2. 범죄 유형 분류 - AutoML (pycaret)

https://dacon.io/competitions/official/236109/codeshare/8423

주제/목표

범죄 유형을 분류하는 AI 알고리즘 개발

사건 발생 장소 및 기후 데이터 분석을 통해 어떤 유형의 범죄가 발생할지 예측, 예측 결과를 바탕으로 범죄에 대한 빠른 대응이 가능하도록 함

데이터 : 음악 샘플의 특징 정보

- **train (84406개)**
  - 84406개의 데이터
  - ID : 샘플 고유 ID
  - `feature` : 각 범죄 관련 변수
    - 시간 관련 : (사건 발생) 월, 요일, 시간
    - 장소 관련 : 소관경찰서(사건 발생 구역의 담당 경찰서), 소관지역(사건 발생 구역), 사건 발생거리(가장 가까운 경찰서에서 사건 현장까지의 거리)
    - 날씨 관련 : 강수량(mm), 강설량(mm), 적설량(cm), 풍향, 안개, 짙은 안개, 번개, 진눈깨비, 서리, 연기/연무, 눈날림
    - 범죄발생지(ex. 차도, 주거지, 주유소, 인도...)
  - `label` : 범죄 타입 (총 3개)
    - 0 : 강도 / 1 : 절도 / 2: 상해
- **test (17289개)**

코드 흐름

AutoML 중 하나인 pycaret을 통해 모델 선정 > GridSearchCV로 하이퍼 파라미터 튜닝 > 최종 모델 결정

(0) 시드 고정

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

seed_everything(42) # Seed 고정
```

*나중에 그룹 프로젝트할 때 유용하게 사용할 수 있을 듯!

(1) 데이터 전처리

- ID drop
- LableEncoder() : 요일, 범죄 발생지와 같은 범주형+문자열 자료를 숫자형으로 바꿔줌

(2) 모델 선택 ; pycaret

`from pycaret.classification import *`

```
clf = setup(data=train,
            target='TARGET',
            normalize=True,
            # polynomial_features=True,
            session_id=42,
            use_gpu =True
            )
```

- setup() : Train data, Test data, label column 등을 설정하는 부분이며, normalize, normalize_method, transformation, fold_strategy 등의 옵션을 통해 여러 전처리 기법을 적용할 수 있음
- session_id : random_state
- use_gpu : gpu 사용 여부

- setup() 함수 실행 후 models() 메서드를 입력하면 어떤 모델을 사용하고 있는지 확인할 수 있음

```
top3 = compare_models(sort='f1', n_select=5, fold=5)
```

- compare_models() : 입력한 모델들의 성능(MAE, MSE, RMSE, R2 등)을 데이터 프레임 형태로 제공함
- sort : 모델을 sort할 때 기준으로 사용할 metric
- n_select : sort된 모델을 top n 순서로 저장

결과 확인 코드

```
result = pull() ; display(result)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.5461	0.6989	0.5461	0.5418	0.5358	0.2840	0.2894	3.138
gbc	Gradient Boosting Classifier	0.5454	0.6957	0.5454	0.5424	0.5347	0.2826	0.2885	16.822
ada	Ada Boost Classifier	0.5391	0.6861	0.5391	0.5363	0.5288	0.2743	0.2796	2.076
rf	Random Forest Classifier	0.5202	0.6705	0.5202	0.5121	0.5093	0.2437	0.2477	2.768

Blending

```
blended = blend_models(estimator_list = top3[:3], fold = 5, weights=[0.5,0.3,0.2])
```

가장 성능이 좋은 것으로 나왔던 모델 3개를 혼합한 모델을 생성

>이 결과 lightgbm의 결과 대비 유의미한 성능 향상을 얻지 못하여 LightGBM 모델 하나에 대해 하이퍼 파라미터 튜닝하는 것이 좋다고 판단

(3) 하이퍼 파라미터 튜닝 ; GridSearchCV

(4) Class Weight

```
train['TARGET'].hist()
```

class 수가 균형적이라고 보기에는 약간의 차이가 있어 lightgbm 모델 생성 시 class_weight = 'balanced' 옵션을 지정

```
mdl = lgb.LGBMClassifier(boosting_type= 'dart',
                         objective = 'multiclass',
                         n_jobs = 5,
                         colsample_bytree =0.75,
                         min_split_gain =.5,
                         n_estimators=2048,
                         reg_alpha =.75,
                         subsample =.5,
                         max_depth = -1,
                         device= 'gpu',
                         random_state = 42,
                         class_weight='balanced',
                         num_leaves = 256
                )
```

배운점, 느낀점

타겟값이 불균형하다고 판단되면 LGBMClassifier() 모델 생성 시 class_weight='balanced' 옵션을 사용하여 해결할 수 있음

모델 생성 전 EDA 과정에서 target의 분포를 확인하는 것이 더 좋아보임

pycaret 참고할만한 페이지

https://minimin2.tistory.com/137