



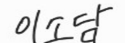
## 2024년 2학기 프로그램호환 프로젝트 최종보고서


### 계절 길이 변화와 극한 기온 발생 빈도 분석 및 시각화

팀명: 9조

팀장: 원 현 아 

팀원: 박 성 빈 

팀원: 이 소 담 

팀원: 임 하 은 

팀원: 최 원 준 

2024년 12월 20일

## 목 차

1. 프로젝트 개요.....	1p
2. 개발 환경 및 도구.....	3p
3. 프로젝트 설계 및 구현.....	5p
4. 팀워크 및 역할 분담.....	10p
5. 프로젝트 결과 및 평가.....	11p
6. 창의적 요소 및 개선 사항.....	12p
7. 결론.....	13p
8. 부록.....	15p

## 1. 프로젝트 개요

### 가. 프로젝트 목적

본 프로젝트의 목적은 기상 데이터를 활용하여 대한민국 기온 변화와 극단적 기상 현상을 분석하고 지역별 기후 변화를 파악할 수 있는 대시보드형 웹 사이트를 구현하는 것을 목표로 한다.

사용자가 원하는 지역의 실시간 기온 정보와 미래 기온 예측 데이터를 조회할 수 있도록 직관적이고 사용자 친화적인 인터페이스를 제공하며, 폭염·열대야·한파 등 주요 기상 현상을 전달할 수 있는 효과적으로 전달할 수 있는 시각적 요소를 포함한다.

### 나. 주제 선정 배경

#### (1) 기후 변화로 인한 계절 불균형의 심화

- 최근 기상청 자료에 따르면 지난 100년 동안 여름은 약 한 달 길어지고 겨울은 약 20일 짧아지는 등 계절 불균형이 심화되고 있다.
- 이러한 변화는 대한민국뿐만 아니라 전 세계적으로 관찰되고 있으며, 이는 지구 온난화와 탄소 배출 증가로 인한 영향으로 해석된다.

#### (2) 극단적 기상 현상의 빈도 및 강도 증가

- 2024년 대한민국의 여름철 평균기온은 25.6도(℃)로 평년보다 1.9도(℃) 높아 역대 최고를 기록했다.
- 열대야 일수는 20.2일로 관측 사상 가장 많은 날수를 기록하며 국민 건강, 경제적 활동, 일상생활에 큰 영향을 미쳤으며 이러한 변화는 단순히 날씨의 변동성을 넘어 구조적 기후 변화로 인식되고 있다.

#### (3) 기후 변화에 대한 데이터 기반 분석의 필요성

- 기상청 및 국제 기후 데이터(IPCC SSP 시나리오<sup>1)</sup>)를 활용하여 계절 길이 변화와 극단적 기상현상의 발생 빈도를 분석하고 시각화함으로써 기후 변화의 심각성을 전달하고자 한다.

### 다. 프로젝트 목표 및 기대 효과

#### (1) 프로젝트 목표

- 계절 변화와 극단적 기상 현상 분석
  - 기상청 기후 데이터(1951~2024)를 활용하여 지난 100년간 계절 길이 변화와 폭염, 열대야, 한파와 같은 극단적 기상 현상의 발생 추이를 분석한다.

1) IPCC 6차 보고서는 기후변화에 관한 정부 간 협의체(IPCC, Intergovernmental Panel on Climate Change)가 발간한 최신 보고서 시리즈로 기후 변화와 관련된 과학적 근거, 영향, 적응 및 완화 방안을 종합적으로 다룬다.

- 데이터 시각화를 통해 계절 불균형 심화와 극단적 기상 현상의 구조적 변화하는 경향을 파악한다.
- 미래 기후 변화 예측 및 시각화
  - 기상청 및 IPCC의 기후 데이터를 활용하여 대한민국 전역의 기온 변화와 극단적 기상 현상의 발생 빈도를 분석한다.
  - 시계열 예측 모델(SARIMAX)과 탄소 배출량 시나리오 SSP(Shared Socioeconomic Pathways)를 기반으로 미래 기온 변화를 예측하고 지역별로 시각화한다.
- 웹 기반 기후 대시보드 개발
  - OpenWeather API 및 OpenStreetMap을 활용하여 실시간 날씨 정보와 과거 기상 데이터를 시각화한다.
  - 사용자가 원하는 지역의 기온 변화, 폭염, 열대야, 한파 등 극단적 기상 현상 데이터를 조회할 수 있도록 다양한 상호작용 요소(드롭다운 메뉴, 지도 확대/축소)를 적용한다.

## (2) 기대 효과

- 기후 변화 대응 역량 강화
  - 기온 변화와 극단적 기상 현상에 대한 데이터를 분석하여 기후 변화의 영향을 사전에 예측하고 대처 방안을 마련할 수 있는 역량을 강화한다.
  - 기후 취약 지역의 데이터를 모니터링하고 극단적 기상 현상의 발생 빈도와 강도를 시각화하여 재난 대비 및 사전 대응 시스템 구축에 기여한다.
- 사회적, 경제적 비용 절감
  - 기후 변화로 인해 발생하는 자연재해와 그로 인한 피해를 사전에 예측하고 대비함으로써 사회적, 경제적 비용을 절감할 수 있다.
  - 지역 맞춤형 정책 수립과 자원 분배 최적화를 통해 효율적인 기후 변화 대응 전략을 마련한다.

## 2. 개발 환경 및 도구

### 가. 사용 언어 및 라이브러리

#### (1) 사용 언어

- ☐ Python: 데이터 처리, 분석 및 시각화를 위한 주요 프로그래밍 언어
- ☐ HTML/CSS/JavaScript: 웹 대시보드 개발 및 인터페이스 개발
- ☐ PHP: 웹 서버와 데이터 연동을 위한 백엔드 언어

#### (2) 라이브러리

- ☐ 데이터 분석 및 시각화
  - Pandas: 데이터 처리 및 분석
  - NumPy: 수치 연산 및 배열 처리
  - Matplotlib, Seaborn: 데이터 시각화를 위한 도구
  - Plotly, Dash: 대화형 웹 기반 시각화를 위한 라이브러리
- ☐ 예측 모델링 및 시계열 분석
  - statsmodels: 시계열 분석 및 SARIMAX 모델 구현
  - scikit-learn: 데이터 전처리 및 모델링
  - Plotly: 인터랙티브 데이터 시각화
- ☐ 지도 및 공간 데이터 시각화
  - GeoPandas: 공간 데이터 처리
  - Leaflet.js: 지도 데이터 시각화 및 상호작용 구현
  - Folium: 지도 기반 데이터 시각화
- ☐ API 연동
  - Requests: API 데이터 호출
  - Flask: API 서버 및 웹 페이지 구축

### 나. 개발 도구 및 환경

#### (1) IDE 및 개발 도구

- ☐ Visual Studio Code: 데이터 분석 및 시각화
- ☐ Jupyter Notebook: 데이터 분석 및 시각화
- ☐ MobaXterm: 서버 접근 및 실행 환경 관리

## 다. 데이터 및 API 출처

### (1) 데이터 출처

#### □ 기상청

- 데이터>기상관측>지상>종관기상관측(ASOS), [링크](https://data.kma.go.kr/data/grnd/selectAsosRltmList.do?pgmNo=36)  
(<https://data.kma.go.kr/data/grnd/selectAsosRltmList.do?pgmNo=36>)
- 기후통계분석>기상현상일수>폭염일수, [링크](https://data.kma.go.kr/climate/heatWave/selectHeatWaveChart.do)  
(<https://data.kma.go.kr/climate/heatWave/selectHeatWaveChart.do>)
- 기후통계분석>기상현상일수>한파일수, [링크](https://data.kma.go.kr/climate/cdwv/selectCdwvChart.do?pgmNo=733)  
(<https://data.kma.go.kr/climate/cdwv/selectCdwvChart.do?pgmNo=733>)
- 기후통계분석>기상현상일수>열대야일수, [링크](https://data.kma.go.kr/climate/tropicalNight/selectTropicalNightChart.do)  
(<https://data.kma.go.kr/climate/tropicalNight/selectTropicalNightChart.do>)

#### □ IPCC 6차 보고서: 공유 사회경제 경로 SSP 시나리오, [링크](https://www.ipcc.ch/report/ar6/wg1/figures/summary-for-policymakers/figure-spm-8)

(<https://www.ipcc.ch/report/ar6/wg1/figures/summary-for-policymakers/figure-spm-8>)

### (2) API 출처

- OpenWeather API: 5day/3hour forecast, [링크](https://openweathermap.org/forecast5) (<https://openweathermap.org/forecast5>)
- OpenStreetMap, [링크](https://download.geofabrik.de/) (<https://download.geofabrik.de/>)

### 3. 프로젝트 설계 및 구현

#### 가. R 및 Python 분석

##### (1) 데이터 분석 과정 및 알고리즘 설명

- 과거 관측 데이터와 외생 변수를 활용한 미래 기온 예측
  - 시계열 예측
    - SARIMAX(Seasonal Autoregressive Integrated Moving Average with Exogenous Variables) 모델을 활용하여 시계열 데이터 기반의 미래 기온을 예측한다.
    - 외생 변수로 IPCC 6차 보고서에서 제공하는 SSP(공유 사회경제 경로) 시나리오 데이터를 사용한다.
  - 데이터 활용
    - 기상청 종관기상관측 연평균 기온 데이터(1951~2023)를 기반으로 시계열 분석을 진행한다.
    - SSP 시나리오
      - SSP1(탄소 감축 시나리오): 온실가스 배출이 감소하며 지속 가능한 발전을 달성한다.
      - SSP2(현행 유지 시나리오): 현재의 발전 수준과 탄소 배출량이 지속적으로 유지된다.
  - 예측 목표
    - 전국 단위 평균 기온 및 광역시·도 단위 지역별 기온을 예측한다.
    - 2100년까지의 기온 추이를 SSP1과 SSP2 시나리오를 기반으로 분석한다.
- 기상현상 발생 현황 추이 시각화
  - 기상청 기후 통계분석 기상현상일수 데이터(1951~2024)를 활용한다.
    - 폭염: 일 최고기온 33도(℃) 이상
    - 열대야: 밤 최저기온 25도(℃) 이상
    - 한파: 아침 최저기온 -12도(℃) 이하
  - 분석 목표
    - 기온 상승에 따라 폭염, 열대야, 한파와 같은 극단적 기상 현상의 발생 빈도 변화를 시각화한다.

##### (2) 코드 설명과 주요 결과

- 과거 관측 데이터와 외생 변수를 활용한 미래 기온 예측
  - 데이터 준비
    - 기상청 종관 기상관측 데이터를 활용하여 특정 지역 데이터를 필터링한다.
    - IPCC 6차 보고서의 탄소 배출량 시나리오 데이터를 활용하여 각 연도별 평균 기온을 추출한다.

- 관측 데이터(평균기온(°C))와 SSP 데이터를 연도별로 병합하여 예측 모델의 학습 데이터를 생성한다.

```
1 daejeon_data = yearly_data[yearly_data['지점명'].str.contains('대전', na=False)]
2 daejeon_data = daejeon_data.rename(columns={'일시': 'Year'}) # 연도 컬럼 이름 변경
```

#### ○ SARIMAX 모델 구축

- SARIMAX 모델은 관측된 시계열 데이터와 외생 변수(Mean)를 결합하여 예측 정확도를 향상한다.
- Auto ARIMA를 활용하여 최적의 (p, d, q) 파라미터를 자동 탐색하여 SARIMAX 모델에 적용한다.

```
1 auto_arima_model = auto_arima(
2     y, exogenous=exog,
3     seasonal=False,
4     stepwise=True,
5     suppress_warnings=True,
6     error_action="ignore",
7     max_order=None
8 )
```

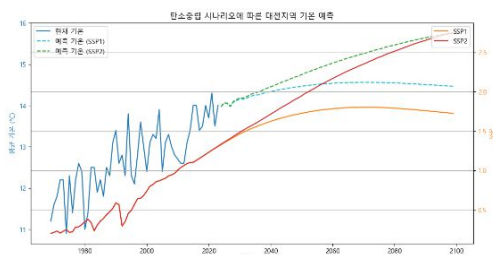
#### ○ 미래 기온 예측

- 2024년부터 2100년까지의 미래 연도 데이터를 생성하고 SSP 데이터를 외생 변수로 사용하여 예측한다.
- SARIMAX 모델의 get\_forecast 메서드를 활용해 예측값을 도출한다.

```
1 future_years = pd.DataFrame({'Year': range(2024, 2100)})
2 forecast = results.get_forecast(steps=len(future_years), exog=future_exog)
3 forecasted_values = forecast.predicted_mean.reset_index(drop=True)
```

#### ○ 주요 결과

- SSP1 시나리오의 경우 탄소 배출량 감소에 따라 기온 상승 폭이 완화되며 안정적인 증가 추이를 보인다
- SSP2 시나리오의 탄소 배출이 지속될 경우 기온 상승이 가속화되며 SSP1과 차이가 점차 벌어진다.



#### □ 기온 현상 발생 현황 추이 시각화

- matplotlib를 활용해 막대그래프와 추세선으로 추이를 시각화한다.
- 주로 기온에 영향을 받는 기상현상인 폭염, 열대야, 한파에 대한 추이 시각화를 진행했다.
- 21세기에 들어서며 더욱 극단적인 추이 변화를 보이고 있으며 특히 열대야의 발생 증가가 두드러짐을 알 수 있다.



## 나. PHP 기반 웹 구현

### (1) 웹 구조 및 기능 설명

#### □ 웹 구조

##### ○ 기후변화 지도 기반 대시보드

- 중앙에 OpenStreetMap 기반의 대화형 지도를 배치하고, 상단과 하단에 기후 데이터를 시각화하는 요소를 통합한 구조이다.

##### ○ 주요 구성 요소

- 상단 영역
  - 왼쪽: SARIMAX 모델을 기반으로 생성된 예측 그래프를 PHP에서 처리하고 iframe으로 삽입한다.
  - 오른쪽: OpenWeather API를 활용한 실시간 날씨 정보 제공하며 오늘과 내일의 기상 정보를 드롭다운 메뉴 아래 표시한다.
- 중앙 지도
  - OpenStreetMap 기반 대화형 지도: 사용자 선택 지역의 데이터를 시각화하며, 지도상에 마커와 팝업으로 관련 정보를 제공한다.
- 하단 데이터 박스
  - 기후 현상 분석: 폭염, 열대야, 한파와 같은 기상현상의 연도별 추이를 시각화하고 SARIMAX 모델 기반 예측 데이터를 통합하여 표시한다.

#### □ 기능 설명

##### ○ 드롭다운 메뉴 기능

- PHP 기반으로 구현된 드롭다운 메뉴를 통해 사용자가 지역을 선택할 수 있으며 선택된 값에 따라 iframe과 모든 관련 데이터가 자동으로 동기화된다.

```
1 <form method="GET" action="">
2   <select name="region" onchange="this.form.submit()">
3     <option value="Seoul">서울</option>
4     <option value="Daejeon">대전</option>
5     <option value="Busan">부산</option>
6   </select>
7 </form>
```

##### ○ 지도 데이터 업데이트

- 선택된 지역 중심으로 지도 이동 및 데이터 시각화를 구현하며 GeoJSON 형식 데이터를 불러와 마커와 팝업으로 관련 정보를 제공한다.

##### ○ OpenWeather API 활용

- API를 호출하여 선택된 지역의 실시간 기온, 강수 확률, 최저/최고 기온 등을 표시하며 드롭다운 값 변경 시 자동으로 데이터를 반영한다.

##### ○ iframe 활용

- SARIMAX 예측 그래프 및 기후 현상 분석 결과는 각각 PHP 파일에서 처리된 데이터를 iframe으로 페이지에 통합한다.

### (2) 결과 출력 방식 및 화면 구성

#### □ 결과 출력 방식

○ iframe 기반 출력

- SARIMAX 모델 기반 예측 데이터 및 시각화 결과는 각각 PHP 파일에서 처리된 후 iframe을 통해 통합된다.

```
1 <iframe src="py/temperature_forecast.php" style="width: 100%; height: 100%; border: none;"></iframe>
```

○ 지도 출력

- OpenStreetMap API를 활용하여 선택된 지역을 중심으로 강조 표시하며, GeoJSON 데이터를 통해 동적으로 업데이트된다.

```
1 function updateMap(region) {  
2     const geoJsonUrl = `/data/geo/${region}.geojson`;   
3     fetch(geoJsonUrl)  
4         .then(response => response.json())  
5         .then(data => {  
6             map.setView([data.coordinates.lat, data.coordinates.lng], 10);  
7             L.geoJSON(data).addTo(map).bindPopup(`지역: ${region}`);  
8         });  
9 }
```

○ OpenWeather API 연동

- 실시간 날씨 데이터를 제공하며, 드롭다운 값 변경 시 자동 반영된다.

□ 화면 구성

○ 상단

- 왼쪽: 전국 및 지역별 SARIMAX 기반 기온 예측 그래프를 시각화한다.
- 오른쪽: 지역 선택 드롭다운 메뉴 및 OpenWeather API 활용 실시간 날씨 정보를 표시한다.

○ 중앙

- OpenStreetMap 기반 대화형 지도를 통해 선택된 지역에 따라 지도 중심 좌표와 마커가 업데이트된다.

○ 하단

- 폭염, 열대야, 한파와 같은 기상현상의 연도별 변화 추이를 시각화한 그래프를 iframe으로 출력한다.

## 다. 웹 사용자 접근성 및 디자인

### (1) 사용자 접근성을 고려한 설계 요소

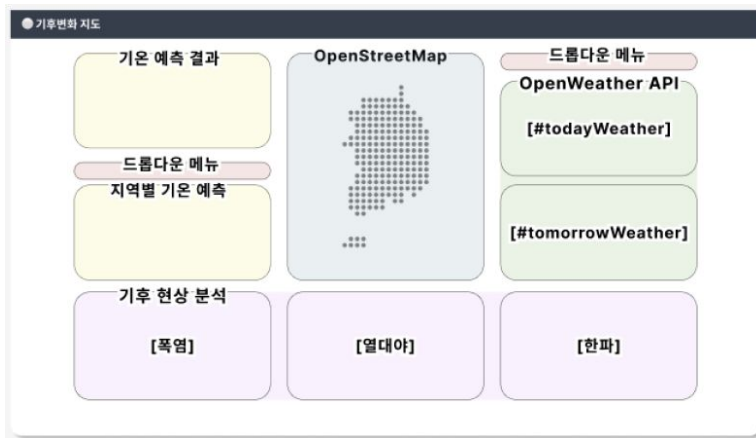
□ 중앙 지도 기반 데이터 시각화

- 화면 중앙에 OpenStreetMap을 배치하여 지도 위에서 지역별 기후 데이터를 직관적으로 확인할 수 있도록 구성한다.
- 기온 예측 결과와 기후 현상 분석은 서로 다른 섹션으로 나누어 배치하여, 사용자가 필요한 정보를 빠르게 탐색할 수 있도록 돕는다.

□ 상호작용을 위한 드롭다운 메뉴 제공

- 드롭다운 메뉴를 통해 사용자가 지역을 선택할 수 있도록 하여 기후 데이터 접근을 간소화한다.
- 사용자의 탐색 흐름을 고려한 상호 연동 레이아웃으로 설계하여 정보가 통합적으로 연결되도록 구성한다.

## (2) 인터페이스 디자인 및 화면 예시



## 4. 팀워크 및 역할 분담

### 가. 팀원별 역할과 업무 내용

- (1) 박성빈 (빅데이터응용학과, 20222303)
  - ☐ 지도 시각화
  - ☐ 드롭 다운 상호작용 연결
  - ☐ 최종 HTML 구현
- (2) 원현아 (수학과, 20202478)
  - ☐ 팀장
  - ☐ 발표
- (3) 이소담 (수학과, 20202450)
  - ☐ 기상 API JavaScript 작성 및 HTML 구성
  - ☐ 발표 PPT 작성
  - ☐ 보고서 작성
- (4) 임하은 (글로벌비즈니스학과, 20211944)
  - ☐ 지역별 기상예측 시각화 및 지역 선택 드롭다운 PHP 구현
  - ☐ OpenWeather API 연동 및 PHP 구현
- (5) 최원준 (빅데이터응용학과, 20191537)
  - ☐ SSP를 통한 미래 기상 예측
  - ☐ 기상현상 발생 추이 시각화
  - ☐ 보고서 작성

### 나. 협업 도구 활용

- (1) Notion을 활용해 팀원들과 프로젝트 정보 및 내용 교환

<https://www.notion.so/2024-2-11387ef208478008a7a9d94a26b9b182?pvs=4>

프로젝트와 관련된 자료 및 진행 상황은 노션에 체계적으로 정리하고 공유하여 모든 팀원이 쉽게 접근할 수 있도록 했다.

### 다. 프로젝트 진행 중 의사소통 및 문제 해결 과정

- (1) 정기 회의를 통한 진행 상황 점검
  - ☐ 팀원들의 기술 역량과 관심 분야를 기반으로 역할을 분담하여 각자 맡은 업무를 효율적으로 진행했다.
  - ☐ 매주 수업 시작 1시간 전에 팀원들이 모여 진행 상황과 결과를 공유하고 프로젝트의 다음 단계와 목표를 논의했다
  - ☐ 회의 중에는 각자의 진행 상황에 대한 피드백을 주고받으며, 작업의 방향성을 점검하고

필요한 조정을 수행했다.

## 5. 프로젝트 결과 및 평가

### 가. 프로젝트 결과 요약

#### (1) 기후 데이터 시각화 대시보드 구축

- OpenStreetMap 기반 대화형 지도와 SARIMAX 모델을 활용한 기온 예측 데이터를 통합하여 대시보드를 구축했다.
- 폭염, 열대야, 한파와 같은 극단적 기상현상의 연도별 발생 추이를 시각화함으로써, 기후 변화가 지역 사회에 미치는 영향을 전달했다.

#### (2) 실시간 데이터 연동

- OpenWeather API를 활용해 오늘과 내일의 날씨 정보를 실시간으로 제공, 지역별 맞춤형 정보 출력 가능하며 선택된 지역에 따라 지도와 기상 데이터가 자동으로 업데이트되어 실시간 모니터링을 진행할 수 있다.

#### (3) 예측 데이터 분석

- SSP1(탄소 배출 감축) 및 SSP2(현행 유지) 시나리오를 적용하여 전국 및 지역별 기온 변화를 예측했다.

### 나. 프로젝트를 통해 얻은 학습 성과

#### (1) 데이터 분석 및 시각화 경험 강화

- GeoJSON과 OpenStreetMap을 활용해 데이터를 시각화하며, 사용자 경험을 향상하는 대화형 지도 구축 방법을 학습했다.

#### (2) PHP를 통한 API 연동 경험

- Python 대신 PHP를 사용하여 OpenWeather API와 기상청 데이터를 연동하고 이를 웹 대시보드에 통합하는 과정을 경험하며 동적 웹 페이지 구축 방법을 익히고 새로운 언어와 프레임워크 활용 능력을 확장했다.

### 다. 프로젝트 수행 중 발생한 문제와 해결 방안

#### (1) PHP와 JavaScript 간의 처리 불일치

- 프로젝트에서 사용된 col.php 파일이 JavaScript를 통해 실행되도록 설계되었으나 다른 PHP 파일과 데이터 동기화가 원활히 이루어지지 않는 문제가 발생했다.
  - col.php가 JavaScript 기반으로 설계된 것이 주요 원인으로 AJAX를 사용해 비동기 통신을 구현하려 했으나, 모든 PHP 파일을 AJAX 방식으로 재설계해야 하는 복잡성이 존재했다.
- col.php를 PHP 기반으로 재설계하여 다른 PHP 파일들과의 동기화하고 AJAX 호출을 최소화하며 PHP 내부에서 데이터를 처리하여 프로젝트의 구조적 일관성과 유지보수성을 높였다.

## 6. 창의적 요소 및 개선 사항

### 가. 창의적 아이디어 및 차별화된 요소

#### (1) 기후 데이터와 대화형 지도의 통합

- OpenStreetMap 기반 대화형 지도를 활용하여 기후 데이터를 시각적으로 통합, 사용자가 선택한 지역의 기온 변화 및 주요 기상현상을 직관적으로 확인할 수 있도록 구현했다.

#### (2) SSP 시나리오 기반 기후 예측

- IPCC의 SSP1(탄소 배출 감축)과 SSP2(현행 유지) 시나리오를 활용하여, 지역별 기온 변화와 극단적 기상현상의 발생 추이를 예측했다.

### 나. 향후 보완 및 개선 방향

#### (1) API 연계 개선

- 단기 예측 API 도입
  - 현재 사용 중인 OpenWeather 무료 API는 3시간 단위의 예측 데이터를 제공하며 이를 통해 날씨 정보를 시각화하고 있다.
  - 기상청 API 허브의 단기 예측 API를 도입하면 1시간 단위의 예측 데이터를 확보할 수 있어 기상 데이터의 예측 정밀도를 향상하고 신뢰성 높은 정보를 제공할 수 있다.
- 종합적인 기후 데이터 통합 제공
  - 기온 예측뿐만 아니라 해수면 상승, 대기질 등 다양한 기후 데이터를 통합하여 제공함으로써 종합적인 기후 정보를 사용자에게 전달한다.

#### (2) 예측 모델 고도화

- SSP 시나리오 적용 확장
  - 현재 프로젝트는 연도별 지역 기온 예측에 SSP 시나리오를 적용하여 미래 기온 변화를 시각화하고 있다.
  - 폭염, 열대야, 한파와 같은 주요 기상현상의 발생 추이를 SSP 시나리오 기반으로 예측하고 시각화한다면 극단적 기상현상의 변화 양상을 직관적으로 전달할 수 있다.
- 모델 성능 향상
  - 기존 SARIMAX 모델에 추가적인 외생 변수를 포함하여 모델의 설명력을 강화한다.
  - 다양한 시계열 모델(LSTM, Prophet)과의 비교를 통해 모델 성능을 향상하고 최적의 모델을 선정하여 기온 예측의 정확도를 높인다.

## 7. 결론

### 가. 프로젝트의 최종 결론

본 프로젝트는 기상 데이터를 기반으로 계절 길이 변화와 극단적 기상현상 발생 추이를 분석하고 시각화하여 사용자들에게 현재와 미래의 기후 변화를 효과적으로 전달하는 것을 목표로 진행되었다.

#### (1) 주요 결과

- **전국 및 지역별 기온 변화 예측:** 과거 기상 관측 데이터와 SSP 시나리오를 활용하여 연도별 기온 변화를 예측하고 시각화하여 지역별 기후 변화 양상을 명확히 제시하였다.
- **폭염·열대야·한파 등의 기상현상 추이 시각화:** 기후 변화로 인해 발생하는 주요 기상현상의 연도별 빈도와 강도를 분석하고 극단적 기상현상의 변화 추이를 시각적으로 표현하였다.
- **사용자 중심의 웹 기반 대시보드 구축:** 실시간 기상 데이터를 제공하고 예측 정보를 시각화하여 사용자들이 데이터를 이해할 수 있는 인터페이스를 구현하였다.

#### (2) 분석 결과

- **계절 변화 심화:** 기온 상승과 계절 길이 변화로 인해 여름은 길어지고 겨울은 짧아지는 경향이 뚜렷하게 나타났다.
- **극단적 기상현상 증가:** 폭염과 열대야의 발생 빈도와 강도는 지속적으로 증가하는 반면, 한파의 발생 빈도는 감소하는 추세를 보였다.

### 나. 프로젝트를 마친 소감

#### (1) 박성빈

- 이번 프로젝트를 통해 OpenStreetMap을 활용한 지도 구현과 WeatherFrame, 지역별 예측 데이터 동기화를 경험하며 데이터 통합과 UX 최적화의 중요성을 배웠습니다. 드롭다운 방식과 JavaScript, PHP를 통해 기능 구현과 호환성 개선을 동시에 고려한 점이 의미 있었고, 문제 해결 능력과 개발 역량을 한 단계 성장시킬 수 있었습니다.

#### (2) 원현아

- 이번 프로젝트를 마치며 나는 아직 배우고 공부해야하는 부분이 많다는 것을 깨달았다. 여러 데이터를 한 페이지에 표현하고 더 나아가 그 부분을 활용해 웹사이트를 제작할 수 있다는 것은 흥미로웠지만 그것을 생각한대로 구현하는 과정에서는 어려움을 많이 느꼈다. 팀원분들과 함께 하였기에 해결할 수 있었던 문제들이 많았기에 나 스스로가 성장하고 앞으로 더 나아가기 위해서는 여기서 그치지 않고 수업시간에 배운 내용과 이번 프로젝트 경험을 활용해 열심히 배우고 공부해야겠다.

#### (3) 이소담

- 이번 프로젝트는 졸업 전 학부과정의 마지막 프로젝트로 그동안 다루지 못했던 웹 개발 경험을 쌓기위해 수강했습니다. 데이터 분석과 예측 작업 이후에 실제 사용자에게 제공될



수 있는 형태로 전환되는 과정을 경험했습니다. 학습했던 이론을 실무적으로 활용해 보고 결과를 웹에서 시각화하는 과정은 앞으로의 진로에도 중요한 밑거름이 될 것이라고 기대합니다.

#### (4) 임하은

□ 프로그램 호환 수업 프로젝트를 마치며 HTML과 PHP를 처음 접했는데, 어려운 부분도 있었지만 웹 페이지에서 결과를 직접 확인할 수 있어 흥미로웠습니다. 특히, MobaXterm을 사용해 파이썬과 API를 연동하는 방법을 배우면서, 원래 Streamlit만 사용해본 저에게 새로운 접근 방식을 익힐 수 있었습니다. PHP를 활용한 작업은 처음이라 도전적이었지만, 이를 통해 웹 개발의 기초를 쌓을 수 있었고, 앞으로 더 배우고 싶다는 생각이 들었습니다.

#### (5) 최원준

□ 이번 프로젝트를 통해 데이터 분석과 시각화에 대한 결과를 웹페이지에 표현하는 과정을 경험했습니다. 지금까지는 Python과 R을 이용하여 데이터 분석만 진행했다면 이번 기회를 통해 서버를 통하여 제가 구현한 결과에 대해 사용자들이 직접 필요한 결과를 찾아볼 수 있는 경험을 제공할 기회가 되었습니다. 앞으로 프로젝트를 진행하며 분석결과를 표현하는 방법에 새로운 옵션을 추가하게 되어 좋은 경험이 된 것 같습니다.

## 8. 부록

### 가. 참고문헌 및 자료 출처

#### (1) 기상청

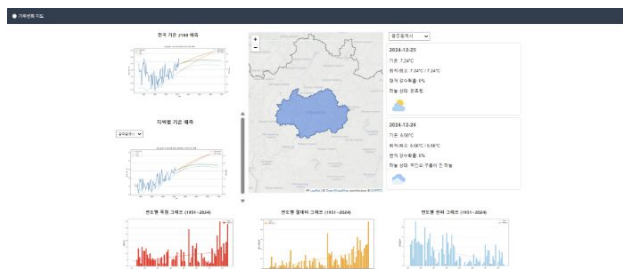
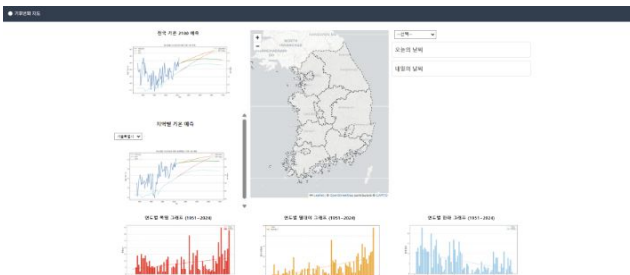
- 기후정보포털 기후변화 상황지도, [링크](http://climate.go.kr/atlas/dsh/ccf) (<http://climate.go.kr/atlas/dsh/ccf>)
- 기후변화 영향정보>기상·기후 부분>계절길이, [링크](http://www.climate.go.kr/home/CCS/contents_2021/influence/inf_2-1.php) ([http://www.climate.go.kr/home/CCS/contents\\_2021/influence/inf\\_2-1.php](http://www.climate.go.kr/home/CCS/contents_2021/influence/inf_2-1.php))

#### (2) 참고 문헌

- 박상현 기자. (2024.10.07). 조선일보. 4계절 패턴도 붕괴... 100년새 여름 한달 늘고, 겨울 20일 줄었다. [링크](https://www.chosun.com/national/transport-environment/2024/10/07/2LX257BN2NCIVBKRB5HVPNC5XA/) (<https://www.chosun.com/national/transport-environment/2024/10/07/2LX257BN2NCIVBKRB5HVPNC5XA/>)
- 이재영 기자. (2024.08.26.). 연합뉴스. 올가을, '지각'...예년 시작일 지났지만 여전히 기온 높아. [링크](https://www.yna.co.kr/view/AKR20240926161400530) (<https://www.yna.co.kr/view/AKR20240926161400530>)
- 김예원 기자. (2024.09.22.). 뉴스1. "30여년 뒤 부산 겨울 사라질 수도"...사계절 길이 조정시 무슨 일이. [링크](https://www.news1.kr/society/incident-accident/5547160) (<https://www.news1.kr/society/incident-accident/5547160>)

### 나. 추가 자료

#### (1) 최종 결과물 [https://bda.dcarecloud.com/~stud\\_10/korea3.php](https://bda.dcarecloud.com/~stud_10/korea3.php)



#### (2) 코드 전체 내용

- korea3.php

```
<!DOCTYPE html>
<html lang="ko">
<head>
```

```

<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>기후변화 지도</title>
<link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />
<link href="https://fonts.googleapis.com/css2?family=Noto+Sans+KR&display=swap" rel="stylesheet">
<style>
  body {
    font-family: 'Noto Sans KR', sans-serif;
    margin: 0;
    padding: 0;
    background-color: #ffffff;
  }

  header {
    background-color: #2c3e50;
    color: white;
    padding: 20px;
    font-size: 1rem;
    display: flex;
    align-items: center;
    justify-content: space-between;
    position: relative;
  }

  .region-selector {
    display: flex;
    align-items: center;
    gap: 10px;
    margin: 20px;
  }

  #regionSelect {
    padding: 5px;
    font-size: 1rem;
  }

  #regionSelectBtn {
    padding: 5px 10px;
    font-size: 1rem;
    background-color: #3498db;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
  }

  #map {
    width: 100%;
    height: 600px;
    margin-top: 20px;
  }

  .container {
    display: grid;
    grid-template-areas:

```

```

        "box1 map box2"
        "box3 map box2"
        "box5 box6 box7";
    grid-template-columns: auto auto auto;
    grid-gap: 10px;
    justify-content: center;
    padding: 20px;
}

.box {
    background-color: #ffffff;
    border: none;
    border-radius: 8px;
    text-align: center;
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 1rem;
    font-weight: bold;
}

.map {
    grid-area: map;
    width: 500px;
    height: 600px;
    background-color: #ffffff;
    border: none;
    border-radius: 8px;
    display: flex;
    justify-content: center;
    align-items: center;
}

.box1 { grid-area: box1; width: 500px; height: 291px; }
.box2 { grid-area: box2; width: 500px; height: 620px; }
.box3 { grid-area: box3; width: 500px; height: 350px; }
.box5 { grid-area: box5; width: 500px; height: 270px; }
.box6 { grid-area: box6; width: 500px; height: 270px; }
.box7 { grid-area: box7; width: 500px; height: 270px; }

.leaflet-popup-content {
    font-family: 'Noto Sans KR', sans-serif;
}
</style>

<script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
</head>
<body>
    <header>&#9898; 기후변화 지도</header>

    <div class="container">
        <div class="box box1">
            <iframe src="py/temperature_forecast.php" style="width: 100%; height:
100%; border: none;"></iframe>
        </div>
        <div class="map">

```

```

        <div id="map" style="width: 100%; height: 100%;"></div>
    </div>
    <div class="box box2">
        <iframe id="weatherFrame" name="weatherFrame" src="py/col2.php"
style="width: 100%; height: 100%; border: none;"></iframe>
    </div>
    <div class="box box3">
        <iframe src="py/WF.php" style="width: 100%; height: 100%; border:
none;"></iframe>
    </div>
    <div class="box box5">
        <iframe src="py/heatwave.php" style="width: 100%; height: 100%; border:
none;"></iframe>
    </div>
    <div class="box box6">
        <iframe src="py/tropical.php" style="width: 100%; height: 100%; border:
none;"></iframe>
    </div>
    <div class="box box7">
        <iframe src="py/coldwave.php" style="width: 100%; height: 100%; border:
none;"></iframe>
    </div>
</div>

<script>
    document.addEventListener('DOMContentLoaded', function() {
        const weatherFrame = document.getElementById('weatherFrame'); // col.php
        const wfFrame = document.querySelector('iframe[src="py/WF.php"]'); // wf.php

        // col.php의 드롭다운 변경 이벤트 처리
        weatherFrame.addEventListener('load', function() {
            const weatherSelect = weatherFrame.contentWindow.document.querySelector('select[name="region"]');
            if (weatherSelect) {
                weatherSelect.addEventListener('change', function() {
                    const selectedRegion = this.value;

                    // WF.php의 드롭다운 값 변경
                    const wfSelect = wfFrame.contentWindow.document.querySelector('select[name="region"]');
                    if (wfSelect) {
                        wfSelect.value = selectedRegion;
                        wfSelect.form.submit();

                        // WF.php의 드롭다운 값이 변경되었으므로 별도의 submit 호출 없이
                        // 지역 값 갱신됨
                        wfSelect.dispatchEvent(new Event('change')); // 'change' 이벤
                        트를 트리거하여 자동으로 반영되도록 함
                    }

                    // 지도 업데이트
                    window.postMessage({ region: selectedRegion }, '*');
                });
            }
        });

        // wf.php의 드롭다운 변경 이벤트 처리

```

```

        wfFrame.addEventListener('load', function() {
            const wfSelect = wfFrame.contentWindow.document.querySelector('select[name="region"]');
            if (wfSelect) {
                wfSelect.addEventListener('change', function() {
                    const selectedRegion = this.value;

                    // col.php의 드롭다운 값 변경
                    const weatherSelect = weatherFrame.contentWindow.document.querySelector('select[name="region"]');
                    if (weatherSelect) {
                        weatherSelect.value = selectedRegion;
                        weatherSelect.form.submit();
                    }

                    // 지도 업데이트
                    window.postMessage({ region: selectedRegion }, '*');
                });
            }
        });
    });
</script>

    // 부모 페이지에서 지역 값으로 지도 업데이트
    window.addEventListener('message', function(event) {
        const region = event.data.region;
        if (region) {
            updateMap(region);
        }
    });
</script>

<script src="js/map.js"></script>
</body>
</html>

```

□ SARIMAX.ipynb

```

import pandas as pd
from statsmodels.tsa.statespace.sarimax import SARIMAX
from pmdarima import auto_arima
import matplotlib.pyplot as plt

plt.rcParams['font.family'] = 'Malgun Gothic'

# 데이터 파일 경로 설정
yearly_data = pd.read_csv(r"C:\Users\choiw\Desktop\Lecture\24_2\프로그램호환\data\yearly_50_23.csv", encoding='euc-kr')
ssp1_data = pd.read_csv(r"C:\Users\choiw\Desktop\Lecture\24_2\프로그램호환\data\SSP1_2_6.csv", encoding='euc-kr')
ssp2_data = pd.read_csv(r"C:\Users\choiw\Desktop\Lecture\24_2\프로그램호환\data\SSP2_4_5.csv", encoding='euc-kr')

# 대전 지역 데이터 필터링

```

```

daejeon_data = yearly_data[yearly_data['지점명'].str.contains('대전', na=False)]
daejeon_data = daejeon_data.rename(columns={'일시': 'Year'}) # 연도 컬럼 이름 변경

# SSP 데이터에서 'Mean' 값 추출
ssp1_mean = ssp1_data[['Year', 'Mean']]
ssp2_mean = ssp2_data[['Year', 'Mean']]

# 대전 데이터 기간에 맞춰 SSP 데이터 필터링
ssp1_mean = ssp1_mean[ssp1_mean['Year'] >= daejeon_data['Year'].min()]
ssp2_mean = ssp2_mean[ssp2_mean['Year'] >= daejeon_data['Year'].min()]

# SARIMAX 예측 함수 정의
def sarimax_forecast(ssp_mean, label):
    combined_data = pd.merge(daejeon_data, ssp_mean, on='Year', how='inner')

    # SARIMAX 모델 학습 및 예측 준비
    y = combined_data['평균기온(°C)']
    exog = combined_data[['Mean']]

    # Auto ARIMA를 사용해 최적의 (p, d, q) 값 찾기
    print(f"Finding best parameters for {label}...")
    auto_arima_model = auto_arima(
        y, exogenous=exog,
        seasonal=False, # 계절성 반영 여부
        stepwise=True, # 단계적으로 탐색
        suppress_warnings=True,
        error_action="ignore",
        max_order=None
    )
    print(f"Best parameters for {label}: {auto_arima_model.order}")

    # SARIMAX 모델 설정 및 학습
    model = SARIMAX(y, exog=exog, order=auto_arima_model.order)
    results = model.fit(dispatch=False)

    # 2024년부터 2099년까지 예측
    future_years = pd.DataFrame({'Year': range(2024, 2100)})
    future_exog = pd.merge(future_years, ssp_mean, on='Year', how='left')['Mean']
    forecast = results.get_forecast(steps=len(future_years), exog=future_exog)

    # 예측 결과 정리
    forecasted_values = forecast.predicted_mean.reset_index(drop=True)
    future_years = future_years.reset_index(drop=True)
    forecasted_data = future_years.copy()
    forecasted_data['평균기온(°C)'] = forecasted_values

    return combined_data, forecasted_data

# SSP1 데이터 예측
combined_data1, forecasted_data1 = sarimax_forecast(ssp1_mean, 'SSP1')

# SSP2 데이터 예측
combined_data2, forecasted_data2 = sarimax_forecast(ssp2_mean, 'SSP2')

# 그래프 출력
fig, ax1 = plt.subplots(figsize=(12, 6))

```

```

# 평균기온 데이터 플롯
ax1.plot(combined_data1['Year'], combined_data1['평균기온(°C)'], label='현재 기온',
color='tab:blue')
ax1.plot(forecasted_data1['Year'], forecasted_data1['평균기온(°C)'], label='예측 기온
(SSP1)', linestyle='--', color='tab:cyan')
ax1.plot(forecasted_data2['Year'], forecasted_data2['평균기온(°C)'], label='예측 기온
(SSP2)', linestyle='--', color='tab:green')
ax1.set_xlabel('Year')
ax1.set_ylabel('평균 기온 (°C)', color='tab:blue')
ax1.tick_params(axis='y', labelcolor='tab:blue')
ax1.legend(loc='upper left')

# SSP Mean 데이터 플롯 (별도 y축 설정)
ax2 = ax1.twinx()
ax2.plot(ssp1_mean['Year'], ssp1_mean['Mean'], label='SSP1', color='tab:orange')
ax2.plot(ssp2_mean['Year'], ssp2_mean['Mean'], label='SSP2', color='tab:red')
ax2.set_ylabel('SSP', color='tab:orange')
ax2.tick_params(axis='y', labelcolor='tab:orange')
ax2.legend(loc='upper right')

plt.title('탄소중립 시나리오에 따른 대전지역 기온 예측')
plt.grid()
plt.show()

```