캐글 코리아 4차 대회

학습내용

- 상위 솔루션을 분석해 봅니다.(2nd)

- 대회 링크 : https://www.kaggle.com/c/kakr-4th-competition/overview (https://www.kaggle.com/c/kakr-4th-competition/overview)
- 참고 링크 : https://www.kaggle.com/code/okeong/2nd-place-simple-gbm/notebook (https://www.kaggle.com/code/okeong/2nd-place-simple-gbm/notebook)

목차

01. 데이터 준비 및 라이브러리 임포트

목차로 이동하기

설치

- pip install [라이브러리명]

In [1]:

```
import os
import random

import numpy as np
import pandas as pd
from category_encoders.ordinal import OrdinalEncoder
from lightgbm import LGBMClassifier
import lightgbm as lgb
from sklearn.metrics import f1_score
from sklearn.model_selection import KFold

import warnings
warnings.filterwarnings('ignore')

from IPython.display import display

pd.options.display.max_rows = 10000
pd.options.display.max_columns = 1000
pd.options.display.max_colwidth = 1000
```

In [2]:

```
print("lightgbm ver : ", lgb.__version__)
```

```
lightgbm ver :  3.1.1
```

데이터 탐색

데이터 정보

age : 나이
workclass : 고용 형태
fnlwgt : 사람 대표성을 나타내는 가중치 (final weight의 약자)
education : 교육 수준 (최종 학력)
education_num : 교육 수준 수치
marital_status: 결혼 상태
occupation : 업종
relationship : 가족 관계
race : 인종
sex : 성별
capital_gain : 양도 소득
capital_loss : 양도 손실
hours_per_week : 주당 근무 시간
native_country : 국적
income : 수익 (예측해야 하는 값, target variable)

02. 데이터 전처리

목차로 이동하기

In [3]:

```
db_root = "data/4th_kaggle"

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

In [4]:

```
# 결측치
# 각 컬럼별 ?값이 있는 컬럼과 해당 값(라벨 인코딩) 확인하기
def get_miss_count(df, encoder):
    col_list = {}
    # print(encoder.mapping)
    for mapping in encoder.mapping:
        # print(mapping)
        # print()
        if '?' in mapping['mapping']:
            col_list[mapping['col']] = mapping['mapping']['?']

    return col_list
```

In [5]:

```
## get_miss_count() 함수 이해.
## 데이터 불러와, 전처리 후, get_miss_count() 함수 실행.
train_all = pd.read_csv(os.path.join(db_root, 'train.csv'))
test_all = pd.read_csv(os.path.join(db_root, 'test.csv'))

print(train_all.shape, test_all.shape)

train = train_all.drop(['education', 'id'], axis=1)
test = test_all.drop(['education', 'id'], axis=1)

print(train.shape, test.shape)

# 데이터 전처리
# 수익을 True, False로 변경 후, income 없애기.
# OrdinalEncoder으로 라벨 인코딩
train_target = train['income'] != '<=50K'
train_in = train.drop(['income'], axis=1)

encoder = OrdinalEncoder()
train_input_org = encoder.fit_transform(train_in, train_target)
display(train_input_org.head())
col_list = get_miss_count(train_input_org.copy(), encoder)
col_list
```

```
(26049, 16) (6512, 15)
(26049, 14) (6512, 13)
```

	age	workclass	fnlwgt	education_num	marital_status	occupation	relationship	race	sex
0	40	1	168538	9	1	1	1	1	1
1	17	1	101626	5	2	2	2	1	1
2	18	1	353358	10	2	3	2	1	1
3	21	1	151158	10	2	4	2	1	2
4	24	1	122234	10	2	5	3	2	2

Out[5]:

```
{'workclass': 3, 'occupation': 7, 'native_country': 2}
```

In [6]:

```python
# 결측치가 있는 컬럼 확보
# df
# encoder
# miss_rate

def augment_feature(df, encoder, miss_rate=0.1):
    # augment missing features
    col_list = get_miss_count(df, encoder)

    # {'workclass': 3, 'occupation': 7, 'native_country': 2}
    # num_miss : 전체 행 데이터 중에서 miss_rate 비율만큼 데이터 샘플을 뽑는다.
    for col in col_list.keys():
        num_miss = int(df.shape[0] * miss_rate)
        sample = df.sample(num_miss).index

        # 해당 샘플에 라벨 인코딩한 값을 넣는다.
        # {'workclass': 3, 'occupation': 7, 'native_country': 2}
        df.loc[sample, col] = col_list[col]
        # print(df.loc[sample,col])

    # gaussian noise
    noise = ['age', 'hours_per_week']

    for col in noise:
        min_val, max_val = min(df[col]), max(df[col])
        scale = (max_val - min_val) / 30
        # print(min_val, max_val, scale)

        # random.normal(loc, scale, size)
        # loc : 분포의 평균, scale : 표준 편차, size : 추출할 데이터 사이즈
        add_noise = np.random.normal(0.0, scale, df.shape[0])

        # 정규분포 noise를 더해주고, 양쪽 min, max를 자른다.
        df[col] = (df[col] + add_noise.astype(int)).clip(min_val, max_val)

    return df
```

In [7]:

```
# 함수 이해
param = {
    'seed': 20863,
    'num_fold': 15,
    'num_aggr': 2,
    'miss_rate': 0.1
}

miss_rate = float(param['miss_rate'])
encoder = OrdinalEncoder()
train_input_org = encoder.fit_transform(train_in, train_target)

train_input = augment_feature(train_input_org.copy(), encoder, miss_rate)
train_input
```

Out[7]:

	age	workclass	fnlwgt	education_num	marital_status	occupation	relationship	race	s
0	35	1	168538	9	1	1	1	1	
1	19	1	101626	5	2	2	2	1	
2	20	1	353358	10	2	3	2	1	
3	21	1	151158	10	2	4	2	1	
4	26	1	122234	10	2	5	3	2	
...	...	...	...	...	...	...	...	...	
26044	58	1	250201	7	1	3	1	1	
26045	23	3	238092	13	2	4	2	1	
26046	78	3	165694	14	6	7	3	1	
26047	27	4	151626	9	2	4	2	2	
26048	20	3	99891	10	2	7	2	1	

26049 rows × 13 columns

03. 모델 구축하기

목차로 이동하기

In [8]:

```
# param     : 'seed': 20863, 'num_fold': 15, 'num_aggr': 2, 'miss_rate': 0.1
# gbm_param : "min_child_samples": 10, "n_estimators": 80, "num_leaves": 25,
#                     "subsample_freq": 4, "learning_rate": 0.3
# train     : train 데이터 셋
# test      : test 데이터 셋

def run(param, gbm_param, train, test):
    s = param['seed']
    rand_seed(s)

    miss_rate = float(param['miss_rate'])
    num_fold = int(param['num_fold'])
    num_aggr = int(param['num_aggr'])

    train = train.drop(['education', 'id'], axis=1)
    test = test.drop(['education', 'id'], axis=1)

    y_preds = np.zeros(test.shape[0])

    train_target = train['income'] != '<=50K'
    train_in = train.drop(['income'], axis=1)

    encoder = OrdinalEncoder()
    train_input_org = encoder.fit_transform(train_in, train_target)
    test = encoder.transform(test)

    f1_list = []

    for c in range(num_aggr):
        folds = KFold(n_splits=num_fold, shuffle=True)

        train_input = augment_feature(train_input_org.copy(), encoder, miss_rate)
        splits = folds.split(train_input, train_target)

        for fold_n, (train_index, valid_index) in enumerate(splits):
            model = LGBMClassifier(objective='binary',
                                   verbose=-1,
                                   **gbm_param)

            X_train, X_valid = train_input.iloc[train_index], train_input.iloc[valid
            y_train, y_valid = train_target.iloc[train_index], train_target.iloc[val

            eval_set = [(X_valid, y_valid)]

            model.fit(X_train, y_train,
                      eval_set=eval_set,
                      early_stopping_rounds=10,
                      verbose=False)

            predict_valid = model.predict(X_valid)

            f1 = f1_score(y_valid, predict_valid, average='micro')
            f1_list.append(f1)

            predict_test = model.predict(test)
            y_pred = predict_test.astype(int) / (num_fold * num_aggr)
            y_preds += y_pred

    val_f1 = np.mean(f1_list)
```

```
    print(f'val_f1={val_f1}')

    sample_submission = pd.read_csv(os.path.join(db_root, 'sample_submission.csv'))
    sample_submission['prediction'] = (y_preds > 0.5).astype(int)

    test_csv = f'submission_4th.csv'
    sample_submission.to_csv(test_csv, index=False)
```

In [9]:

```
# 기본 파라미터 설정
def default_param():
    return {
        "min_child_samples": 10,
        "n_estimators": 80,
        "num_leaves": 25,
        "subsample_freq": 4,
        "learning_rate": 0.3
    }

if __name__ == '__main__':
    os.environ['OMP_NUM_THREADS'] = "4"

    train_all = pd.read_csv(os.path.join(db_root, 'train.csv'))
    test_all = pd.read_csv(os.path.join(db_root, 'test.csv'))

    param = {
        'seed': 20863,
        'num_fold': 15,
        'num_aggr': 2,
        'miss_rate': 0.1
    }

    gbm_param = default_param()

    if 'num_leaves' in gbm_param:
        gbm_param['num_leaves'] = int(gbm_param['num_leaves'])
    if 'n_estimators' in gbm_param:
        gbm_param['n_estimators'] = int(gbm_param['n_estimators'])

    run(param, gbm_param, train_all, test_all)
```

```
val_f1=0.8698792080206087
```