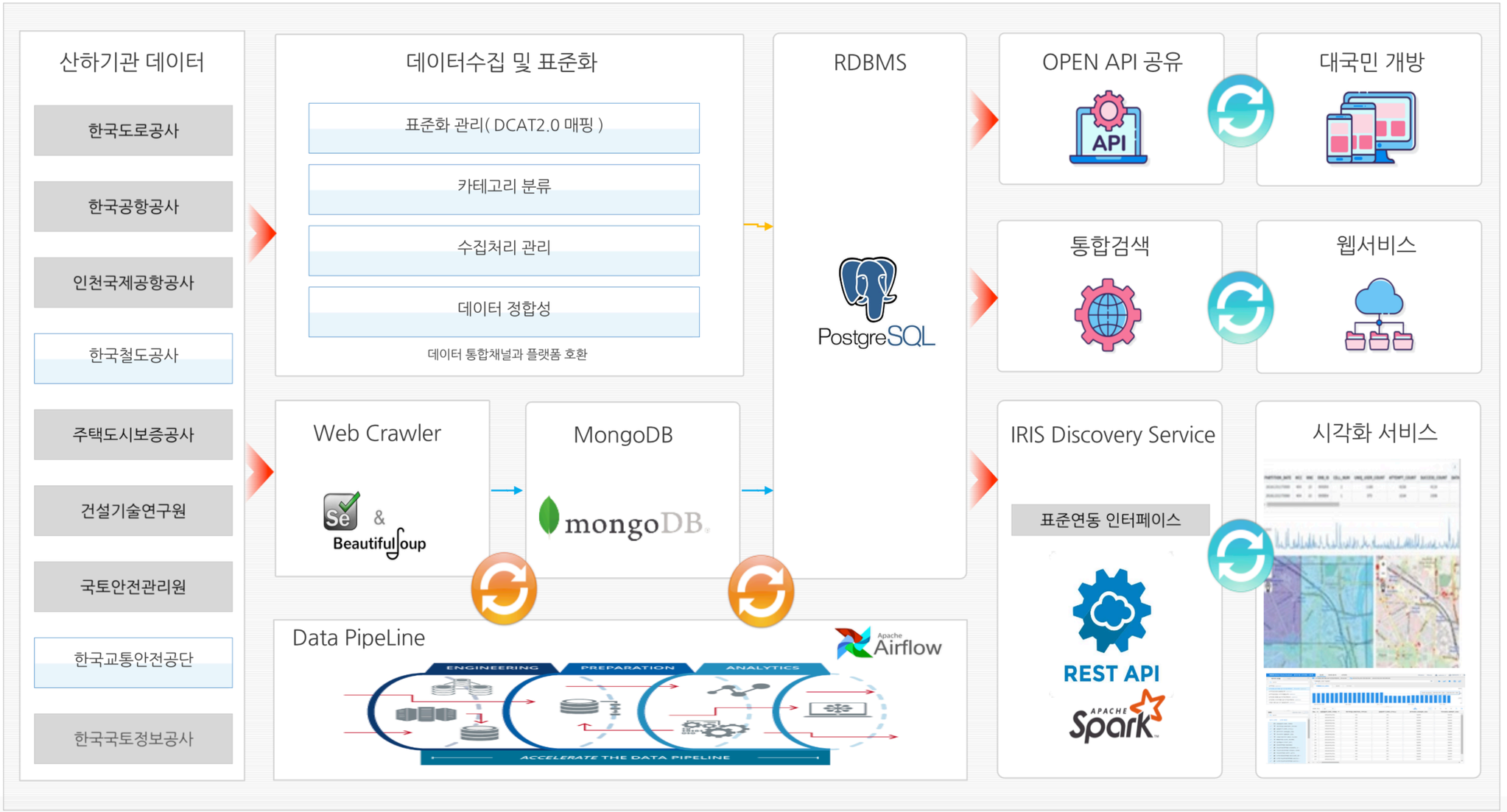
산업인공지능 개론

Miniproject 보고서 *#2*

학과 : 산업인공지능학과
학번 : 2024254022
이름 : 정현일

2024.05.02.

데이터 통합채널 공공데이터 수집 구성도



https://data.molit.go.kr/

Biz 메타 데이터 설명

국토교통 데이터는 **국토도시, 주택토지, 건설, 교통물류, 항공, 도로철도, 일반, 융합(기타)** *8*개 분야로 데이터를 관리 하며, 신규 *Biz* 메타데이터 등록 시에 각 데이터를 *8*개 분야 중 적절한 카테고리로 분류해야 합니다. 이를 통해 여러 분야에 데이터를 효과적으로 접근할 수 있도록 돕는 역할을 합니다.

	카테고리	데이터명	설명	키워드
0	교통물류	서울시 역외부코드로 지하철 막차시간표 정보 검색	역외부코드로 막차시간표 정보를 검색할 수 있도록 하는 API입니다._x000D_(역...	하행선,상행선,막차시간,종착역,지하철역,막차.시간표,외부코드,교통
1	교통물류	고속도로 주요 지점별 기상실황 데이터	고속도로 노선 주요 지점(93개)과 기상청 데이터 상품을 융·복합하여 일기 내용 및 ...	온도,습도,시정,기온,실황,풍속,운형,일기,강수,불쾌지수,운량,기상,적설량,풍향
3	교통물류	전국 고속도로 공사(작업) 계획 및 상태 데이터(1일 단위)	고속도로 작업에 대한 위치, 교통노선, 구간, 연장거리, 공사일자 및 기간, 공사구...	연장거리,교통노선,고속도로구간,부서,사업진행,공사기간,공사구간,사업관리,사업위치,이...
4	국토도시	서울시 부동산 실거래가 정보	서울 부동산 정보광장에서 제공하는 서울특별시 부동산 실거래가 정보입니다. 자치구, ...	건물,아파트,실거래,부동산,부동산,실거래가
5	교통물류	국토교통부 정보시스템 가이드북	국토교통 관련 정보시스템에 대한 구축 데이터 현황, 주요기능, 이용방법 등 가이드 ...	매뉴얼,프로그램,안내서
...	...	...	...	...
994	국토도시	국토교통부 공동주택 단지 관리비 정보	(공동주택관리정보)공동주택 관리비 투명성 확보를 위해 공동주택 관리주체가 관리비 내...	K-APT,관리비,공동주택
996	국토도시	국가철도공단 대구도시철도공사 호차별교통약자정보	대구도시철도공사 에서 관리하는 대구소재 역들의 호차별교통약자정보에 대한 철도운영기관...	대구도시광역철도,호차별교통약자정보,역사기본정보
997	주택토지	국토교통부 국토지리정보원 구지형도 시군구코드	국토지리정보원의 구지형도 관련 메타데이터 중 시군구 코드정보입니다. (시도코드, 시...	시군구,구지도,지리원
998	주택토지	국토교통부 국토지리정보원 구지형도 읍면동코드	국토지리정보원의 구지형도 관련 메타데이터 중 읍면동코드 입니다. (시군구코드, 읍면...	읍면동정보,구지도,지리
999	주택토지	국토교통부 국토지리정보원 국토정보 발급 신청상세	국토지리정보원에서 대민서비스인 국토정보플랫폼 측량기준점의 발급 신청 상세 정보를 제...	측량기준점,신청상세,발급신청

모델 설명 - DecisionTreeClassifier (1/2)

데이터 로드

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

import platform
system_name = platform.system()

excel_path = 'molit.xlsx'

# Load the Excel file
df = pd.read_excel(excel_path)

# Display the first few rows of the dataframe
print(df.head())
```

```
   resourceseq       분류   카테고리                                                데이터명   \
0         2940  개방 데이터   국토도시                                국토교통부_공동주택 수의계약 공지 정보제공 서비스
1         3043  개방 데이터   국토도시   제주국제자유도시개발센터_JDC지정면세점_입점업체 정보(2002-2015년)
2         3353  개방 데이터     일반                                한국부동산원_주간아파트동향 조회 서비스
3         3355  개방 데이터   교통물류     국토교통부_공공용지 취득실적-취득방법별 면적 및 보상액 기준 실적 현황
4         3356  개방 데이터   교통물류     국토교통부_공공용지 취득실적-기관별 취득면적 및 보상액 기준 실적 현황

                                                   설명                    키워드   \
0  (공동주택 수의계약정보)공동주택관리시스템에 축적된 공동주택별 공동주택위탁관리, 공사...      수의계약,K-APT,공동주택
1                          JDC지정면세점_입점업체 정보(2002-2015년)      지정면세점,입점 정보,JDC
2  ※ 한국부동산원에서는 '부동산통계정보시스템(R-ONE)' 고도화로 인해 해당 API...   가격동향,주간아파트동향,주간아파트
3  기준년도, 전년도, 전전년도(총계 금액, 총계 면적, 협의취득 금액, 협의취득 면적...    취득실적,공공용지,취득방법별면적
4  소관부처명, 사업기관조직코드, 토지면적(기준년도, 전년도, 전전년도), 토지금액(기...    취득실적,보상액기준실적,공공용지
```

텍스트 데이터 전처리

```
columns_to_keep = ['카테고리', '데이터명', '설명', '키워드']
df = df[columns_to_keep]

#결측치 삭제
df = df.dropna(subset=['카테고리', '데이터명', '설명', '키워드'])

#데이터 치환 '_' -> ' '
df['데이터명'] = df['데이터명'].str.replace('_', ' ', regex=False)

# Convert '설명' non-string data
df['데이터명'] = df['데이터명'].astype(str)
```

텍스트 데이터 벡터화

```
from sklearn.feature_extraction.text import CountVectorizer

cvect = CountVectorizer()
dtm = cvect.fit_transform(df["설명"])
#dtm
```

DTM(document-term matrix)

```
# 피처 가져오기
feature_names = cvect.get_feature_names_out()
#feature_names
```

```
# dtm(document-term matrix)
cvect.vocabulary_

# vocabulary_ 속성을 가져옴
vocab_dict = cvect.vocabulary_

# 딕셔너리를 데이터프레임으로 변환
vocab_df = pd.DataFrame(list(vocab_dict.items()), columns=['Word', 'Index'])
```

데이터 준비

```
# 독립변수로 사용할 X 변수에 dtm array 를 할당

X = dtm.toarray()
X.shape
```

```
(22568, 55704)
```

```
# 종속변수로 사용할 y 변수에 레이블값 "카테고리" 값을 할당

y = df["카테고리"]
y
```

```
0          국토도시
1          국토도시
2            일반
3          교통물류
4          교통물류
           ...
23674      국토도시
23675      국토도시
23676      국토도시
23677      국토도시
23678      국토도시
Name: 카테고리, Length: 22568, dtype: object
```

모델 설명 - *DecisionTreeClassifier (2/2)*

데이터 나누기 (train_test_split)

```
split_count = int(df.shape[0] * 0.8)
#split_count
```

```
# train set 준비
X_train = X[:split_count]
y_train = y[:split_count]

print(X_train.shape, y_train.shape)
#X_train, y_train
```

```
(18054, 55704) (18054,)
```

```
# test set 준비
X_test = X[split_count:]
y_test = y[split_count:]

print(X_test.shape, y_test.shape)
#X_test, y_test
```

```
(4514, 55704) (4514,)
```

머신러닝 모델 로드하기(DecisionTree)

```
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()
model
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

모델 학습 fit(X_train, y_train)

```
# fitting
model.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

예측 predict(X_test)

```
# predict
y_predict = model.predict(X_test)
y_predict.shape
```

```
(4514,)
```

평가 evaluate(y_test, y_predict)

```
from sklearn.metrics import accuracy_score

accuracy_score(y_test, y_predict)
```

```
0.7053610988037218
```

* 정확도 약 70%

```
# confusion matrix
pd.crosstab(y_test, y_predict)
```

col_0 카테고리	건설	교통물류	국토도시	도로철도	융합(기타)	일반	주택토지	항공
건설	21	3	134	0	0	0	0	0
교통물류	22	110	278	10	1	158	3	10
국토도시	28	124	2882	41	0	160	14	29
도로철도	7	20	51	69	0	9	1	1
융합(기타)	8	4	32	1	3	6	2	2
일반	1	5	22	1	0	10	9	1
주택토지	7	2	20	4	0	83	77	2
항공	10	1	2	1	0	0	0	12

모델 설명 - *LSTM (1/3)*

라이브러리 import

```
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

데이터 준비

```
# 엑셀 파일 로딩
file_path = 'molit.xlsx'
data = pd.read_excel(file_path)

# 설명 컬럼과 레이블 컬럼 선택
texts = data['설명'].astype(str)  # 설명 컬럼
labels = data['카테고리']  # 카테고리 레이블

texts, labels
```

```
(0        (공동주택 수의계약정보)공동주택관리시스템에 축적된 공동주택별 공동주택위탁관리, 공사...
 1                              JDC지정면세점_입점업체 정보(2002-2015년)
 2        ※ 한국부동산원에서는 '부동산통계정보시스템(R-ONE)' 고도화로 인해 해당 API...
 3        기준년도, 전년도, 전전년도(총계 금액, 총계 면적, 협의취득 금액, 협의취득 면적...
 4        소관부처명, 사업기관조직코드, 토지면적(기준년도, 전년도, 전전년도), 토지금액(기...
                                ...
 23674                  통계청(KOSIS)에서 제공하는 교통문화지수(시도/시/군/구)
 23675        통계청(KOSIS)에서 제공하는 자동차 천대당 교통사고발생건수(시도/시/군/구)
 23676            통계청(KOSIS)에서 제공하는 1인당 자동차 등록대수(시도/시/군/구)
 23677                    통계청(KOSIS)에서 제공하는 지가변동률(시도/시/군/구)
 23678                       일간 수집된 교통 데이터의 노선, 정류장 별 이용통계
 Name: 설명, Length: 23679, dtype: object,
 0        국토도시
 1        국토도시
 2          일반
 3        교통물류
 4        교통물류
          ...
 23674    국토도시
 23675    국토도시
 23676    국토도시
 23677    국토도시
 23678    국토도시
 Name: 카테고리, Length: 23679, dtype: object)
```

텍스트 토큰, 레이블 -> 훈련, 테스트 데이터 분리

```
# 텍스트 토큰화 및 시퀀스 변환
tokenizer = Tokenizer(num_words=10000)
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
padded_sequences = pad_sequences(sequences, maxlen=100)

# 레이블 인코딩
encoder = LabelEncoder()
encoded_labels = encoder.fit_transform(labels)

# 훈련 데이터와 테스트 데이터 분리
X_train, X_test, y_train, y_test = \
    train_test_split(padded_sequences, encoded_labels, test_size=0.2, random_state=42)
```

모델 구축

```
model = Sequential([
    Embedding(10000, 128),
    LSTM(64),
    Dense(len(encoder.classes_), activation='softmax')
])

model
```

```
<Sequential name=sequential, built=False>
```

모델 컴파일

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

model
```

```
<Sequential name=sequential, built=False>
```

모델 설명 - *LSTM (2/3)*

모델 훈련

```
history = model.fit(X_train, y_train, batch_size=64, epochs=12, validation_data=(X_test, y_test))
```

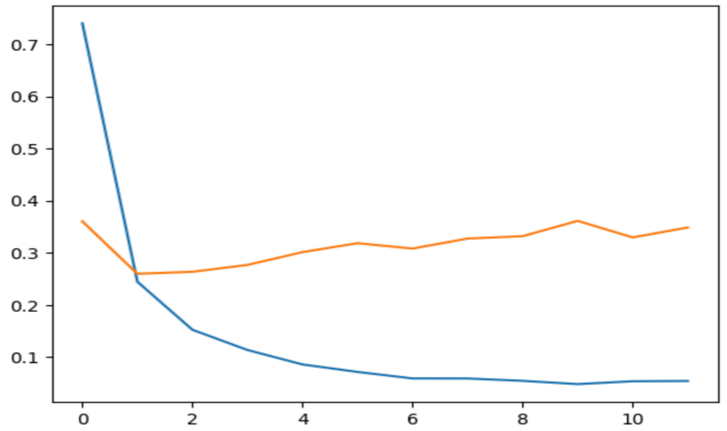
```
Epoch 1/12
296/296 ━━━━━━━━━━━━━━━━━━━━ 14s 46ms/step - accuracy: 0.6532 - loss: 1.1154 - val_accuracy: 0.8887 - val_loss: 0.3603
Epoch 2/12
296/296 ━━━━━━━━━━━━━━━━━━━━ 14s 47ms/step - accuracy: 0.9206 - loss: 0.2591 - val_accuracy: 0.9193 - val_loss: 0.2597
Epoch 3/12
296/296 ━━━━━━━━━━━━━━━━━━━━ 14s 47ms/step - accuracy: 0.9545 - loss: 0.1499 - val_accuracy: 0.9221 - val_loss: 0.2635
Epoch 4/12
296/296 ━━━━━━━━━━━━━━━━━━━━ 14s 48ms/step - accuracy: 0.9677 - loss: 0.1106 - val_accuracy: 0.9221 - val_loss: 0.2767
Epoch 5/12
296/296 ━━━━━━━━━━━━━━━━━━━━ 14s 48ms/step - accuracy: 0.9749 - loss: 0.0851 - val_accuracy: 0.9255 - val_loss: 0.3013
Epoch 6/12
296/296 ━━━━━━━━━━━━━━━━━━━━ 14s 48ms/step - accuracy: 0.9803 - loss: 0.0670 - val_accuracy: 0.9250 - val_loss: 0.3184
Epoch 7/12
296/296 ━━━━━━━━━━━━━━━━━━━━ 14s 47ms/step - accuracy: 0.9843 - loss: 0.0526 - val_accuracy: 0.9272 - val_loss: 0.3081
Epoch 8/12
296/296 ━━━━━━━━━━━━━━━━━━━━ 14s 49ms/step - accuracy: 0.9838 - loss: 0.0544 - val_accuracy: 0.9257 - val_loss: 0.3272
Epoch 9/12
296/296 ━━━━━━━━━━━━━━━━━━━━ 14s 46ms/step - accuracy: 0.9843 - loss: 0.0513 - val_accuracy: 0.9274 - val_loss: 0.3318
Epoch 10/12
296/296 ━━━━━━━━━━━━━━━━━━━━ 15s 51ms/step - accuracy: 0.9841 - loss: 0.0479 - val_accuracy: 0.9274 - val_loss: 0.3612
Epoch 11/12
296/296 ━━━━━━━━━━━━━━━━━━━━ 15s 50ms/step - accuracy: 0.9825 - loss: 0.0486 - val_accuracy: 0.9236 - val_loss: 0.3295
Epoch 12/12
296/296 ━━━━━━━━━━━━━━━━━━━━ 15s 49ms/step - accuracy: 0.9848 - loss: 0.0464 - val_accuracy: 0.9248 - val_loss: 0.3484
```

loss, val_loss

```
import matplotlib.pyplot as plt

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

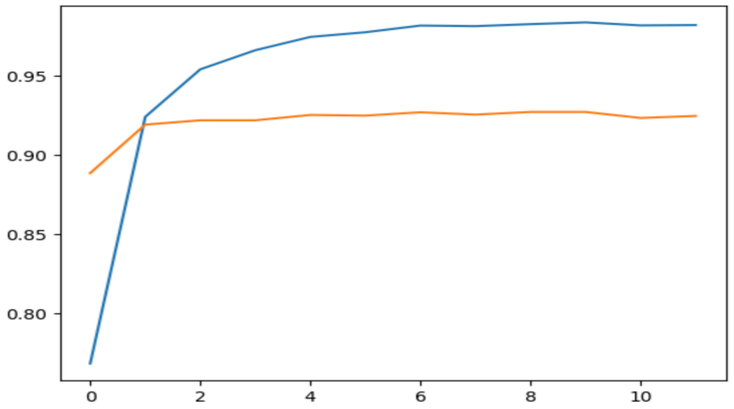
```
[<matplotlib.lines.Line2D at 0x17af37250>]
```



accuracy, val_accuracy

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
```

```
[<matplotlib.lines.Line2D at 0x17aff0750>]
```



모델 설명 - *LSTM (3/3)*

- 예측

```
predictions = model.predict(X_test)
predicted_categories = encoder.inverse_transform(predictions.argmax(axis=1))

predictions, predicted_categories
```

```
148/148 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step
(array([[1.87762691e-06, 5.50771947e-05, 9.99677956e-01, ...,
         4.28951216e-06, 2.40883528e-05, 1.46418614e-07],
        [2.41562020e-06, 2.03292484e-06, 9.99961078e-01, ...,
         2.64458254e-06, 1.38659680e-05, 1.87325114e-07],
        [7.15864280e-06, 1.06629186e-05, 4.59552075e-06, ...,
         7.73096474e-07, 8.87569513e-06, 4.97943574e-06],
        ...,
        [5.14633238e-01, 3.96769308e-03, 1.23587875e-02, ...,
         3.20646018e-02, 7.55911227e-03, 1.15594314e-03],
        [5.12791644e-07, 3.09317534e-06, 9.99986887e-01, ...,
         2.79669166e-07, 5.54744793e-06, 5.27176347e-08],
        [1.50008333e-07, 4.58101204e-06, 9.99987483e-01, ...,
         1.56270445e-07, 5.10200516e-06, 4.81026312e-08]], dtype=float32),
 array(['국토도시', '국토도시', '도로철도', ..., '건설', '국토도시', '국토도시'], dtype=object))
```

성능 평가 - *LSTM*

모델 평가

```
# 모델 평가
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy * 100:.2f}%")
```

```
148/148 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.9167 - loss: 0.3892
Test Accuracy: 92.48%
```

정확도 약 92.5%

예측 결과 분류 보고서

```
from sklearn.metrics import classification_report, confusion_matrix

# 예측
#predictions = model.predict(X_test)
predicted_classes = predictions.argmax(axis=1)

# 인코더의 classes_ 속성을 사용하여 실제 클래스 이름을 가져옵니다.
class_labels = encoder.classes_

# 분류 보고서
print(classification_report(y_test, predicted_classes, target_names=class_labels))
pd.crosstab(y_test, predicted_categories)
```

```
              precision    recall  f1-score   support

          건설       0.90      0.89      0.89       206
        교통물류       0.91      0.85      0.88       673
        국토도시       0.96      0.97      0.96      2443
        도로철도       0.92      0.90      0.91       650
      융합(기타)       0.50      0.62      0.56        32
          일반       0.85      0.87      0.86       333
        주택토지       0.89      0.92      0.90       353
          항공       0.69      0.78      0.73        46

    accuracy                           0.92      4736
   macro avg       0.83      0.85      0.84      4736
weighted avg       0.93      0.92      0.93      4736
```

col_0 / row_0	건설	교통물류	국토도시	도로철도	융합(기타)	일반	주택토지	항공
0	183	4	10	3	1	3	2	0
1	2	574	29	33	4	19	4	8
2	5	24	2363	10	6	15	18	2
3	6	23	18	588	2	7	3	3
4	0	0	9	0	20	2	1	0
5	5	2	18	2	3	291	10	2
6	3	0	12	4	2	6	325	1
7	0	1	3	1	2	0	3	36

프로젝트 수행 후 경험

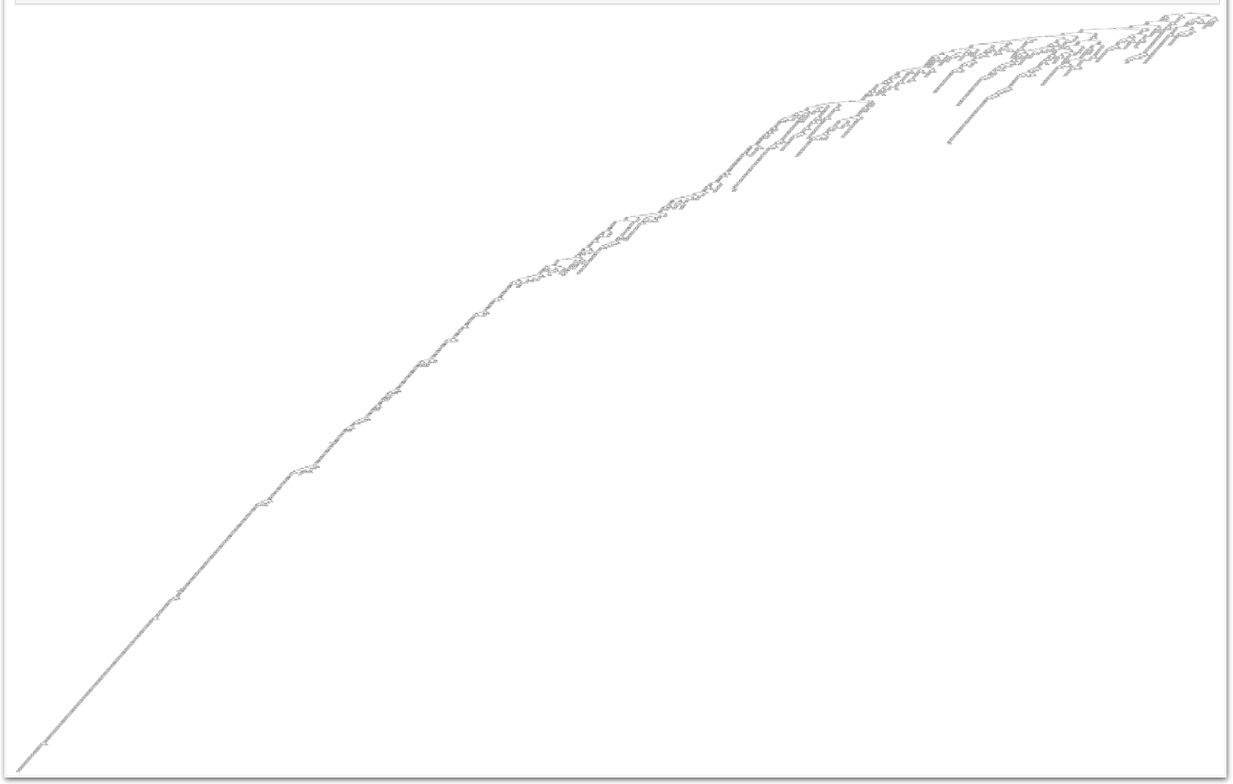
DecisionTree 모델

의사결정 나무*(Decision Tree)* 알고리즘을 사용하는 분류 모델로, 의사결정 나무는 데이터를 트리 구조로 나타내어 각 노드에서 특정 특성*(feature)*에 대한 조건을 검사하여 분류하는 방법을 사용하는데, 데이터 균형이 맞지 않아 좌편향 트리로 모델이 생성되고, 국토교통 분야 *Biz* 메타 데이터 분류에는 적합하기 않은 것으로 판단되고 좀 더 시간을 들여서 데이터 정재 작업이 필요하다고 느꼈습니다.

```
# plot_tree 로 시각화 하기
from sklearn.tree import plot_tree

plt.figure(figsize=(200,100))  # Set the figure size (width, height) in inches

plot_tree(model, feature_names=feature_names, rounded=True)
plt.show()
```



RNN - LSTM 모델

국토도시 클래스의 정밀도*(0.96),* 재현율*(0.97), F1* 점수*(0.96)*가 가장 높아, 모델이 이 클래스에 대해 매우 정확하게 분류 함.
융합*(기타)*와 항공 클래스의 *F1* 점수가 상대적으로 낮은데, 이는 이 클래스의 데이터 양이 적어 모델이 제대로 학습하지 못한것 같습니다.
모델은 국토도시, 교통물류, 건설과 같은 클래스에서 매우 좋은 성능을 보이고 있지만, 데이터가 상대적으로 적은 융합*(기타)*와 항공에서는 성능이 저하되는 경향을 보임.

	precision	recall	f1-score	support
건설	0.90	0.89	0.89	206
교통물류	0.91	0.85	0.88	673
국토도시	0.96	0.97	0.96	2443
도로철도	0.92	0.90	0.91	650
융합(기타)	0.50	0.62	0.56	32
일반	0.85	0.87	0.86	333
주택토지	0.89	0.92	0.90	353
항공	0.69	0.78	0.73	46
accuracy			0.92	4736
macro avg	0.83	0.85	0.84	4736
weighted avg	0.93	0.92	0.93	4736

총평

국토교통 통합채널에서 서비스하고 있는 *Biz* 메타 데이터를 활용해서 다중분류 문제를 *ML*의 *Decision Tree*와 *RNN LSTM* 모델 *2*개로 다중 분류 문제를 해결해 봤습니다. *Decision Tree* 분류 예측 정확도 약 *70%* 성능을 보였으며, *RNN LSTM*모델은 약 *92.5%* 정확도를 나타냈습니다.
이 결과는 *RNN* 기반의 모델이 국토교통 *BIZ* 메타 데이터 패턴을 더 잘 포착할 수 있음을 보여주고 있습니다.
앞으로 메터데이터의 불용어 제거와 형태소 분석을 통해 더 정제된 데이터를 이용한 학습이 성능 향상에 기여할 수 있을 같고,
언어 데이터를 다루는 작업에서는 데이터의 품질이 중요하다고 느꼈으며, 정제된 데이터가 모델이 더 나은 패턴을 학습할 수 있도록 도움을줄 것으로 예상해 봅니다.