Library

[1]:
```
# Ignore the warnings
import warnings
# warnings.filterwarnings('always')
warnings.filterwarnings('ignore')

# System related and data input controls
import os

# Python path
import sys
base_folder = 'DataScience'
location_base = os.path.join(os.getcwd().split(base_folder)[0], base_folder)
location_module = [os.path.join(location_base, 'Module')]
for each in location_module:
    if each not in sys.path:
        sys.path.append(each)

# Auto reload of library
%reload_ext autoreload
%autoreload 2

from import_KK import *
DeviceStrategy_GPU()
from preprocessing_KK import *
from preprocessing_project_KK import *
from description_KK import *
from algorithm_machinelearning_KK import *
from algorithm_deeplearning_KK import *
from evaluation_KK import *
from visualization_KK import *
```

Last executed at 2025-06-03 14:08:00 in 22.38s

```
JAVA is in the system path?:  False
JAVA is in the system path?:  Adding...
Operation Machine:  x86_64
Operation Platform:  64bit
OS Type:  Linux
OS Version:  5.15.167.4-microsoft-standard-WSL2
Python Version:  3.12.2 | packaged by conda-forge | (main, Feb 16 2024, 20:50:58) [GCC 12.3.0]
[nltk_data] Downloading package stopwords to /home/kk/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /home/kk/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to /home/kk/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package tagsets_json to /home/kk/nltk_data...
[nltk_data]   Package tagsets_json is already up-to-date!
2025-06-03 14:07:47.463497: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical res
ults due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONED
NN_OPTS=0`.
2025-06-03 14:07:47.624829: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to register cuFFT factory: Attempting to re
gister factory for plugin cuFFT when one has already been registered
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1748927267.711331    1612 cuda_dnn.cc:8310] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN whe
n one has already been registered
E0000 00:00:1748927267.736573    1612 cuda_blas.cc:1418] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS
when one has already been registered
2025-06-03 14:07:47.897223: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU ins
tructions in performance-critical operations.
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
I0000 00:00:1748927279.487952    1612 gpu_device.cc:2022] Created device /job:localhost/replica:0/task:0/device:GPU:0 with 5520 MB memory:  -
> device: 0, name: NVIDIA GeForce RTX 4070 Laptop GPU, pci bus id: 0000:01:00.0, compute capability: 8.9
Cuda is Ready?  True
Cuda Version:  12.5.1
Cudnn Version:  9

Tensorflow Version:  2.18.0
Keras Version:  3.7.0
Torch Version:  2.5.0+cu124
Torch Cuda Version: 12.4
Torch Cudnn Version:90600

There are 1 GPU(s) available.
We will use the GPU: NVIDIA GeForce RTX 4070 Laptop GPU
```

Hyperparameters

[6]:
```
# Data
FOLDER_LOCATION = os.path.join('.', 'Data')
DF_YEAR = [2022, 2021, 2020, 2019, 2018]
Y_colname = '비형식교육 참여자'
```

```
X_custom = {
    '문C1) 앞으로 참여하길 희망하는 프로그램은 무엇입니까? (중복응답)1': {99: 0},
    '학력별': {4: 0},
    '문H1) 귀하께서는 전반적으로 현재의 생활에 얼마나 만족하십니까?': {99: 1},
    '직장의 규모': {9: 0},
    '근무기간': {9: 0}
}
X_dummy = [
    'DQ1. 귀하의 최종 학력은 어떻게 되십니까?__상태구분',
    'DQ6. 귀하께서는 건강에 자신 있습니까?',
    'DQ7. 귀하의 주된 소득원천은 무엇입니까?',
    'DQ9. 귀하의 현재 고용형태는 어떻게 되십니까?',
    'DQ10. 귀하는 현재 어떤 직업에 종사하고 계십니까?',
    '직업'
]
X_reverse = [
    '문A1-1) 작년(21.01.01~21.12.31) (1) 학위(졸업장) 취득을 위한 교육 참여 경험 여부',
    '문B1-5) 귀하께서는 작년에 다음과 같은 학습에 참여해 본 적이 있으십니까?__학습을 목적으로 텔레비전^ 라디오 등을 활용해서 새로운 지식을 습득한 적이 있다',
    '문B1-7) 귀하께서는 작년에 다음과 같은 학습에 참여해 본 적이 있으십니까?__역사적·자연적·산업적 장소를 방문해서 지식을 습득한 적이 있다',
    '문B1-1) 귀하께서는 작년에 다음과 같은 학습에 참여해 본 적이 있으십니까?__가족^ 친구 또는 직장동료^ 상사의 도움이나 조언을 통해 지식을 습득한 적이 있다',
    '문B1-2) 귀하께서는 작년에 다음과 같은 학습에 참여해 본 적이 있으십니까?__트위터^ 페이스북^ 카페^ 블로그^ 밴드 등을 활용해서 새로운 정보나 기술을 습득한 적이 있다',
    '문B1-3) 귀하께서는 작년에 다음과 같은 학습에 참여해 본 적이 있으십니까?__유튜브(Youtube) 등을 활용해서 새로운 정보나 기술을 습득한 적이 있다',
    '문B1-4) 귀하께서는 작년에 다음과 같은 학습에 참여해 본 적이 있으십니까?__인터넷 뉴스^ E-book 등 온라인매체를 활용해서 새로운 정보나 기술을 습득한 적이 있다',
    '문B1-6) 귀하께서는 작년에 다음과 같은 학습에 참여해 본 적이 있으십니까?__책이나 전문잡지 등 인쇄매체를 활용해서 지식을 습득한 적이 있다',
    '문B1-8) 귀하께서는 작년에 다음과 같은 학습에 참여해 본 적이 있으십니까?__도서관 등을 방문해서 새로운 사실을 배운 적이 있다',
    '문B1-9) 귀하께서는 작년에 다음과 같은 학습에 참여해 본 적이 있으십니까?__축제^ 박람회^ 음악회 등에 참여해서 무언가를 새롭게 배우거나 깊이 있게 알게된 적이 있다',
    '문B1-10) 귀하께서는 작년에 다음과 같은 학습에 참여해 본 적이 있으십니까?__스포츠^ 등산 등 신체를 움직이는 활동에 참여해서 무언가를 새롭게 배우거나 깊이 있게 알게 된 적이
    '문C2) 귀하께서는 작년(교육이나 학습 프로그램의 참여여부와 관계없이)에 참여하고 싶었지만 참여하지 못했던 평생교육 프로그램이 있었습니까?',
    '문D1) 귀하께서는 작년에 평생교육 프로그램 관련 정보를 접한 적이 있습니까?',
    'DQ8. 귀하께서는 현재 수입을 목적으로 일하고 계십니까?',
    '경활상태',
    '지역규모별',
    '건강에 대한 자신감',
    '취약계층',
    '취업구분',
    '고용형태',
    '평생학습 참여자',
    '형식교육 참여자',
    '비형식교육 참여자',
    '직업관련 비형식교육 참여자',
    '동시 참여자'
]
X_delete = [
    'ID', '원가중치', '평생학습 참여자', '직업관련 비형식교육 참여자', '동시 참여자',
    # DQ컬럼과 중복(by 조은지)
    '학력별', '자녀유무 및 막내자녀 상태', '동거가족 유무', '부양가족 유무', '취학 전의 손자/손녀 유무',
    '건강에 대한 자신감', '고용형태', '직업', '직장의 규모', '근무기간',
    '문A1-1) 작년(21.01.01~21.12.31) (1) 학위(졸업장) 취득을 위한 교육 참여 경험 여부'
]
TEST_SIZE = 0.2
RANDOM_STATE = 123
CLASS_STAT = True
SAMPLING_METHOD = 'auto'
SAMPLING_STRATEGY = 'auto'    # 'minority', 'not majority', 'not minority', 'all', 'auto'
SCALER = 'minmax'
LABEL_LIST = ['Formal Education', 'Non-formal Ecudation']

# Model
################
tf.random.set_seed(123)
NODE_MLP = [128, 256, 128, 64, 32, 10]
NODE_CNN1 = [128, 256, 128]
NODE_CNN2 = [64, 32, 10]
KERNEL_SIZE = 5
STRIDE = 1
PADDING = 'same'
POOL_SIZE = 2
POOL_STRIDE = 2
HIDDEN_ACTIVATION = 'relu'
OUTPUT_ACTIVATION = 'sigmoid'
REGULARIZER = None
LEARNING_RATE = 0.001
# REGULARIZER = regularizers.l2(LEARNING_RATE)
DROPOUT_RATIO = 0.25
WEIGHT_METHOD = 'sample'   # None, 'class', 'sample'
if WEIGHT_METHOD != None:
    LOSS = 'categorical_crossentropy'
else:
    LOSS = 'sparse_categorical_crossentropy'
METRICS = ['accuracy']    # 'accuracy', 'Precision', 'Recall', 'AUC', 'F1Score'
VALIDATION_SPLIT = 0.25
VALIDATION_DATA = None
BATCH_SIZE = 64
EPOCHS = 1000
VERBOSE = 0
################
SHAP = True
X_TOP_DISPLAY = 50
DEPENDENCY = True
################
MONITOR = 'val_accuracy'    # 'val_accuracy', 'val_precision', 'val_recall', 'val_f1'
MONITOR_MODE = 'max'
```

```
EARLYSTOP_PATIENT = int(EPOCHS*0.1)

# Save
```

Last executed at 2025-06-03 14:08:53 in 104ms

Data Merge

[3]:
```
# Data Loading
df_dict = dict()
df_columns = pd.read_excel(os.path.join(FOLDER_LOCATION, '평생교육실태조사_코드명통합_수정본.xlsx')).iloc[:,2:]
for year in DF_YEAR:
    folder_location=os.path.join(FOLDER_LOCATION, '{}_총괄_20250320_36011.csv'.format(year))
    df_dict[year] = pd.read_csv(folder_location, encoding='cp949')
    print(year, df_dict[year].shape)
    ## 변수명 DF_YEAR[0]기준으로 동일하게 변경
    if year != DF_YEAR[0]:
        df_sub = pd.concat([df_columns[[col for col in df_columns.columns if col.split('_')[1] == str(year)]], df_columns.iloc[:,[0]]], axis=1
        df_rename = dict(df_sub[df_sub.iloc[:,1] != 0].dropna().reset_index().iloc[:,1:].values)
        df_dict[year] = df_dict[year].rename(columns=df_rename)

# Feature Comparison
colname_compare, colname_common = table_compare_dfscolname(df_dict[DF_YEAR[0]], [df_dict[i] for i in DF_YEAR[1:]],
                                                           colname_list=DF_YEAR)

## Final Feature
for year in DF_YEAR:
    ## 사용할 변수명 추출
    colnames = list(colname_common[[i for i in colname_common.columns if i.split('_')[1] == str(year)]].values.flatten())

    ## 최종선택과 추가해야할 변수 구분
    colname_select, colname_add = [], []
    for idx, colname in enumerate(colnames):
        if colname != 0:
            colname_select.append(colname)
        else:
            colname_add.append([i for i in colname_common.iloc[idx,:].unique() if i != 0][0])

    ## 변수 업데이트
    df_dict[year] = df_dict[year][colname_select]
    for add in colname_add:
        df_dict[year][add] = np.nan

## df 생성
df = dict_to_concatdf(df_dict)
```

Last executed at 2025-06-03 14:08:09 in 9.09s

```
2022 (9968, 437)
2021 (9905, 454)
2020 (9776, 434)
2019 (9973, 511)
2018 (11747, 600)
4it [00:02,  1.39it/s]
100%|| 5/5 [00:00<00:00, 26.70it/s]
Final Shape of df:  (51369, 368)
```

Preprocessing

[4]:
```
X_train, X_test, X_colname, \
Y_train, Y_test = preprocessing_edu(df, Y_colname=Y_colname,
                                    X_custom=X_custom, X_dummy=X_dummy, X_reverse=X_reverse, X_delete=X_delete,
                                    test_size=TEST_SIZE, class_stat=CLASS_STAT,
                                    sampling_method=SAMPLING_METHOD, sampling_strategy=SAMPLING_STRATEGY,
                                    scaler=SCALER, random_state=RANDOM_STATE,
                                    label_list=LABEL_LIST)
```

Last executed at 2025-06-03 14:08:13 in 3.59s

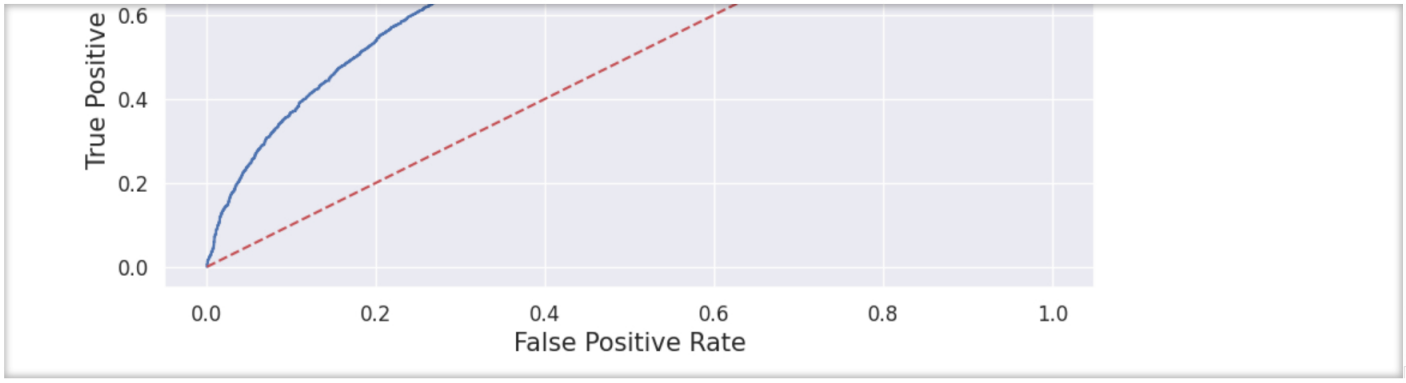
```
Preprocessing for Data Prepare...
Initial Shape:  (51369, 368)

Deleting Non-meaning Variables...
Shape:  (51369, 352)

Preprocessing NAN...
Column Deleting...
These  82.67045454545455 %( 291 ) columns are containing more than  50.0 % null values!: ['문A2-1) 프로그램 유형__(1)학위취득을 위한 교육_1', '문A2-6) 연간참여시간__(1)학위취득을 위한 교육_1', '문A2-7) 자기부담학습비__(1)학위취득을 위한 교육_1', '문A1-2) 참여한 프로그램 수는 몇개입니까? (2) 성인기초 및 문자해득교육(문해교육)', '문A1-2) 참여한 프로그램 수는 몇개입니까? (3) 직업능력향상교육', '문A1-2) 참여한 프로그램 수는 몇개입니까? (5) 문화예술스포츠교육', '문A1-2) 참여한 프로그램 수는 몇개입니까? (6) 시민참여교육', '문A1-2) 참여한 프로그램 수는 몇개입니까? (4) 인문교양교육', '문A2-5) 참여목적__(1)학위취득을 위한 교육_1', '문A2-1) 프로그램 유형__(2)성인기초 및 문자해득교육_1', '문A2-2) 프로그램 형태__(2)성인기초 및 문자해득교육_1', '문A2-3) 기관 유형__(2)성인기초 및 문자해득교육_1', '문A2-5) 참여목적__(2)성인기초 및 문자해득교육_1', '문A2-6) 연간참여시간__(2)성인기초 및 문자해득교육_1', '문A2-7) 자기부담학습비__(2)성인기초 및 문자해득교육_1', '문A2-8) 학습비 외부 지원여부__(2)성인기초 및 문자해득교육_1', '문A2-9) 학습비 외부 지원기관__(2)성인기초 및 문자해득교육_1', '문A2-10) 프로그램만족도__(2)성인기초 및 문자해득교육_1', '문A2-11) 프로그램 불만족 요인(중복응답)__(2)성인기초 및 문자해득교육_1-1', '문A2-11) 프로그램 불만족 요인(중복응답)__(2)성인기초 및 문자해득교육_1-2', '문A2-11) 프로그램 불만족 요인(중복응답)__(2)성인기초 및 문자해득교육_1-3', '문A2-11) 프로그램 불만족 요인(중복응답)__(2)성인기초 및 문자해득교육_1-5', '문A2-11) 프로그램 불만족 요인
```

Modeling

Machine Learing

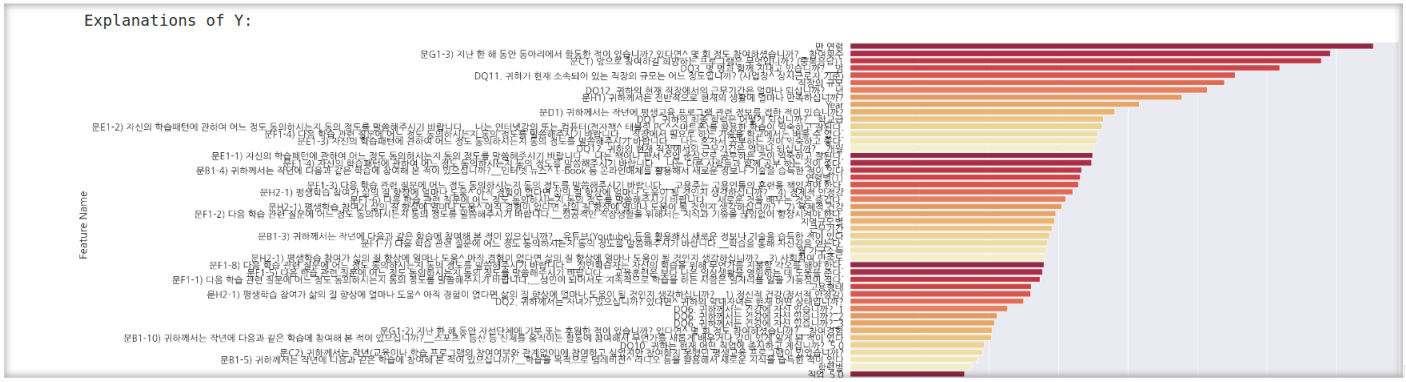
[37]:
```
model_lr, Score_te_lr, Score_trte_lr = modeling_LogisticRegression(X_train, Y_train, X_test, Y_test,
                                                                   X_colname=X_colname,
                                                                   shap=not SHAP,
                                                                   X_top_display=X_TOP_DISPLAY,
                                                                   dependency=DEPENDENCY,
                                                                   label_list=LABEL_LIST)
display(model_lr, Score_te_lr, Score_trte_lr)
```

Last executed at 2025-05-28 20:55:09 in 1.28s



[38]:
```
model_rf, Score_te_rf, Score_trte_rf = modeling_RandomForestClassifier(X_train, Y_train,
                                                                       X_test, Y_test,
                                                                       X_colname=X_colname,
                                                                       shap=not SHAP,
                                                                       X_top_display=X_TOP_DISPLAY,
                                                                       dependency=DEPENDENCY,
                                                                       label_list=LABEL_LIST)
display(model_rf, Score_te_rf, Score_trte_rf)
```

Last executed at 2025-05-28 20:55:16 in 7.17s



[39]:
```
model_xgb, Score_te_xgb, Score_trte_xgb = modeling_XGBClassifier(X_train, Y_train,
                                                                 X_test, Y_test,
                                                                 X_colname=X_colname,
                                                                 shap=not SHAP,
                                                                 X_top_display=X_TOP_DISPLAY,
                                                                 dependency=DEPENDENCY,
                                                                 label_list=LABEL_LIST)
display(model_xgb, Score_te_xgb, Score_trte_xgb)
```

Last executed at 2025-05-28 20:55:17 in 1.12s

[40]:
```
model_lgbm, Score_te_lgbm, Score_trte_lgbm = modeling_LGBMClassifier(X_train, Y_train,
                                                                     X_test, Y_test,
                                                                     X_colname=X_colname,
                                                                     shap=not SHAP,
                                                                     X_top_display=X_TOP_DISPLAY,
                                                                     dependency=DEPENDENCY,
                                                                     label_list=LABEL_LIST)
```

```
display(model_lgbm, Score_te_lgbm, Score_trte_lgbm)
```

Last executed at 2025-05-28 20:55:19 in 1.17s

```
[LightGBM] [Info] Number of positive: 26629, number of negative: 27589
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.004130 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 3386
[LightGBM] [Info] Number of data points in the train set: 54218, number of used features: 111
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500000 -> initscore=0.000000
[LightGBM] [Info] Start training from score 0.000000
Explanations of Y:
```

[41]:
```
model_catb, Score_te_catb, Score_trte_catb = modeling_CatBoostClassifier(X_train, Y_train,
                                                                         X_test, Y_test,
                                                                         X_colname=X_colname,
                                                                         shap=not SHAP,
                                                                         X_top_display=X_TOP_DISPLAY,
                                                                         dependency=DEPENDENCY,
                                                                         label_list=LABEL_LIST)
display(model_catb, Score_te_catb, Score_trte_catb)
```

Last executed at 2025-05-28 20:55:20 in 1.33s

```
Learning rate set to 0.46818
0:      learn: 0.6291833        total: 12.7ms   remaining: 1.25s
1:      learn: 0.5943113        total: 20.9ms   remaining: 1.02s
2:      learn: 0.5598701        total: 28.2ms   remaining: 912ms
3:      learn: 0.5119235        total: 33.1ms   remaining: 795ms
4:      learn: 0.5041867        total: 38ms     remaining: 722ms
5:      learn: 0.5000818        total: 43ms     remaining: 674ms
6:      learn: 0.4912305        total: 47.5ms   remaining: 631ms
7:      learn: 0.4844208        total: 52.3ms   remaining: 601ms
8:      learn: 0.4816652        total: 56.3ms   remaining: 569ms
9:      learn: 0.4692627        total: 60.3ms   remaining: 543ms
10:     learn: 0.4667274        total: 64.2ms   remaining: 520ms
11:     learn: 0.4627682        total: 67.9ms   remaining: 498ms
12:     learn: 0.4612322        total: 72.2ms   remaining: 483ms
13:     learn: 0.4504749        total: 76.2ms   remaining: 468ms
14:     learn: 0.4486805        total: 80.7ms   remaining: 457ms
15:     learn: 0.4466609        total: 85.1ms   remaining: 447ms
16:     learn: 0.4453188        total: 89.6ms   remaining: 438ms
```

Deep Learning

MLP

[7]:
```
# reshape
if WEIGHT_METHOD != None:
    Y_train_dl, Y_test_dl = reshape_YtoOneHot(Y_train, Y_test)
else:
    Y_train_dl, Y_test_dl = Y_train.copy(), Y_test.copy()
X_train_dl, X_test_dl = X_train.copy(), X_test.copy()

# 모델링
## 모델 구축
ALGO_NAME='MLP'
model = modeling_MLP(X_train_dl, Y_train_dl,
                     node_MLP=NODE_MLP,
                     HIDDEN_ACTIVATION=HIDDEN_ACTIVATION, OUTPUT_ACTIVATION=OUTPUT_ACTIVATION,
                     REGULARIZER=REGULARIZER, DROPOUT_RATIO=DROPOUT_RATIO, LOSS=LOSS)
## 검증데이터 설정
if VALIDATION_SPLIT == None:
    VALIDATION_DATA = (X_test_dl, Y_test_dl)
else:
    VALIDATION_DATA = None
## 데이터 학습
model, FILENAME = learning(model, X_train_dl, X_test_dl, Y_train_dl,
                           WEIGHT_METHOD=WEIGHT_METHOD,
                           VALIDATION_SPLIT=VALIDATION_SPLIT, VALIDATION_DATA=VALIDATION_DATA,
                           BATCH_SIZE=BATCH_SIZE, EPOCHS=EPOCHS, VERBOSE=VERBOSE,
                           MONITOR=MONITOR, MONITOR_MODE=MONITOR_MODE, EARLYSTOP_PATIENT=EARLYSTOP_PATIENT,
                           shap=not SHAP, X_colname=X_colname, X_top_display=X_TOP_DISPLAY)
Score_te_mlp, Score_trte_mlp = prediction_class(model, X_train_dl, Y_train_dl, X_test_dl, Y_test_dl,
                                                LABEL_LIST=LABEL_LIST, ALGO_NAME=ALGO_NAME)
display(Score_te_mlp, Score_trte_mlp)

# 베스트 모델 로딩 및 예측
model_mlp = load_model(FILENAME)
Score_te_mlp, Score_trte_mlp = prediction_class(model, X_train_dl, Y_train_dl, X_test_dl, Y_test_dl,
                                                LABEL_LIST=LABEL_LIST, ALGO_NAME=ALGO_NAME)
display(model_mlp, Score_te_mlp, Score_trte_mlp)
```

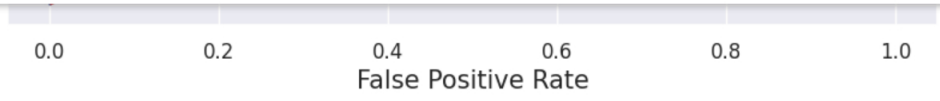
Last executed at 2025-06-03 14:44:27 in 35m 21.58s

input_layer (InputLayer)	(None, 87)	0
dense (Dense)	(None, 128)	11,264
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 256)	33,024
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32,896
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 64)	8,256
dropout_3 (Dropout)	(None, 64)	0

CNN

[8]:
```
# Reshape
if WEIGHT_METHOD != None:
    Y_train_dl, Y_test_dl = reshape_YtoOneHot(Y_train, Y_test)
else:
    Y_train_dl, Y_test_dl = Y_train.copy(), Y_test.copy()
X_train_dl, X_test_dl = reshape_X2Dto3D(X_train, X_test)

# 모델링
## 모델 구축
ALGO_NAME='CNN'
model = modeling_CNN1D(X_train_dl, Y_train_dl,
                       node_CNN1=NODE_CNN1,
                       node_CNN2=NODE_CNN2,
                       HIDDEN_ACTIVATION=HIDDEN_ACTIVATION, OUTPUT_ACTIVATION=OUTPUT_ACTIVATION,
                       KERNEL_SIZE=KERNEL_SIZE, STRIDE=STRIDE, PADDING=PADDING,
                       POOL_SIZE=POOL_SIZE, POOL_STRIDE=POOL_STRIDE,
                       REGULARIZER=REGULARIZER, DROPOUT_RATIO=DROPOUT_RATIO, LOSS=LOSS)
## 검증데이터 설정
if VALIDATION_SPLIT == None:
    VALIDATION_DATA = (X_test_dl, Y_test_dl)
else:
    VALIDATION_DATA = None
## 데이터 학습
model, FILENAME = learning(model, X_train_dl, X_test_dl, Y_train_dl,
                           WEIGHT_METHOD=WEIGHT_METHOD,
                           VALIDATION_SPLIT=VALIDATION_SPLIT, VALIDATION_DATA=VALIDATION_DATA,
                           BATCH_SIZE=BATCH_SIZE, EPOCHS=EPOCHS, VERBOSE=VERBOSE,
                           MONITOR=MONITOR, MONITOR_MODE=MONITOR_MODE, EARLYSTOP_PATIENT=EARLYSTOP_PATIENT,
                           shap=not SHAP, X_colname=X_colname, X_top_display=X_TOP_DISPLAY)
Score_te_cnn, Score_trte_cnn = prediction_class(model, X_train_dl, Y_train_dl, X_test_dl, Y_test_dl,
                                                LABEL_LIST=LABEL_LIST, ALGO_NAME=ALGO_NAME)
display(Score_te_cnn, Score_trte_cnn)

# 베스트 모델 로딩 및 예측
model_cnn = load_model(FILENAME)
Score_te_cnn, Score_trte_cnn = prediction_class(model, X_train_dl, Y_train_dl, X_test_dl, Y_test_dl,
                                                LABEL_LIST=LABEL_LIST, ALGO_NAME=ALGO_NAME)
display(model_cnn, Score_te_cnn, Score_trte_cnn)
```

Last executed at 2025-06-03 15:21:35 in 37m 8.08s



<Functional name=functional_1, built=True>

	N	True Positive	True Negative	False Positive	False Negative	Precision	Recall	Specificity	F1-score	Accuracy	Balanced Accuracy	AUC
Test set	10274	6233	242	3409	390	0.6464	0.9411	0.0663	0.7664	0.6302	0.5037	0.5830

	N	True Positive	True Negative	False Positive	False Negative	Precision	Recall	Specificity	F1-score	Accuracy	Balanced Accuracy	AUC
Entire population	51369	32094	470	17647	1158	0.6452	0.9652	0.0259	0.7734	0.6339	0.4956	0.6183

Summary

[11]:
```
folder_location = os.path.join(os.getcwd(),'Result')
prediction_summary(folder_location=folder_location,
                   # algonames=['Logistic Regression', 'Random Forest', 'XGBoost', 'LGBM', 'CatBoost'])
                   algonames=['Logistic Regression', 'Random Forest', 'XGBoost', 'LGBM', 'CatBoost', 'MLP', 'CNN'])
```

Last executed at 2025-06-03 18:52:30 in 221ms

	Dataset	Algorithm	N	True Positive	True Negative	False Positive	False Negative	Precision	Recall	Specificity	F1-score	Accuracy	Balanced Accuracy	AUC
0	Test set	Logistic	10274	4503	2522	1129	2120	0.799538	0.679903	0.690770	0.734884	0.683765	0.685337	0.756177

		Regression												
1	Test set	Random Forest	10274	6023	1632	2019	600	0.748943	0.909407	0.447001	0.821412	0.745085	0.678204	0.799526
2	Test set	XGBoost	10274	4938	2537	1114	1685	0.815929	0.745584	0.694878	0.779172	0.727565	0.720231	0.796893
3	Test set	LGBM	10274	4690	2660	991	1933	0.825559	0.708138	0.728568	0.762354	0.715398	0.718353	0.796603
4	Test set	CatBoost	10274	4798	2593	1058	1825	0.819331	0.724445	0.710216	0.768972	0.719389	0.717331	0.792135
5	Test set	MLP	10274	2368	3241	410	4255	0.852412	0.357542	0.887702	0.503776	0.545941	0.622622	0.727351
6	Test set	CNN	10274	6233	242	3409	390	0.646443	0.941114	0.066283	0.766431	0.630232	0.503699	0.582958

	Dataset	Algorithm	N	True Positive	True Negative	False Positive	False Negative	Precision	Recall	Specificity	F1-score	Accuracy	Balanced Accuracy	AUC
0	Entire population	Logistic Regression	51369	22846	12483	5634	10406	0.802177	0.687056	0.689021	0.740167	0.687749	0.688039	0.756619
1	Entire population	Random Forest	51369	32652	16097	2020	600	0.941740	0.981956	0.888503	0.961427	0.948996	0.935229	0.988716
2	Entire population	XGBoost	51369	26816	14567	3550	6436	0.883093	0.806448	0.804051	0.843032	0.805603	0.805250	0.888322
3	Entire population	LGBM	51369	24444	13785	4332	8808	0.849458	0.735114	0.760888	0.788160	0.744204	0.748001	0.829464
4	Entire population	CatBoost	51369	25410	13952	4165	7842	0.859172	0.764165	0.770105	0.808888	0.766260	0.767135	0.848300
5	Entire population	MLP	51369	13732	17334	783	19520	0.946056	0.412968	0.956781	0.574958	0.604762	0.684874	0.875695
6	Entire population	CNN	51369	32094	470	17647	1158	0.645222	0.965175	0.025942	0.773415	0.633923	0.495559	0.618256

Explanation

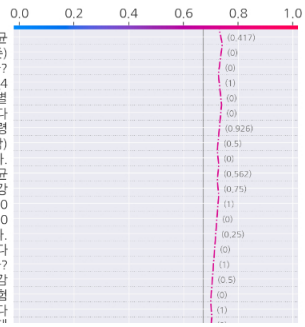
[12]:
```
if platform.system() == 'Darwin': #맥
    plt.rc('font', family='AppleGothic')
elif platform.system() == 'Windows': #윈도우
    # FONT_NAME = 'Malgun Gothic'
    FONT_NAME = 'malgun'
    plt.rc('font', family=FONT_NAME)
    plt.rcParams['font.family'] = FONT_NAME
    mpl.rc('font', family=FONT_NAME)
    sns.set(font=FONT_NAME)
    sys_font = font_manager.findSystemFonts()
    FONT_PATHS = [path for path in sys_font if 'malgun' in path]
    if len(FONT_PATHS) != 0:
        rc('font', family=font_manager.FontProperties(fname=FONT_PATHS[0]).get_name())
elif platform.system() == 'Linux':
    FONT_NAME = 'NanumGothic'
    plt.rc('font', family=FONT_NAME)
    plt.rcParams['font.family'] = FONT_NAME
    mpl.rc('font', family=FONT_NAME)
    sns.set(font=FONT_NAME)
    sys_font = font_manager.findSystemFonts()
    FONT_PATHS = [path for path in sys_font if 'NanumGothic.ttf' in path]
    if len(FONT_PATHS) != 0:
        rc('font', family=font_manager.FontProperties(fname=FONT_PATHS[0]).get_name())
```

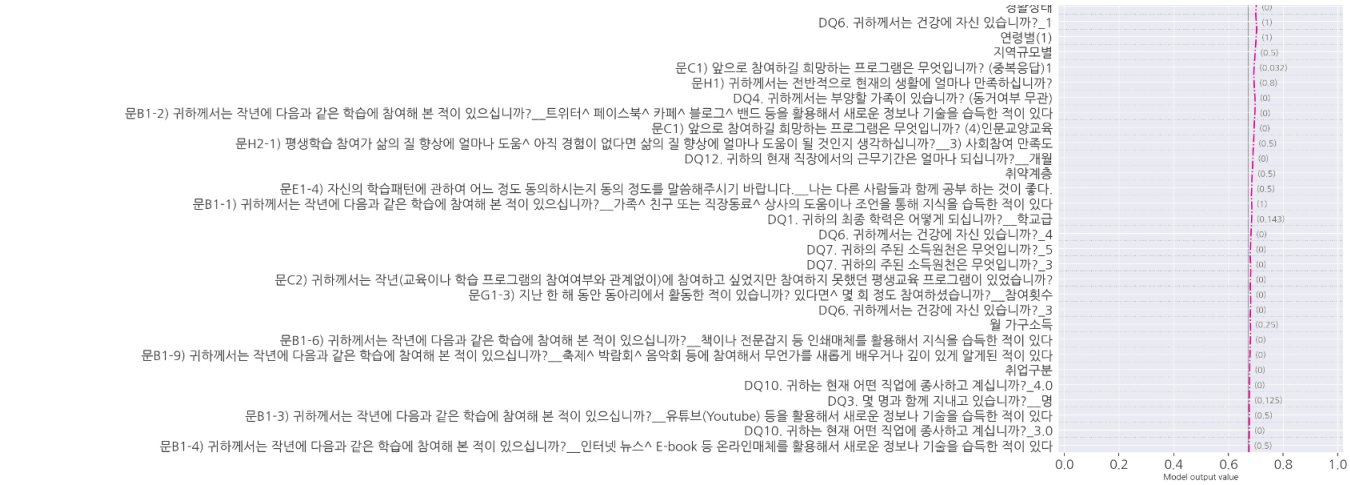
Last executed at 2025-06-03 18:52:34 in 83ms

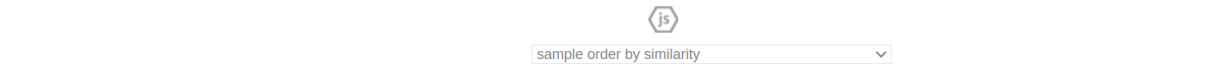
[13]:
```
model_rf, Score_te_rf, Score_trte_rf = modeling_RandomForestClassifier(X_train, Y_train,
                                                       X_test, Y_test,
                                                       X_colname=X_colname,
                                                       shap=SHAP,
                                                       X_top_display=X_TOP_DISPLAY,
                                                       dependency=not DEPENDENCY,
                                                       label_list=LABEL_LIST)
display(model_rf, Score_te_rf, Score_trte_rf)
```

Last executed at 2025-06-03 18:53:05 in 23.98s

```
Train Dataset:
Individual Explanation(1 Decision Plot -> 1 Force Plot -> 1000 Force Plot)...
```



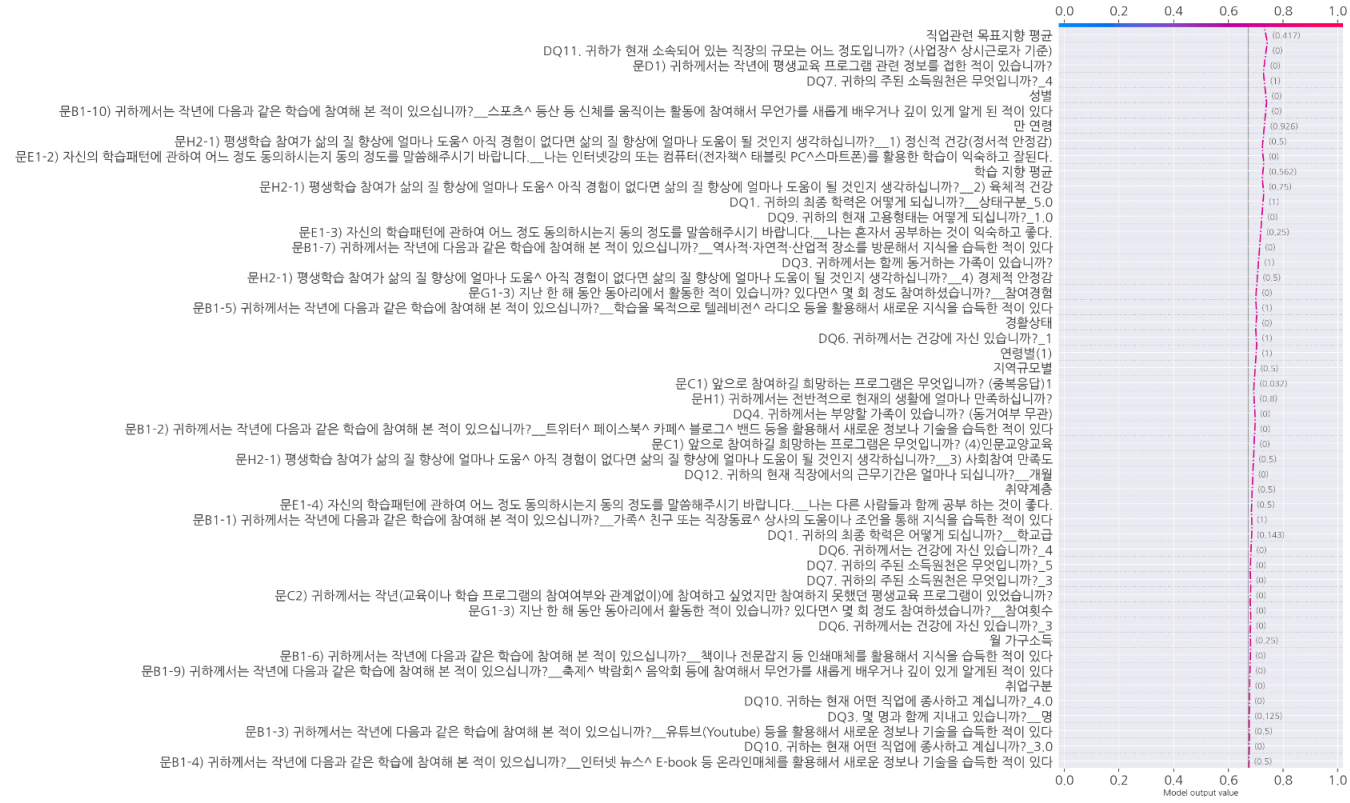




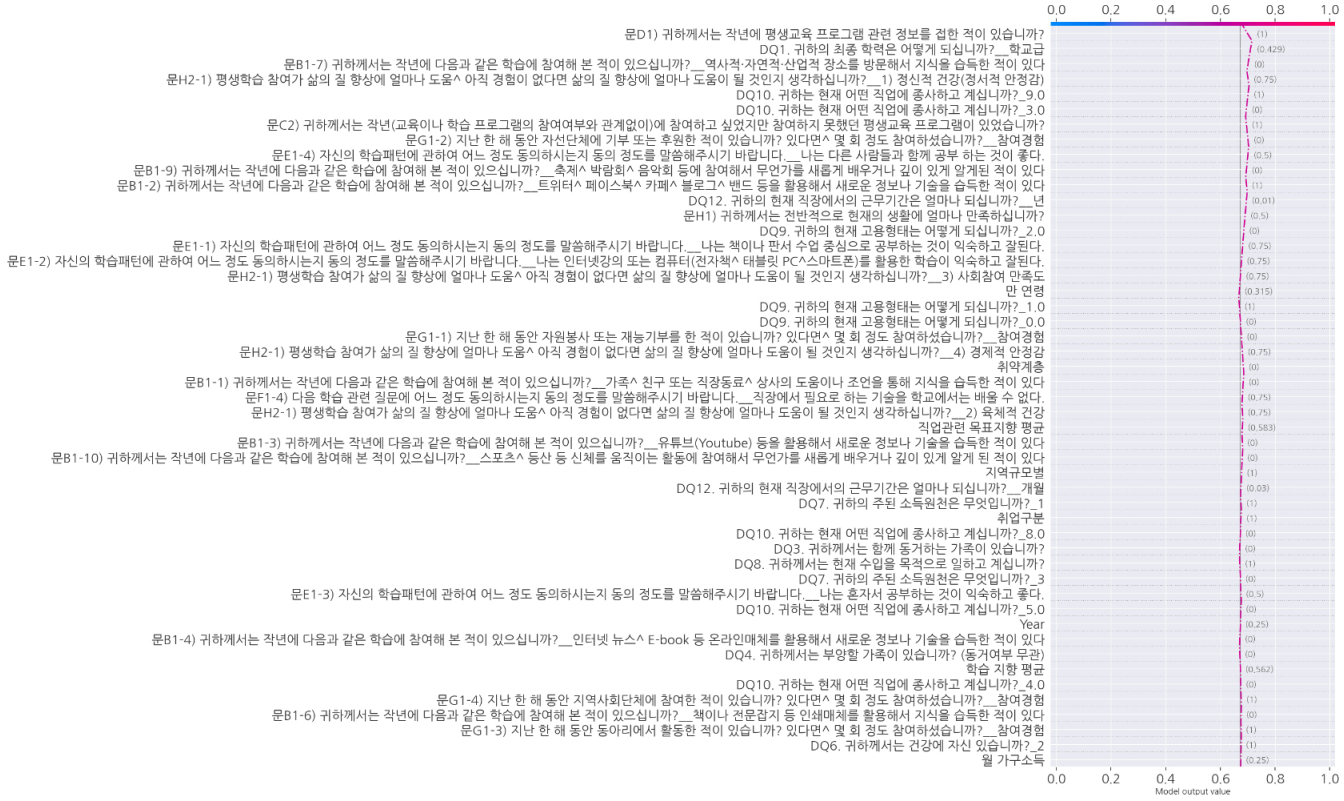

```
                                X_colname=X_colname,
                                shap=SHAP,
                                X_top_display=X_TOP_DISPLAY,
                                dependency=not DEPENDENCY,
                                label_list=LABEL_LIST)
display(model_rf, Score_te_rf, Score_trte_rf)
```

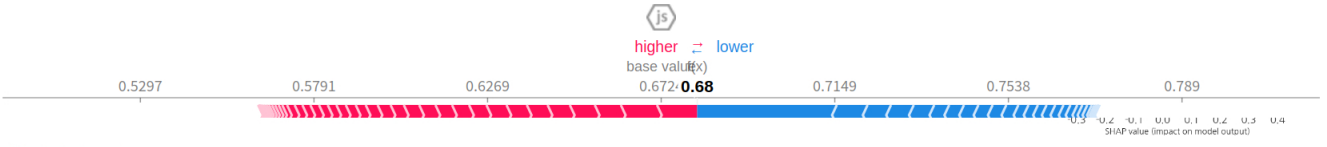
```
Last executed at 2025-06-03 18:53:05 in 23.98s
```

```
Train Dataset:
Individual Explanation(1 Decision Plot -> 1 Force Plot -> 1000 Force Plot)...
```

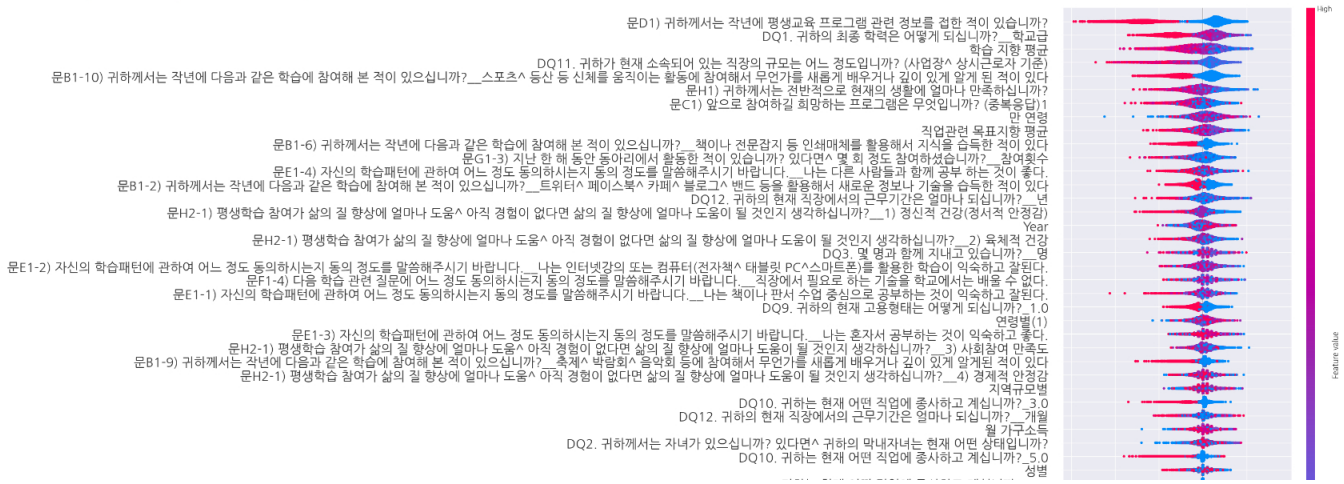


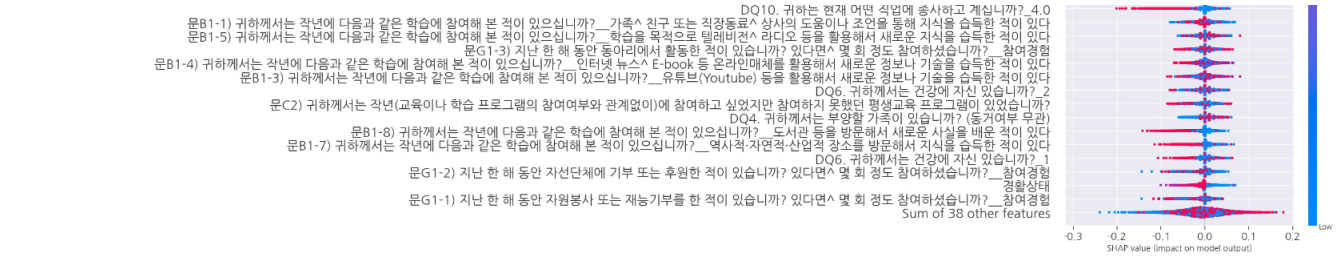
Test Dataset:
Individual Explanation(1 Decision Plot -> 1 Force Plot -> 1000 Force Plot)...



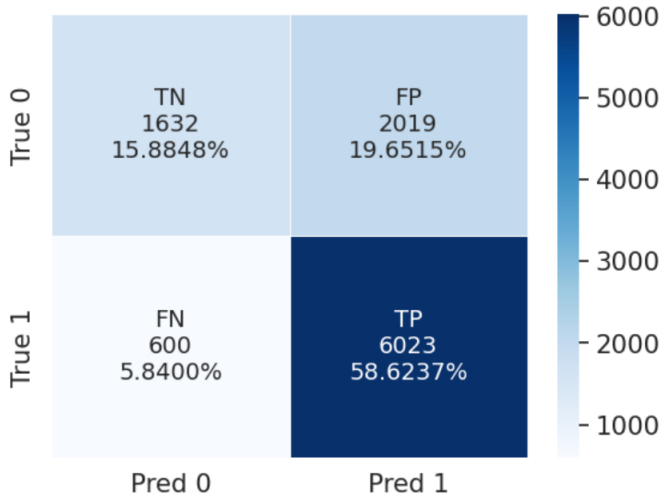


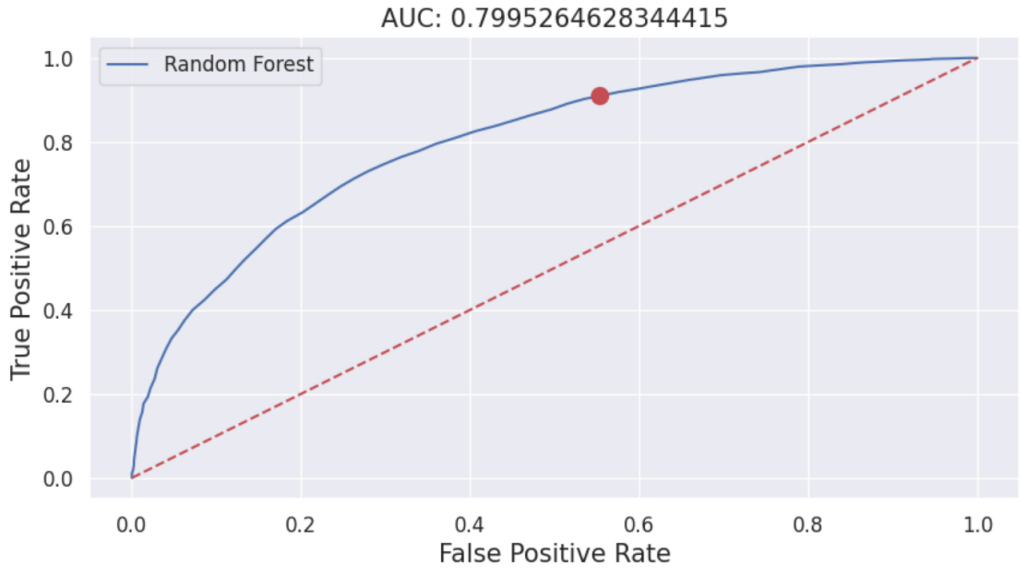
Test Dataset:
Total Explanation(Beeswarm Plot)...





Performance:





RandomForestClassifier

```
RandomForestClassifier(class_weight='balanced', random_state=123)
```

	N	True Positive	True Negative	False Positive	False Negative	Precision	Recall	Specificity	F1-score	Accuracy	Balanced Accuracy	AUC
Test set	10274	6023	1632	2019	600	0.7489	0.9094	0.4470	0.8214	0.7451	0.6782	0.7995

	N	True Positive	True Negative	False Positive	False Negative	Precision	Recall	Specificity	F1-score	Accuracy	Balanced Accuracy	AUC
Entire population	51369	32652	16097	2020	600	0.9417	0.9820	0.8885	0.9614	0.9490	0.9352	0.9887

[14]:
```
# model_xgb, Score_te_xgb, Score_trte_xgb = modeling_XGBClassifier(X_train, Y_train,
#                                                  X_test, Y_test,
#                                                  X_colname=X_colname,
#                                                  shap=SHAP,
#                                                  X_top_display=X_TOP_DISPLAY,
#                                                  dependency=not_DEPENDENCY,
#                                                  label_list=LABEL_LIST)
# display(model_xgb, Score_te_xgb, Score_trte_xgb)
```

N/A (2m 40.03s)

[15]:
```
# model_lgbm, Score_te_lgbm, Score_trte_lgbm = modeling_LGBMClassifier(X_train, Y_train,
#                                                  X_test, Y_test,
#                                                  X_colname=X_colname,
```

```
#                                                   shap=SHAP,
#                                                   X_top_display=X_TOP_DISPLAY,
#                                                   dependency=not DEPENDENCY,
#                                                   label_list=LABEL_LIST)
# display(model_lgbm, Score_te_lgbm, Score_trte_lgbm)
```

Last executed at 2025-06-03 22:18:52 in 65ms

[ ]:

[ ]:

[ ]: