데이콘 x BDA 제 2회 학습자 수료 예측 AI 경진대회

Team: Placeholder/ 15th place/ Private Score: 0.41626

Feb 25th 2026

Presented by knowin_kyeong (Team Leader), William Han, 긴카호, 코나치

데이콘 x BDA 제 2회 학습자 수료 예측 AI 경진대회

Problem Definition:
각각의 학습자 정보를 입력으로 받아 수료 여부 (completed)를 예측

Input: 각 학생의 BDA 설문조사 응답
Output: 각 학생의 최종 수료 여부 분류 (Classification)
Approach:

- 데이터의 수가 매우 적고 noise가 큼 -> Robust한 모델 제작 필요 -> Seed Ensemble 결정
- 수가 많지만 overfitting 위험성이 높은 text column + 유의하지만 수가 적은 numeric column로 구성
- 각각 범주형/숫자형 column에 특화된 catboost, xgboost로 처리
- 예측값을 Hard voting (두 모델이 모두 true라고 해야 수료한 것으로 예측)
- 기수에 따른 설문 응답 형식 차이로 인한 distribution shift에 유의하여, Local CV를 너무 과신하지 않음

Caution) 800개 이하의 적은 train data로 인한 Local CV perturbation 고려 필수 (다른 대회와 다른 점)

데이콘 x BDA 제 2회 학습자 수료 예측 AI 경진대회

Data Structure & Key Insights (EDA)

데이터 구조
- train.csv의 데이터 수는 747개로 일반적인 대회보다 데이터 수가 많이 적었음
- train.csv의 completed가 1인 데이터는 전체의 약 29.8%로 class imbalance가 존재
- 전체적으로 정형 형식으로 구성되어 있음

Feature
- 몇 개의 숫자형 (numeric) feature, 대부분의 범주형 (categorical) feature, 몇 개의 자유 응답 text feature
- School1 같은 몇몇 범주형 feature는 숫자로 Encoding된 상태
- 어떤 Feature의 경우 ,로 구분된 여러 값들이 train.csv 한 칸에 포함되어 있음

➔ 아마 설문조사의 “다음 중 여러 개를 고르시오“에 해당되는 것으로 추정

➔ 또한 설문조사 항목 중 “기타“에 해당하는 게 text feature로 들어간 것으로 추정됨

결측치
- 대부분의 값이 채워져 있지만 소량의 결측치도 존재함
- 범주형이나 text feature에서 주로 관측

결론
- 데이터의 수가 적은 것을 의식해 overfitting 억제, 특히 text feature로 인한 overfitting을 억제해야 함

데이콘 x BDA 제 2회 학습자 수료 예측 AI 경진대회

Model 1: Catboost (1/4)

모델 목적: text 형식으로 들어온 column들의 의미를 이해하고 수료 예측에 반영

전처리

- (공통) 컴퓨터 전공 여부를 keyword_regex로 정해진 substring이 포함되었는지를 통해 판별

```
    # IT Major
    keyword_regex = 'IT|정보|컴퓨터|소프트'
```

- detect_text_cols 함수와 train.csv를 사용해 string 형식 데이터 중 “자유응답 text column”을 분리

- 또한 EDA 결과 자유응답 형식이나 여러 고유값을 가지는 column들을 text column들로 명시적으로 지정

```
def detect_text_cols(X: pd.DataFrame, candidate_cols: List[str]) -> List[str]:
    """
    Detect text columns from candidate features.
    Check uniques >= 30 and mean length >= 15 for text columns in train.csv
    """
    text_cols = []
    for c in candidate_cols:
        if c not in X.columns:
            continue

        s = X[c].astype("string")
        non = s.dropna()
        if len(non) == 0:
            continue

        mean_len = non.str.len().mean()
        nunq = non.nunique(dropna=True)

        if mean_len >= 15 and nunq >= 30:
            text_cols.append(c)
    return text_cols
```

```
MANUAL_TEXT_COLS = [
    "whyBDA", "what_to_gain", "incumbents_lecture",
    "certificate_acquisition", "incumbents_lecture_scale_reason", "onedayclass_topic"
]

# keep some short-ish domain/job fields as candidates
KNOWN_TEXT_COLS = [
    "expected_domain",
    "desired_job",
    "desired_job_except_data",
    "desired_certificate",
    "interested_company"
]
```

데이콘 x BDA 제 2회 학습자 수료 예측 AI 경진대회

Model 1: Catboost (2/4)

전처리

- 위에서 지정한 text_column들과 categorical_column, 그리고 numerical_column들을 분리

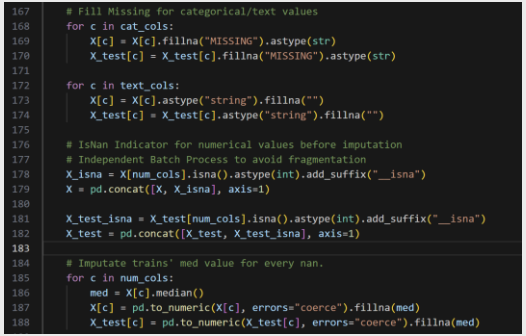
- 숫자로 encoding되어 있지만 범주형 성격이 강한 column들은 string 형 변환을 통해 categorical로 명시

```
37  FORCE_CAT_COLS = ["school1", "class1", "class2", "class3", "class4"]
```

- 또한 train.csv 기준 결측 값 비율이 70%를 넘길 경우 그 column은 drop하여 noise overfitting 방지

- Numerical column은 원래 값이 결측 값인지를 나타내는 __isna 파생 변수 생성

<결측 값 보간 기준>
Categorical, Text: 각각 “MISSING”, “”로 결측 값 대체
Numerical: Train 데이터의 중앙값으로 보간 (Test 데이터 통계량 사용 X)



```
# Fill Missing for categorical/text values
for c in cat_cols:
    X[c] = X[c].fillna("MISSING").astype(str)
    X_test[c] = X_test[c].fillna("MISSING").astype(str)

for c in text_cols:
    X[c] = X[c].astype("string").fillna("")
    X_test[c] = X_test[c].astype("string").fillna("")

# IsNan Indicator for numerical values before imputation
# Independent Batch Process to avoid fragmentation
X_isna = X[num_cols].isna().astype(int).add_suffix("__isna")
X = pd.concat([X, X_isna], axis=1)

X_test_isna = X_test[num_cols].isna().astype(int).add_suffix("__isna")
X_test = pd.concat([X_test, X_test_isna], axis=1)

# Imputate trains' med value for every nan.
for c in num_cols:
    med = X[c].median()
    X[c] = pd.to_numeric(X[c], errors="coerce").fillna(med)
    X_test[c] = pd.to_numeric(X_test[c], errors="coerce").fillna(med)
```

데이콘 x BDA 제 2회 학습자 수료 예측 AI 경진대회

Model 1: Catboost (3/4)

전처리

- 마지막으로 각 index별 독립적으로 nan_count를 세서 파생 변수에 추가

- 또한 time_input를 로그 변환해서 log_time_input를 파생 변수로 추가

```
# Nan Count & Log Time
X['nan_count'] = X.isnull().sum(axis=1)
X_test['nan_count'] = X_test.isnull().sum(axis=1)
if 'time_input' in X.columns:
    X['log_time_input'] = np.log1p(pd.to_numeric(X['time_input'], errors='coerce').fillna(0))
    X_test['log_time_input'] = np.log1p(pd.to_numeric(X_test['time_input'], errors='coerce').fillna(0))
```

- 모델을 학습시킬 때 이 정보를 Catboost 분류기에 넘겨서 범주형/텍스트형 자료를 처리하도록 함

```
415     ens_oof, ens_test = run_final_training(
416         best_params, X, y, groups, X_test,
417         cat_feature_indices, text_feature_indices
418     )
```

데이콘 x BDA 제 2회 학습자 수료 예측 AI 경진대회

Model 1: Catboost (4/4)

모델 학습 (school_id 기반 Stratified Group 5Fold를 사용해 Fold 분류)

- Optuna를 사용해 Hyperparameters 탐색, 단일 seed & 5 fold로 oof validation logloss minimize
- 찾은 params로 5 seed ensemble 수행 (5 seed * 5 fold)
- Seed ensemble 방법) 각 seed에 대한 oof validation, test inference 결과를 독립적으로 평균을 구함
- 마지막으로 oof validation의 F1 score를 극대화하는 final threshold로 예측 결과를 0/1로 분류

  Test data의 상위 n%를 1로 예측하는 것은 Data leakage 위험이 있으므로

  oof validation을 통해 얻은 고정된 threshold로 예측 수행
- 실제로는 Optuna Hyperparameters를 찾는 과정 (Tuning)의 경우 비결정성이 심해 이전에 찾아 놓은 Hyperparameters를 코드에 첨부하여 (MODE = Load를 통해) 실행할 수 있게 재현성을 보장함
- Device 환경에 따라 (MODE = Tune으로 변경하여) Tuning부터 실행할 수 있도록 Option 제공

데이콘 x BDA 제 2회 학습자 수료 예측 AI 경진대회

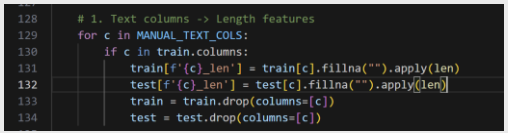
Model 2: Xgboost (1/3)

모델 목적: time_input과 같은 주요한 숫자 feature들에 주목하여 수료 예측에 반영

전처리

- (공통) 컴퓨터 전공 여부를 keyword_regex로 정해진 substring이 포함되었는지를 통해 판별

- 자유응답 text column의 경우 응답의 내용 대신 응답 길이만 파생 변수로 사용



```
# 1. Text columns -> Length features
for c in MANUAL_TEXT_COLS:
    if c in train.columns:
        train[f'{c}_len'] = train[c].fillna("").apply(len)
        test[f'{c}_len'] = test[c].fillna("").apply(len)
        train = train.drop(columns=[c])
        test = test.drop(columns=[c])
```

- 예외적으로 유의한 몇 자유응답 text column의 경우 특정 키워드가 포함되었는지 여부로 파생 변수 생성

```
    # 2. Certificate
    if 'certificate_acquisition' in df.columns:
        df['cert_count'] = df['certificate_acquisition'].fillna("").apply(lambda x: x.count(',') + 1 if x != "" else 0)
        df['has_adsp'] = df['certificate_acquisition'].fillna("").str.contains('ADsP', case=False).astype(int)
        df['has_sqld'] = df['certificate_acquisition'].fillna("").str.contains('SQLD', case=False).astype(int)

    # 3. Job Keywords
    if 'desired_job_except_data' in df.columns:
        df['want_uiux'] = df['desired_job_except_data'].fillna("").str.contains('UI|UX', case=False).astype(int)
        df['want_pm'] = df['desired_job_except_data'].fillna("").str.contains('PM|기획', case=False).astype(int)
```

데이콘 x BDA 제 2회 학습자 수료 예측 AI 경진대회

Model 2: Xgboost (2/3)

전처리

- 범주형 feature의 경우 train.csv의 고유값을 사용해 숫자로 변환 (Data leakage 방지)
- Train.csv에 없던 값을 test.csv에서 마주한 경우 -1(Unknown)으로 Encoding
- 나머지 과정은 catboost와 동일, 예외적으로 school1==0인 index에서 train.csv의 이상치가 많아 따로 분류

```python
    # 2. Categorical columns -> Custom Label Encoding
    cat_cols = [c for c in train.columns if train[c].dtype == 'object']

    for c in cat_cols:
        # Fill NA with specific string to treat it as a category
        train_vals = train[c].fillna("MISSING").astype(str)
        test_vals = test[c].fillna("MISSING").astype(str)

        # Create mapping ONLY from Train unique values
        unique_train = train_vals.unique()
        mapping = {val: i for i, val in enumerate(unique_train)}

        # Transform Train
        train[c] = train_vals.map(mapping)

        # Transform Test (Map unknown values to -1)
        test[c] = test_vals.apply(lambda x: mapping.get(x, -1))

        # Convert to int
        train[c] = train[c].astype(int)
        test[c] = test[c].astype(int)

    return train, test
```

```python
    # 4. School
    if 'school1' in df.columns:
        df['is_school_0'] = (df['school1'] == 0).astype(int)
```

데이콘 x BDA 제 2회 학습자 수료 예측 AI 경진대회

Model 2: Xgboost (3/3)

모델 학습 (school id 기반 Stratified Group 5Fold를 사용해 Fold 분류)

- Optuna를 사용해 Hyperparameters 탐색, 단일 seed & 5 fold로 oof validation logloss minimize
- 찾은 params로 10 seed ensemble 수행 (10 seed * 5 fold)
- Seed ensemble 방법) Catboost와 동일

  역시 Data leakage를 피하기 위해 비율이 아닌 oof에서 구한 고정된 threshold 값으로 분류
- 마찬가지로 Device 환경에 따라 (MODE = Tune으로 변경하여) Tuning부터 실행할 수 있도록 Option 제공

데이콘 x BDA 제 2회 학습자 수료 예측 AI 경진대회

Reproducibility (재현성)

재현성 고정을 위해 requirements.txt로 패키지 버전 명시

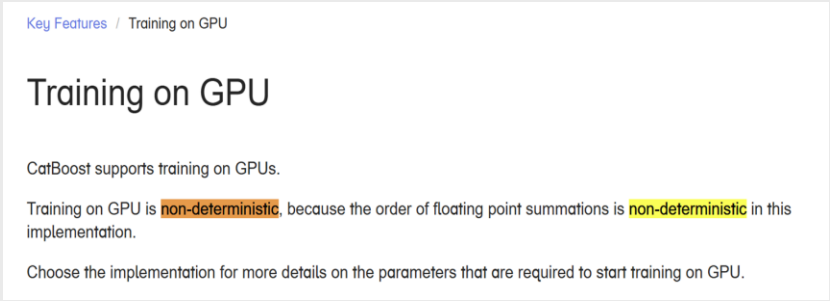
- 또한 코드의 각 부분마다 seed 고정 로직을 추가

```python
def seed_everything(seed=42):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)

    torch.manual_seed(seed)
    if torch.cuda.is_available():
        torch.cuda.manual_seed(seed)
        torch.cuda.manual_seed_all(seed)
    torch.backends.cudnn.deterministic = True
    torch.backends.cudnn.benchmark = False

    os.environ['CUBLAS_WORKSPACE_CONFIG'] = ':4096:8'
```

- 그러나 catboost의 경우 라이브러리 level 비결정성 존재 -> 몇 개 정도 예측 결과가 달라질 수 있음



Key Features / Training on GPU

Training on GPU

CatBoost supports training on GPUs.

Training on GPU is non-deterministic, because the order of floating point summations is non-deterministic in this implementation.

Choose the implementation for more details on the parameters that are required to start training on GPU.

데이콘 x BDA 제 2회 학습자 수료 예측 AI 경진대회

Ensemble

- Xgboost와 Catboost의 oof validation 파일의 확률 분포가 달라 soft ensemble은 무리가 있다고 판단
- 결국 두 모델이 모두 1(수료함)이라고 예측할 때만 최종 결과를 1로 예측하도록 hard voting

```
75    sub_ens = (sub_cb['completed'] * 0.5) + (sub_xgb['completed'] * 0.5)
76    sub_final = (sub_ens > 0.5).astype(int)
```

- 전체적으로 train.csv의 1 비율이 낮아 (30% 정도) F1 score를 극대화하기 위해 모델이 False Negative (FN, 실제 값이 1인 데이터를 0으로 예측하는 것)을 극도로 혐오하는 경향성이 있음
- 따라서 모호한 데이터는 전부 1로 예측하여 각 모델 oof validation의 1 비율이 70% 정도인 문제점 발생
- Solution: Hard voting을 통해 과대평가된 1 비율을 줄이면서도 FN 증가를 어느 정도 억제할 수 있음

이는 “각 모델이 명백히 수료하지 못할 것 같은 학생들을 걸러내는 filter로 작용한다“ 고 직관적으로 이해 가능

데이콘 x BDA 제 2회 학습자 수료 예측 AI 경진대회

Solutions / Conclusion

- 두 tree 기반 모델을 사용해 target의 수료 여부를 예측하는 solution을 제작
- 0, 1을 구분짓는 threshold는 data leakage 문제를 막기 위해 비율 기반(상위 n%)를 사용하는 대신 oof validation에서 고정된 값을 사용하도록 함
- 예측이 1로 편중되는 것을 억제하기 위해 majority hard voting 사용
- 전체적으로 data 수가 작았기 때문에 overfitting을 억제하고 robust한 모델을 만들기 위해 seed ensemble 사용

- 최종적으로 public 12등, private 15등을 달성할 수 있었음