (Processed) Memory Usage: 0.68 MB (0.1% of available memo
ry)
Data preprocessing and feature engineering runtime = 0.05s ...
AutoGluon will gauge predictive performance using evaluation metric: 'accura
cy'
        To change this, specify the eval_metric parameter of Predictor()
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}}, {},
'GBMLarge'],
        'CAT': {},
        'XGB': {},
        'FASTAI': {},
        'RF': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini', 'pro
blem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}}, {'crite
rion': 'squared_error', 'ag_args': {'name_suffix': 'MSE', 'problem_types':
['regression', 'quantile']}}],
        'XT': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini', 'pro
blem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}}, {'crite
rion': 'squared_error', 'ag_args': {'name_suffix': 'MSE', 'problem_types':
['regression', 'quantile']}}],
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
Fitting 13 L1 models ...
Fitting model: KNeighborsUnif_BAG_L1 ...
        0.663    = Validation score   (accuracy)
        0.01s    = Training   runtime
        0.06s    = Validation runtime
Fitting model: KNeighborsDist_BAG_L1 ...
        0.6741   = Validation score   (accuracy)
        0.01s    = Training   runtime
        0.07s    = Validation runtime
Fitting model: LightGBMXT_BAG_L1 ...
        Fitting 8 child models (S1F1 - S1F8) | Fitting with SequentialLocalF
oldFittingStrategy
        Warning: Exception caused LightGBMXT_BAG_L1 to fail during training
(ImportError)... Skipping this model.
                cannot import name 'log_evaluation' from 'lightgbm.callback'
(/Users/toypanda/opt/anaconda3/lib/python3.9/site-packages/lightgbm/callbac
k.py)
Fitting model: LightGBM_BAG_L1 ...
        Fitting 8 child models (S1F1 - S1F8) | Fitting with SequentialLocalF
oldFittingStrategy
        Warning: Exception caused LightGBM_BAG_L1 to fail during training (I
mportError)... Skipping this model.
                cannot import name 'log_evaluation' from 'lightgbm.callback'
(/Users/toypanda/opt/anaconda3/lib/python3.9/site-packages/lightgbm/callbac
k.py)
Fitting model: RandomForestGini_BAG_L1 ...
        0.728    = Validation score   (accuracy)
```

```
        0.78s    = Training   runtime
        0.2s     = Validation runtime
Fitting model: RandomForestEntr_BAG_L1 ...
        0.7327   = Validation score   (accuracy)
        0.82s    = Training   runtime
        0.18s    = Validation runtime
Fitting model: CatBoost_BAG_L1 ...
        Fitting 8 child models (S1F1 - S1F8) | Fitting with SequentialLocalF
oldFittingStrategy
        0.7446   = Validation score   (accuracy)
        11.21s   = Training   runtime
        0.01s    = Validation runtime
Fitting model: ExtraTreesGini_BAG_L1 ...
        0.7305   = Validation score   (accuracy)
        0.43s    = Training   runtime
        0.19s    = Validation runtime
Fitting model: ExtraTreesEntr_BAG_L1 ...
        0.7216   = Validation score   (accuracy)
        0.46s    = Training   runtime
        0.19s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 ...
        Fitting 8 child models (S1F1 - S1F8) | Fitting with SequentialLocalF
oldFittingStrategy
        Warning: Exception caused NeuralNetFastAI_BAG_L1 to fail during trai
ning (ImportError)... Skipping this model.
                Import fastai failed. A quick tip is to install via `pip ins
tall autogluon.tabular[fastai]==0.8.2`.
Fitting model: XGBoost_BAG_L1 ...
        Fitting 8 child models (S1F1 - S1F8) | Fitting with SequentialLocalF
oldFittingStrategy
        0.7371   = Validation score   (accuracy)
        4.99s    = Training   runtime
        0.03s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 ...
        Fitting 8 child models (S1F1 - S1F8) | Fitting with SequentialLocalF
oldFittingStrategy
        Warning: Exception caused NeuralNetTorch_BAG_L1 to fail during train
ing (ImportError)... Skipping this model.
                Unable to import dependency torch
A quick tip is to install via `pip install torch`.
The minimum torch version is currently 1.6.
Fitting model: LightGBMLarge_BAG_L1 ...
        Fitting 8 child models (S1F1 - S1F8) | Fitting with SequentialLocalF
oldFittingStrategy
        Warning: Exception caused LightGBMLarge_BAG_L1 to fail during traini
ng (ImportError)... Skipping this model.
                cannot import name 'log_evaluation' from 'lightgbm.callback'
(/Users/toypanda/opt/anaconda3/lib/python3.9/site-packages/lightgbm/callbac
k.py)
Fitting model: WeightedEnsemble_L2 ...
        0.7477   = Validation score   (accuracy)
        1.1s     = Training   runtime
        0.01s    = Validation runtime
AutoGluon training complete, total runtime = 21.43s ... Best model: "Weighte
dEnsemble_L2"
```

```
TabularPredictor saved. To load, use: predictor = TabularPredictor.load("Aut
ogluonModels/ag-20230817_140254/")
```

성능 평가

In [124…

```
# 검증 데이터로 성능 평가
results = predictor.evaluate(valid_data)
print(results)
```

```
Evaluation: accuracy on test data: 0.7257142857142858
Evaluations on test data:
{
    "accuracy": 0.7257142857142858,
    "balanced_accuracy": 0.7150881415183002,
    "mcc": 0.422768082751542,
    "roc_auc": 0.7991497659221263,
    "f1": 0.6437847866419296,
    "precision": 0.6152482269503546,
    "recall": 0.6750972762645915
}
{'accuracy': 0.7257142857142858, 'balanced_accuracy': 0.7150881415183002, 'm
cc': 0.422768082751542, 'roc_auc': 0.7991497659221263, 'f1': 0.6437847866419
296, 'precision': 0.6152482269503546, 'recall': 0.6750972762645915}
```

In [125…

```
# 리더보드 확인
leaderboard = predictor.leaderboard(valid_data)
leaderboard
```

```
                  model  score_test  score_val  pred_time_test  pred_time
_val   fit_time  pred_time_test_marginal  pred_time_val_marginal  fit_time_m
arginal  stack_level  can_infer  fit_order
0      WeightedEnsemble_L2    0.725714   0.747679        0.227498       0.58
5905  14.346524                0.002507                0.006913
1.103948            2       True          9
1  RandomForestEntr_BAG_L1    0.720714   0.732679        0.067739       0.17
9828   0.815762                0.067739                0.179828
0.815762            1       True          4
2          CatBoost_BAG_L1    0.720000   0.744643        0.011952       0.01
1556  11.214865                0.011952                0.011556          1
1.214865            1       True          5
3  RandomForestGini_BAG_L1    0.718571   0.728036        0.070960       0.19
8037   0.778918                0.070960                0.198037
0.778918            1       True          3
4    ExtraTreesGini_BAG_L1    0.716429   0.730536        0.074340       0.18
9571   0.433031                0.074340                0.189571
0.433031            1       True          6
5    ExtraTreesEntr_BAG_L1    0.714286   0.721607        0.081979       0.18
9625   0.462652                0.081979                0.189625
0.462652            1       True          7
6           XGBoost_BAG_L1    0.711429   0.737143        0.093405       0.02
5645   4.992033                0.093405                0.025645
4.992033            1       True          8
7    KNeighborsDist_BAG_L1    0.678571   0.674107        0.031866       0.06
8873   0.007776                0.031866                0.068873
0.007776            1       True          2
8    KNeighborsUnif_BAG_L1    0.666429   0.663036        0.033424       0.06
3336   0.011535                0.033424                0.063336
0.011535            1       True          1
```

Out[125]:

	model	score_test	score_val	pred_time_test	pred_time_val	fit_tir
0	WeightedEnsemble_L2	0.725714	0.747679	0.227498	0.585905	14.3465
1	RandomForestEntr_BAG_L1	0.720714	0.732679	0.067739	0.179828	0.8157
2	CatBoost_BAG_L1	0.720000	0.744643	0.011952	0.011556	11.2148
3	RandomForestGini_BAG_L1	0.718571	0.728036	0.070960	0.198037	0.7789
4	ExtraTreesGini_BAG_L1	0.716429	0.730536	0.074340	0.189571	0.4330
5	ExtraTreesEntr_BAG_L1	0.714286	0.721607	0.081979	0.189625	0.4626
6	XGBoost_BAG_L1	0.711429	0.737143	0.093405	0.025645	4.9920
7	KNeighborsDist_BAG_L1	0.678571	0.674107	0.031866	0.068873	0.0077
8	KNeighborsUnif_BAG_L1	0.666429	0.663036	0.033424	0.063336	0.0115

In [126…

```
# 피처 중요도 확인
feature_importance =
predictor.feature_importance(valid_data)
print(feature_importance)
```

```
Computing feature importance via permutation shuffling for 16 features using
1400 rows with 5 shuffle sets...
        24.57s  = Expected runtime (4.91s per shuffle set)
        3.83s   = Actual runtime (Completed 5 of 5 shuffle sets)
```

	importance	stddev	p_value	n	p99_high	p99_low
헤모글로빈	0.049857	0.005877	0.000023	5	0.061958	0.037756
키(cm)	0.039000	0.007433	0.000151	5	0.054305	0.023695
중성 지방	0.019000	0.006480	0.001399	5	0.032342	0.005658
나이	0.017429	0.003297	0.000147	5	0.024216	0.010641
충치	0.009429	0.004933	0.006456	5	0.019586	-0.000729
고밀도지단백	0.008143	0.007330	0.033955	5	0.023235	-0.006949
저밀도지단백	0.006857	0.003373	0.005226	5	0.013802	-0.000088
시력	0.006000	0.004860	0.025414	5	0.016007	-0.004007
공복 혈당	0.005429	0.004244	0.022958	5	0.014167	-0.003310
혈청 크레아티닌	0.001714	0.002505	0.100357	5	0.006872	-0.003444
BMI	0.001571	0.004386	0.233955	5	0.010602	-0.007459
혈압	0.001571	0.004268	0.228288	5	0.010359	-0.007216
콜레스테롤	0.001286	0.006215	0.333847	5	0.014082	-0.011510
간 효소율	0.001286	0.006877	0.348692	5	0.015446	-0.012875
몸무게(kg)	-0.000286	0.003373	0.570501	5	0.006659	-0.007231
요 단백	-0.000714	0.001429	0.836918	5	0.002227	-0.003656

예측

In [ ]:

```
# 테스트 데이터로 예측
predictions = predictor.predict(test)

print(predictions)
```

제출 파일 생성

In [50]:

```
# 제출 파일을 읽어옵니다.
submit = pd.read_csv('data/sample_submission.csv')

# 예측한 값을 TARGET 컬럼에 할당합니다.
submit['label'] = predictions
submit.head()
```

Out[50]:

	ID	label
0	TEST_0000	0
1	TEST_0001	0
2	TEST_0002	1
3	TEST_0003	1
4	TEST_0004	0

In [51]:
```
# 예측한 결과를 파일로 저장합니다. index 인자의 값을 False로 설정하지 않으면 제출이 정상적으로 진행되지 않습니다.
submit.to_csv('submission/submission_'+datetime.datetime.now(
index = False)
```

끝.