https://www.kaggle.com/code/darkdevil18/0-98530-can-you-eat

데이터

- **train_df**: 버섯의 외형적 특징 피처들 (총 20개), UCI Mushroom dataset을 기반으로 딥러닝을 통해 생성됨.
  - ['cap-diameter', 'cap-shape', 'cap-surface', 'cap-color', 'does-bruise-or-bleed', 'gill-attachment', 'gill-spacing', 'gill-color', 'stem-height', 'stem-width', 'stem-root', 'stem-surface', 'stem-color', 'veil-type', 'veil-color', 'has-ring', 'ring-type', 'spore-print-color', 'habitat', 'season']
  - target: 예측 목표, 버섯의 독성 여부. e 또는 p.
- **orig_df**: UCI Mushroom dataset의 원본 데이터셋. 이용여부는 선택.
- **test_df**: 이 데이터 셋의 target인 class를 예측하는 것이 목표. 역시 UCI Mushroom dataset을 기반으로 딥러닝을 통해 생성됨.

코드 흐름

(1) **EDA**:

- 범주형 피처가 17개, 숫자형 피처가 3개. 둘을 구분해서 eda 및 시각화함.
- describe(include='0') 메소드는 범주형 피처에 대한 count, unique, top, freq 정보를 알려줌.
- 범주형 피처에 대한 시각화는 countplot으로 막대그래프를 그렸고 hue=target으로 하여 각 범주별로 타겟이 얼마나 큰 차이를 보이는지 한눈에 알아볼 수 있도록 함.
- 숫자형 피처에 대한 시각화는 히스토그램과 선그래프를 겹쳐그림. 역시 각 타겟이 피처에 따라 얼마나 큰 차이를 보이는지 한눈에 파악가능. 숫자형 피처는 그외에도 box plot, violin plot을 그림.

(2) **Feature Engineering & Processing**:

- Null value를 채우는 과정. 숫자형 피처인 cap-diameter는 train 데이터와 original 데이터 전체의 cap-diameter의 평균을 구해서 채워넣음. 그 외의 숫자형 피처엔 null value가 없음. 범주형 피처는 아래의 전처리 과정을 적용함.

```
def cleaner(df):
    for col in categorical_features:
        # df[col]에 있는 결측값(NaN)을 'missing'이라는 문자열로 채움.
        df[col] = df[col].fillna('missing')
        # df[col]에서 해당 열의 값이 100번 미만으로 나타나는 경우, 그 값을 'noise'라는 값으로 대체함. 데이터 내에서 빈도가 낮은(희귀한) 카테고리를 하나의 그룹(노이즈)으로 묶어주는 역할.
        df.loc[df[col].value_counts(dropna=False)[df[col]].values < 100, col] = "noise"
        # df[col]의 데이터 타입을 category로 변환함. 범주형 데이터로 변환하면 메모리 사용을 최적화하고, 일부 머신러닝 모델에서도 범주형으로 처리하기 쉬움.
        df[col] = df[col].astype('category')
```

```
    return df
```

- LabelEncoder을 이용해서 타겟 피처를 범주형에서 숫자로 변환함.

(3) **Modeling**:
- 평가방식으로 cv=5의 k-fold 방식을 적용해서 f1_score, matthew_corrcoef를 구하고 confusion matrix까지 시각화함.
- XGBClassifier, CatBoostClassifier, LGFMClassifier 사용.
- 각 모델의 예측 확률을 평균내어 가장 높은 확률 값을 가지는 클래스를 최종 예측으로 선택하여 `oof_preds`에 저장.

(4) **Hyperparameter Tuning**:
베이지안 최적화 기법의 optuna 패키지를 이용함. suggest_int, suggest_float 등의 API를 이용해서 Seach space를 설정한 점이 새로움.

인상깊었던 점

범주형 피처와 숫자형 피처를 구분하여 EDA한 점이 인상깊었다. 하이퍼 파라미터 튜닝을 위해 search space를 생성할 때 suggest_int, suggest_float 등의 API를 이용하는 것이 유용하겠다.