LightGBM을 활용한 판결문
주장/사실/판단/결론 분류 모델

목표

- AIHub에서 텍스트 **데이터 수집**을 진행한다.
- 데이터에 대한 **탐색적 분석을 수행**하고 활용 **목적에 알맞는 전처리를 수행**한다.
- **TF-IDF를 활용**해 **텍스트 데이터를 임베딩**한다.
- **임베딩된 데이터**로 **LightGBM모델**을 **학습**시켜 **판결문의 주장/사실/판단/결론**에 대한 **분류를 수행**한다.

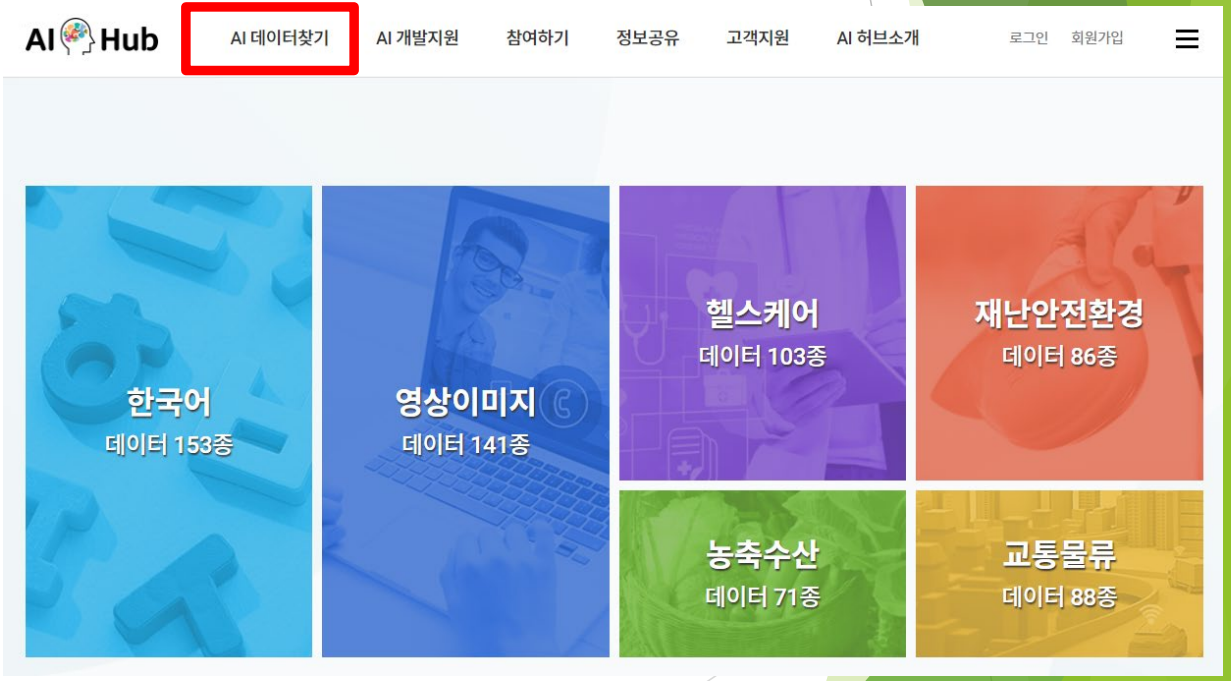
진행 순서

1. 데이터 수집
2. EDA 및 전처리
3. 모델링 및 결과 확인

1. 데이터 수집

1.1 AIHub에서의 데이터 수집

- AIHub 사이트에 접속한 뒤 상단의 AI 데이터찾기를 클릭한다.

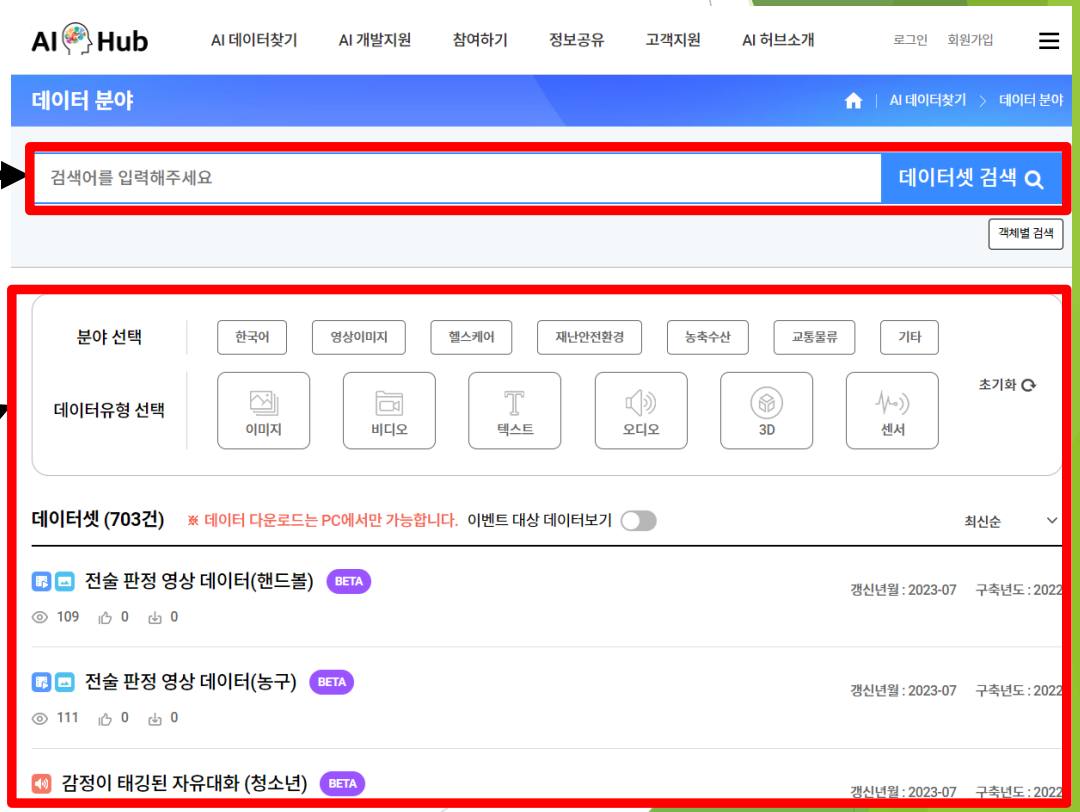


1. 데이터 수집

1.1 AIHub에서의 데이터 수집

1. 검색창에 사용하고자 하는 데이터를 검색하거나

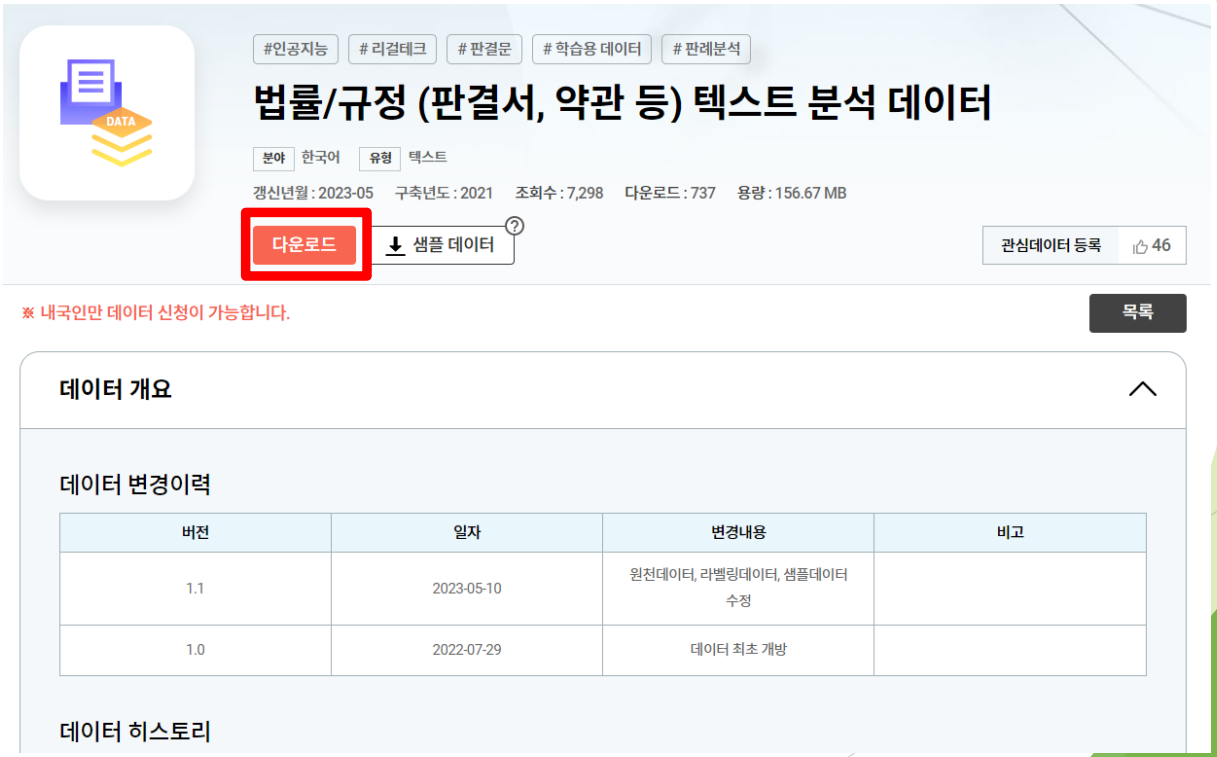
2. 아래의 데이터 중 사용하고자 하는 데이터를 클릭한다.



1. 데이터 수집

1.1 AIHub에서의 데이터 수집

- 본 실습에서는 법률/규정 (판결서, 약관 등) 텍스트 분석 데이터를 사용한다.



2. EDA 및 전처리

2.1 라이브러리 import 및 초기설정

- 사용하고자 하는 분석에 필요한 라이브러리들을 불러온다.

```
import pandas as pd
import numpy as np
import re
import io
import os
from tqdm.notebook import tqdm
tqdm.pandas()
from torch.utils.data import Dataset, DataLoader
from ml_things import fix_text
from sklearn.metrics import classification_report, accuracy_score

data_path ='./archive/라벨링데이터/판결문'

# 분류 라벨
labels_ids = {'01_주장':0, '02_사실':1, '03_판단':2, '04_결론':3}

# How many labels are we using in training.
# This is used to decide size of classification head.
n_labels = len(labels_ids)
```

2. EDA 및 전처리

2.2 EDA 수행 및 데이터 전처리

- 사용하고자 하는 데이터 샘플을 불러와 살펴본다.

```python
import json

# 데이터 살펴보기
f_path ='./archive/라벨링데이터/판결문/01.민사/1981~2016/(전주)2012나186.json'

with open(f_path, "r", encoding='UTF8') as json_file:
    json_data  =  json.load(json_file)

for key in json_data:
    print(key)
    print(json_data[key])
```

0.0s Python

```
info
{'caseField': '1', 'detailField': '1', 'trailField': '2', 'caseNm': '손해배상(기)', 'courtNm': '광주고등법원', 'judmnAdjuDe': '2012. 12. 20.', 'caseNo': '(전주)2012나186', 'relateLaword': ['환경영향평가법 제22조', '환경영향평가법 시행령 제3
concerned
{'acusr': '2', 'dedat': '3'}
org
{'orgJdgmnCourtNm': '전주지방법원', 'orgJdgmnAdjuDe': '2011. 12. 9.', 'orgJdgmnCaseNo': '2011가합1452 '}
disposal
{'disposalform': '10', 'disposalcontent': ['제1심판결 중 피고 패소부분을 취소하고, 그 취소부분에 해당하는 원고의 청구를 기각한다.', '원고의 항소 및 당심에서 확장된 원고의 청구를 모두 기각한다.', '소송총비용은 원고가 부담한다.']}
mentionedItems
{'rqestObjet': ['피고는 원고에게 924,768,639원과 그 중 605,415,092원에 대하여는 2011. 3. 22.부터2011. 12. 9.까지는 연 5%의, 그 중 47,206,945원에 대하여는 2011. 10. 21.부터 이 사건 2012. 2. 3.자 청구취지 및 원인변경신청서 송달일까지는 연 5%의, 각
assrs
{'acusrAssrs': ['피고 소속 공무원인 완주군수는 전라북도행정심판위원회의 이 사건 1차 재결에도 불구하고 이 사건 채석장에 대한 토석채취 기간을임의대로 단축하여 이 사건 허가처분을 하였다.', '나아가 이 사건 2차 재결에 따라 이 사건 허가처분이 취소되어 이 사건
facts
{'bsisFacts': ["원고는 1995. 5. 3.경부터 석재 및 골재판매업 등을 해온 회사로서, 피고 소속 공무원인 완주군수로부터 1995. 8. 25. 0000 (주소 생략)(이하 '이 사건 채석장' 이라 한다)에 벌채구역 66,787㎡, 채석면적 42,337㎡, 토석채취량1,113,748㎥, 허가기간
dcss
{'courtDcss': ['재결의 기속력은 재결의 주문 및 그 전제가 된 요건사실의 인정과 판단, 즉 처분 등의 구체적 위법사유에 관한 판단에만 미친다고 할 것이다(대법원 1998. 2. 27. 선고 96누13972 판결, 2001. 3. 23. 선고 99두5238 판결 등 참조).', '종전 처분이 재
close
{'cnclsns': ['제1심 판결 중 피고 패소부분을 취소하고, 그 취소부분에 해당하는 원고의 청구를 기각한다.', '원고의 항소 및 당심에서 확장된 원고의 청구를 모두 기각한다. ', '소송총비용은 원고가 부담한다.']}
```

- json 확장자 파일에 9개의 key (info, concerned, org, disposal, mentionedItems, assrs, facts, dcss, close) 와 각 key에 해당하는 내용들이 포함되어 있다.

2. EDA 및 전처리

2.2 EDA 수행 및 데이터 전처리

1. 읽어온 JSON 객체에 특정 **key의 존재 여부를 확인**하기위한 **is_json_key_present 함수를 정의**한다.

2. 설정한 경로에서 **데이터를 불러와 딕셔너리로 반환하는 JsonDataset 클래스를 정의**한다.

```python
# JSON 객체에서 특정 키(key)가 존재하는지 여부를 확인
def is_json_key_present(json, key):
    try:
        buf = json[key]
    except KeyError:
        return False
    return True
```

```python
class JsonDataset(Dataset):


  def __init__(self, path):

    # Check if path exists.
    if not os.path.isdir(path):
      # Raise error if path is invalid.
      raise ValueError('Invalid `path` variable! Needs to be a directory')
    self.texts = []
    self.labels = []


    for label in ['01.민사', '02.형사', '03.행정']:
      sentiment_path = os.path.join(path, label)

      # Get all files from path.
      for (root, directories, files) in os.walk(sentiment_path):
      # Go through each file and read its content.
        for file_name in tqdm(files, desc=f'{root} files'):
          file_path = os.path.join(root, file_name)
          with open(file_path, "r") as json_file:
            json_data  =  json.load(json_file)
            if(is_json_key_present(json_data,"assrs") and is_json_key_present(json_data['assrs'],"acusrAssrs")) :
              for data in json_data['assrs']['acusrAssrs'] :
                self.texts.append(fix_text(data))
                self.labels.append('01_주장')

            if(is_json_key_present(json_data,"assrs") and is_json_key_present(json_data['assrs'],"dedatAssrs")) :
              for data in json_data['assrs']['dedatAssrs'] :
                self.texts.append(fix_text(data))
                self.labels.append('01_주장')

            if(is_json_key_present(json_data,"facts") and is_json_key_present(json_data['facts'],"bsisFacts")) :
              for data in json_data['facts']['bsisFacts'] :
                self.texts.append(fix_text(data))
                self.labels.append('02_사실')

            if(is_json_key_present(json_data,"dcss") and is_json_key_present(json_data['dcss'],"courtDcss")) :
              for data in json_data['dcss']['courtDcss'] :
                self.texts.append(fix_text(data))
                self.labels.append('03_판단')

            if(is_json_key_present(json_data,"close") and is_json_key_present(json_data['close'],"cnclsns")) :
              for data in json_data['close']['cnclsns'] :

                self.texts.append(fix_text(data))
                self.labels.append('04_결론')
    # Number of exmaples.
    self.n_examples = len(self.labels)
    return

  def __len__(self):
    return self.n_examples

  def __getitem__(self, item):
    return {'text':self.texts[item],
            'label':self.labels[item]}
```

2. EDA 및 전처리

2.2 EDA 수행 및 데이터 전처리

1. 사전 정의한 JSONdataset 클래스로 데이터를 불러온다.

2. 불러온 데이터에 **pd.DataFrame.from_records**를 적용해 데이터프레임으로 만든다.

3. train_test_split()을 통해 데이터프레임을 **9:1의 비율로** 나눈다.

4. label 별로 나누어져 있는 데이터프레임을 합쳐 훈련 데이터와 테스트 데이터를 만든다.

※ GridSearchCV의 cv argument를 통해 validation이 가능하므로 train, test로만 분리한다.

```python
import pandas as pd
from sklearn.model_selection import train_test_split

# JSON Data
dataset = JsonDataset(path=data_path)
df = pd.DataFrame.from_records(dataset)
df.groupby('label').size()

df = df.sample(frac=0.05, random_state=123)

df1 = df[df['label'].str.startswith('01')]
df2 = df[df['label'].str.startswith('02')]
df3 = df[df['label'].str.startswith('03')]
df4 = df[df['label'].str.startswith('04')]

# 전체 데이터에서 train, test 분리 (9 : 1)
df1_train, df1_test = train_test_split(df1, test_size=0.1, random_state=123)
df2_train, df2_test = train_test_split(df2, test_size=0.1, random_state=123)
df3_train, df3_test = train_test_split(df3, test_size=0.1, random_state=123)
df4_train, df4_test = train_test_split(df4, test_size=0.1, random_state=123)

df_train = pd.concat([df1_train,df2_train,df3_train,df4_train])
df_test = pd.concat([df1_test,df2_test,df3_test,df4_test])
```

```
./archive/라벨링데이터/판결문\01.민사 files:  0/0 [00:00<?, ?it/s]
./archive/라벨링데이터/판결문\01.민사\1981~2016 files: 100% 3257/3257 [00:25<00:00, 136.38it/s]
./archive/라벨링데이터/판결문\01.민사\2017 files: 100% 225/225 [00:01<00:00, 125.66it/s]
./archive/라벨링데이터/판결문\01.민사\2018 files: 100% 154/154 [00:01<00:00, 121.47it/s]
...
./archive/라벨링데이터/판결문\03.행정\2019 files: 100% 75/75 [00:00<00:00, 144.30it/s]
./archive/라벨링데이터/판결문\03.행정\2020 files: 100% 42/42 [00:00<00:00, 129.86it/s]
./archive/라벨링데이터/판결문\03.행정\2021 files: 100% 108/108 [00:00<00:00, 140.48it/s]
```