AI Challenge for Biodiversity

분석보고서

2022. 10. 31

김동하, 김서영, 박상진, 백지헌, 이주형
(BCCDS)

1. 분석 주제명

기후 변화에 따른 생물 다양성에 이상적인 환경 예측

2. 배경**:** 분석 방법론 간략 설명

현재 한국의 환경은 과거와 다르게 많은 점이 변화한 것을 쉽게 체감할 수 있다. 그 예로 올해에는 '장마'라는 단어가 아닌 '우기'라는 명칭을 사용하자는 회의가 있었다. 또한, 한 기사에 따르면 지난 30년간 벚꽃과 개나리 등 봄꽃의 개화 시기가 10년 전 과거보다 최소 2일에서 최대 20일까지 빨라졌다고 서술한다.[1] 이는 계속해서 증가하는 평균 기온, 에어로졸 등의 기후 변화가 영향을 미치는 것으로 알려져 있고 점점 더 가속화되고 있다. 이러한 현상은 단순히 꽃이 늦게 폈다고만 받아들여서는 안 된다. 과거 우리나라에서 재배할 수 없었던 식물이 이제는 기후 영향으로 인해 재배할 수 있는 부분, 동해안에서 어획되는 어류가 변화한 점, 철새였던 종이 텃새화가 되어 계속 관측이 가능한 점 등 기후 변화에 따른 급격하게 변화하는 생물 다양성의 심각성을 인지해야 한다. 또한, 우리나라는 사계절이 뚜렷한 지역이었지만 이제는 그렇다고 말할 수 없다. 따라서 앞서 언급한 예시처럼 기후 변화가 생물다양성에 어떤 영향을 미치는지, 더 나아가 예측된 미래 기후로 앞으로 탄소 중립 시행과 같은 노력을 했을 때 원래의 안정적인 생물다양성이 보장되는 기후가 됐다고 말할 수 있는 그 분기점을 예측해보려 한다.

우선 지금까지 평균기온, 강수량, 이산화탄소량 그리고 풍속 등 쉽게 관측할 수 있는 지표를 구하여 1990년대 이전 날짜에는 레이블 0을, 2017년 이후에는 레이블 1을 부여했다. 위 기사에 따라 30년 전의 기후 즉, 생물 다양성이 크게 변화하지 않은 환경은 0, 생물다양성이 변한 현재의 기후인 2017년 이후는 1을 부여한 것이다.

그다음, 레이블이 없는 1990년부터 2017년 사이의 데이터는 Semi-supervised Learning 중 하나인 LabelSpreading을 사용하여 레이블이 있는 데이터로부터 예측 및 생성하여 레이블을 부여했다. 그 후 Binary Classification 모델을 생성하여 앞서 언급한 생물다양성이 보장되는 기후가 되는 그 분기점을 예측하여 찾았다.

3. 구현 방법

분석하기 위해 수집한 데이터 중 최근 일자의 데이터는 존재하여 실제 값을 사용했지만, 과거 1990년 이전 데이터는 찾을 수 없어서 결측치가 발생했다. 모델의 이상적인 학습을 위해서는 결측치를 처리해야만 했는데, 이 과정에서 값을 단순히 대치하면 날짜, 월 및 계절성을 띠는 패턴이 생겨서 모델에 악영향을 미칠 것으로 판단하였다. 이에, 월별 평균을 구하여 해당 값을 대치했다.

---

[1] "지난 30년간 벚꽃 등 봄꽃 개화, 이전 10년에 비해 최소 2일에서 최대 20일까지 빨라져", 에너지단열경제, 2021-03-23, http://kienews.com/news/newsview.php?ncode=1065597829283508

```
mean_li = list() # 데이터로 월별 평균을 구하여 리스트에 저장

for i in range(1, 13, 1):
    # 1월부터 12월까지 반복
    month_dict = dict()
    query_month = i
    filtered3 = climate_df.query('관측일자.dt.month == @query_month')
    # query구문을 통하여 월별로 필터링
    for col in filtered3:
        # 각 컬럼별 평균을 구해 월별 딕셔너리에 저장
        month_dict[col] = filtered3[col].mean()
    # 각 월별 딕셔너리를 평균리스트에 추가
    mean_li.append(month_dict)
```

월별로 데이터를 추출하여 평균을 구하고, 그 값을 리스트로 저장 후 결측치에 대입했다.

```
# 새로운 데이터 프레임을 추가
res = pd.DataFrame()
for i in range(1, 13, 1):
    # 1월부터 12월까지 반복
    query_month = i
    filtered = climate_df.query('관측일자.dt.month == @query_month')
    for col in filtered:
        # 결측치를 평균 값으로 대치
        filtered[col].fillna(mean_li[i-1][col], inplace = True)
    # 새로운 데이터 프레임에 병합하면서 모든 데이터를 재구성
    res = pd.concat([res, filtered])
```

또한, 분석에 사용한 데이터가 1990년부터 2018년까지의 데이터이므로 90년 이전은 0, 2017년 이후는 1로 임의로 지정했다. 여기서 0은 생물 다양성이 원활하게 이뤄지는 환경을 의미한다. 전문가의 의견이나 여러 매체 등을 통해 알 수 있듯이, 현재 우리나라의 기후 환경은 과거와 크게 달라졌다. 따라서 현재보다 과거의 환경이 생물 다양성 유지 및 증진에 더 이상적이므로 과거의 기후환경을 0으로, 변화된 현재의 기후 환경을 1로 레이블을 지정했다.

```
# 라벨링을 할 칼럼을 Na 값으로 초기화화
res["label"] = np.NaN
# 90년대 이전은 0, 17년도 이후는 1로 부여
series_1 = res[res["관측일자"] <= "1990"]["label"].apply(lambda x: 0)
series_2 = res[res["관측일자"] >= "2017"]["label"].apply(lambda x: 1)
label = pd.concat([series_1, series_2])
res["label"] = label
```

위와 같이 라벨링을 하게 되면 1990년대와 2017년 사이의 데이터는 라벨링이 되지 않아서 준지도 학습(Semi-supervised Learning) 기법의 하나인 LabelSpreading을 사용하여 레이블을 부여했다.

```
# GridSearch를 사용하여 LabelSpreading의 최적의 파라미터를 찾음
gr_label = GridSearchCV(LabelSpreading(), params, n_jobs = -1)
gr_label.fit(x_train, y_train)
```

GridSearch로 LabelSpreading의 최적 하이퍼파라미터를 찾아 라벨링이 되지 않은 데이터에 효과적으로 라벨링을 할 수 있었다.

```
roc score : 0.9275990099009901
```

테스트 데이터로 구한 roc score는 0.92로 확인했고 새롭게 레이블을 만든 데이터를 적용하여 모든 데이터에 레이블을 부여할 수 있었다.

모델링을 위해 사이킷런(scikit-learn)의 train_test_split 모듈로 train set과 test set으로 분리했다. 해당 모듈의 옵션인 stratify를 설정해 계층적 샘플링을 하였지만, 레이블이 불균형한 분포를 보이는 것을 확인할 수 있다.

```
1    y_train.value_counts()
```

```
1.00000    1042
0.00000     252
Name: label, dtype: int64
```

1인 레이블은 1,042개, 0인 레이블은 252개로 약 750개의 차이가 발생하여 불균형한 레이블의 개수를 맞춰주기 위해 Oversampling 기법의 하나인 SMOTE(Synthetic Minority Over-sampling Technique)를 사용했다.

현재 Microsoft Azure Machine Learning Studio 환경에서 작업 중 SMOTE의 imblearn 라이브러리가 버전 오류가 발생하였고 Workspace Terminal에서 라이브러리 재설치로 오류를 없앨 수 있었다.

```
SMOTE 적용 전 학습용 피처/레이블 데이터 세트 :  (1294, 9) (1294,)
SMOTE 적용 후 학습용 피처/레이블 데이터 세트 : (2084, 9) (2084,)
SMOTE 적용 후 값의 분포 :
 0.00000    1042
1.00000    1042
```

불균형한 라벨링의 분포가 해결되었다. 그 후 이 데이터를 표준화(Standard Scaling)로 특성 스케일링을 하고 train set과 validation set으로 분리했다.

RandomForest는 트리 계열 알고리즘으로 비모수적 모형으로 다음과 같은 특징이 있다. 첫째, 이상치에 크게 영향을 받지 않는다. 둘째, 스케일링이 따로 필요가 없다. 셋째, 결과 해석이 쉽다. 마지막으로 특성 중요도(feature importance)를 확인해 어떤 특성이 모델에 영향을 미쳤는지를 확인할 수 있다. 따라서 분류(classification) 모델은 RandomForeset 알고리즘을 사용했다. 물론 트리 계열 알고리즘에는 Decision Tree나 Extra Tree가 있지만 Decision Tree보다는 ensemble을 이용한 RandomForest 모델의 정확도가 높을 것으로 생각했고, 적은 데이터양과 bootstrap을 사용하지 않는 점으로 Extra Tree를 사용하지 않았다.

```
params = { 'n_estimators' : [10, 100],
          "min_impurity_decrease" : np.arange(0.0001, 0.001, 0.0001),
           'max_depth' : [6, 8, 10, 12],
           'min_samples_leaf' : [8, 12, 18],
           'min_samples_split' : [8, 16, 20]
            }
```

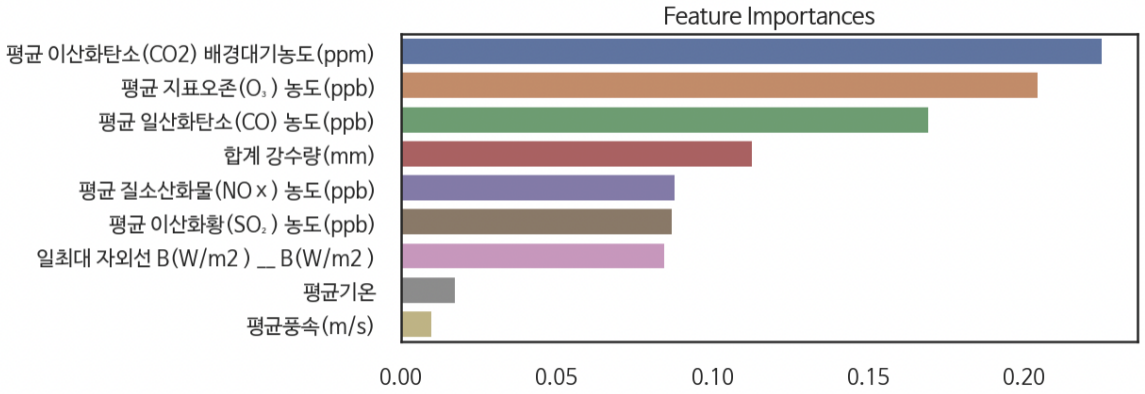
```
params = { 'n_estimators' : randint(10, 100),
           'max_depth' : randint(10, 100),
           'min_samples_leaf' : randint(1, 20),
           'min_samples_split' : randint(2, 20),
            "min_impurity_decrease" : uniform(0.0001, 0.001)
            }
```

위 코드는 각각 GridSearch와 RandomizedSearch에 사용한 파라미터다. 트리 계열 알고리즘은 하이퍼파라미터를 지정하지 않고 모델을 만들면 쉽게 오버피팅이 일어나므로 위와 같은 하이퍼파라미터 튜닝으로 최적의 값을 찾아 모델링을 진행했다.

다음은 최종적인 모델의 train set 평가 지표(classification report)로, accuracy가 0.97이고 f1-score 또한 상당히 높은 값이 나온 것을 확인할 수 있다.

	precision	recall	f1-score	support
0.0	0.95	0.87	0.91	63
1.0	0.97	0.99	0.98	261
accuracy			0.97	324
macro avg	0.96	0.93	0.94	324
weighted avg	0.97	0.97	0.97	324

다음은 최종 모델의 특성 중요도를 시각화한 것이다. 이 결과로 평균 이산화탄소(CO2) 배경대기농도(ppm)가 모델에 가장 큰 영향을 미쳤고 그다음으로 평균 지표 오존(O2) 농도(ppb), 평균 일산화탄소(CO) 농도(ppb) 순으로 영향을 미쳤으며, 반면 평균기온과 평균풍속(m/s)은 크게 영향을 미치지 않는 것을 확인할 수 있다.



4. 결론**:** 참가자의 방법이 왜 좋은 결과를 보였는지 **Insightful**한 해석

위 모델에 필요한 측정값들은 이산화탄소, 오존, 일산화탄소 등으로 우리가 쉽게 구할 수 있는 데이터이고 타기관에서 계속 기록하므로 언제든 현재 기후 환경이 생물 다양성에 적절한 환경인지 확인할 수 있다. 또한, 앞으로의 예측으로 생물다양성 유지 및 증진에 크게 도움이 되는 적절한 기후환경으로 되돌아오는 점 즉, 그 분기점을 구할 수도 있다는 점에 의의가 있다.

다음 데이터는 탄소 중립 시행을 계속 진행하면 변화될 기후를 임의로 예측한 것으로 2033년에 생물 다양성에 이상적인 기후로 돌아온다고 예측하였고, 그때의 지표를 통하여 현재 기후 환경을 2033년에 해당하는 환경으로 바꿀 수 있다면 생물 다양성의 증진이 다시 이루어질 것으로 예측할 수 있다.

9	2032.1.1	-5.00000	0.03200	2.10000	389.00000	36.00000	1.00000	3.70000	260.25000	nan	1.00000
10	2033.1.1	-5.00000	0.03200	2.10000	384.00000	36.00000	1.00000	3.70000	213.06667	nan	0.00000
11	2034.1.1	-5.00000	0.03200	2.10000	379.00000	36.00000	1.00000	3.70000	205.85000	nan	0.00000
12	2035.1.1	-6.45000	0.04050	2.60000	375.00000	34.70000	0.85000	3.50000	198.63333	nan	0.00000

5. 분석 환경 정보

- 사용 환경: ML Studio
- 로그인 정보:
- ID : SKH-ML-GRP003-USR001@junokoreashe.onmicrosoft.com/
- PW : bccdspasswd12@

6. 소스코드

- 파일명 : train_run.ipynb, predict_run.ipynb
- 파일 위치 : Users/SKH-ML-GRP003-USR001/final/src

7. 활용한 외부 데이터

- 조류분포, 등장 시기, 개체 수 데이터
    - 한국과학기술정보연구원_국내 조류분포 **('93~'00)**
        - **https://www.data.go.kr/data/3033734/fileData.do**
    - 국립생태원_제**4**차 전국자연환경조사 조류 조사보고서 **('14~'18)**
        - **https://www.data.go.kr/data/15106213/fileData.do**

- 기온, 자외선, 풍속, 오존, 이산화황, 일산화탄소, 강수량, 이산화탄소 데이터
    - 기상청 기상정보개방포털
        - **https://data.kma.go.kr/cmmn/main.do**

- 발전 방법별 발전량 데이터
    - 한국전력 **KEPCO** 전력통계월보
        - **https://home.kepco.co.kr/kepco/KO/ntcob/ntcobView.do?pageIndex=1&boardSeq=21059203&boardCd=BRD_000097**