진해여자고등학교 AI 창의 융합 독서 프로젝트 활동 강의자료 (7월 19일)

이 자료는 진해여자고등학교 학생들을 대상으로 2025년 7월 18-19일 양일간 진행된 **AI 창의 융합 독서 프로젝트**에서 실제로 사용된 **Jupyter Notebook 기반의 코딩 실습 활동**을 정리한 것입니다.
학생들은 수학적 개념, 프로그래밍 사고력, 데이터 분석 및 시각화 도구를 통합적으로 경험하며, 데이터 과학의 기초를 자연스럽게 익히는 것을 목표로 합니다.

🎯 교육적 목표

각 활동은 다음과 같은 목표 아래 설계되었습니다:

1. **수학과 과학 개념의 실제 적용**
   - 함수와 도함수의 개념을 경사하강법, 선형 회귀, 로지스틱 회귀 등 실제 문제에 적용
   - 유클리드 거리 개념을 통해 데이터 간 '거리' 개념을 직관적으로 이해
   - 확률 함수(시그모이드)와 선형 결합을 이용한 분류 모델 체험
2. **코딩을 통한 문제 해결과 시각화**
   - Python 기반 데이터 생성, 가공, 시각화를 실습하며 알고리즘 사고력 증진
   - Folium, Matplotlib 등을 이용해 지리정보 및 수치 정보를 시각적으로 표현
3. **기초 머신러닝 개념 체험**
   - KNN, 선형 회귀, 로지스틱 회귀 등 지도학습 알고리즘을 실제 데이터를 통해 경험
   - 알고리즘의 작동 원리를 단계별로 관찰하고 예측 과정 이해
4. **데이터 리터러시와 실세계 연결**
   - 농업, 기후 변화, 유전자 등 다양한 주제를 통해 데이터 분석의 실용성 체험
   - 단순 수치가 아닌 '현상 해석'으로 확장할 수 있는 과학적 사고 촉진

✍ 작성자

백대성
젤리코딩학원(창원) 운영 중.

실습 코드 및 설명자료는 아래의 깃허브 소스코드 공유 사이트에 공개 하였습니다.

해당자료를 다운받아, 구글 코랩에서 실행하거나, 컴퓨터에 아나콘다 파이썬을 설치하고 주피터 랩을 실행하여 실습합니다.

https://github.com/dsbaek2020/JGH_ThinkingForSociety/tree/main

🌎 기후 변화와 작물 추천 시뮬레이션

🎯 목표

- 기후 변수(기온, 이산화탄소 농도, 강수량)의 변화 양상을 살펴보고
- 이를 기반으로 **작물 추천 알고리즘을 직접 구현**해 봅니다.
- 단순한 이론이 아닌, 현실적 데이터 기반의 **기후-농업 연결 문제를 탐구**합니다.

1️⃣ 기후 변화 시계열 분석

- **연도별 온도는 오르고**, CO_2 농도 역시 **뚜렷한 증가 추세**를 보입니다.
- 강수량은 일정한 주기를 가지며 변동하고 있습니다.
- 시각화를 통해 **패턴의 존재를 눈으로 확인**합니다.

2️⃣ CO_2와 온도 사이의 관계

- 두 변수의 산점도를 그려보면 거의 일직선처럼 보입니다.
- 선형 회귀를 통해 다음을 확인할 수 있습니다:

```
Temperature ≈ Slope * CO2 + Intercept
```

- 이 결과는 **기온 상승에 CO_2가 영향을 미친다**는 것을 직관적으로 보여줍니다.

3️⃣ 경남 시군 단위 작물 추천 실험

- **창원, 진주, 밀양, 함양, 거제** 등 5개 도시의 연도별 기온·강수량 데이터 생성
- 작물별 선호 조건(T_opt, R_opt)을 설정한 뒤, **거리 기반 점수**를 계산:

$$Score = \exp(-((T - T_{opt})^2)/2) * \exp(-((R - R_{opt})^2)/5000)$$

- 선형 회귀 모델 없이, 점수 최대값을 기준으로 **각 도시별 추천 작물**을 선택

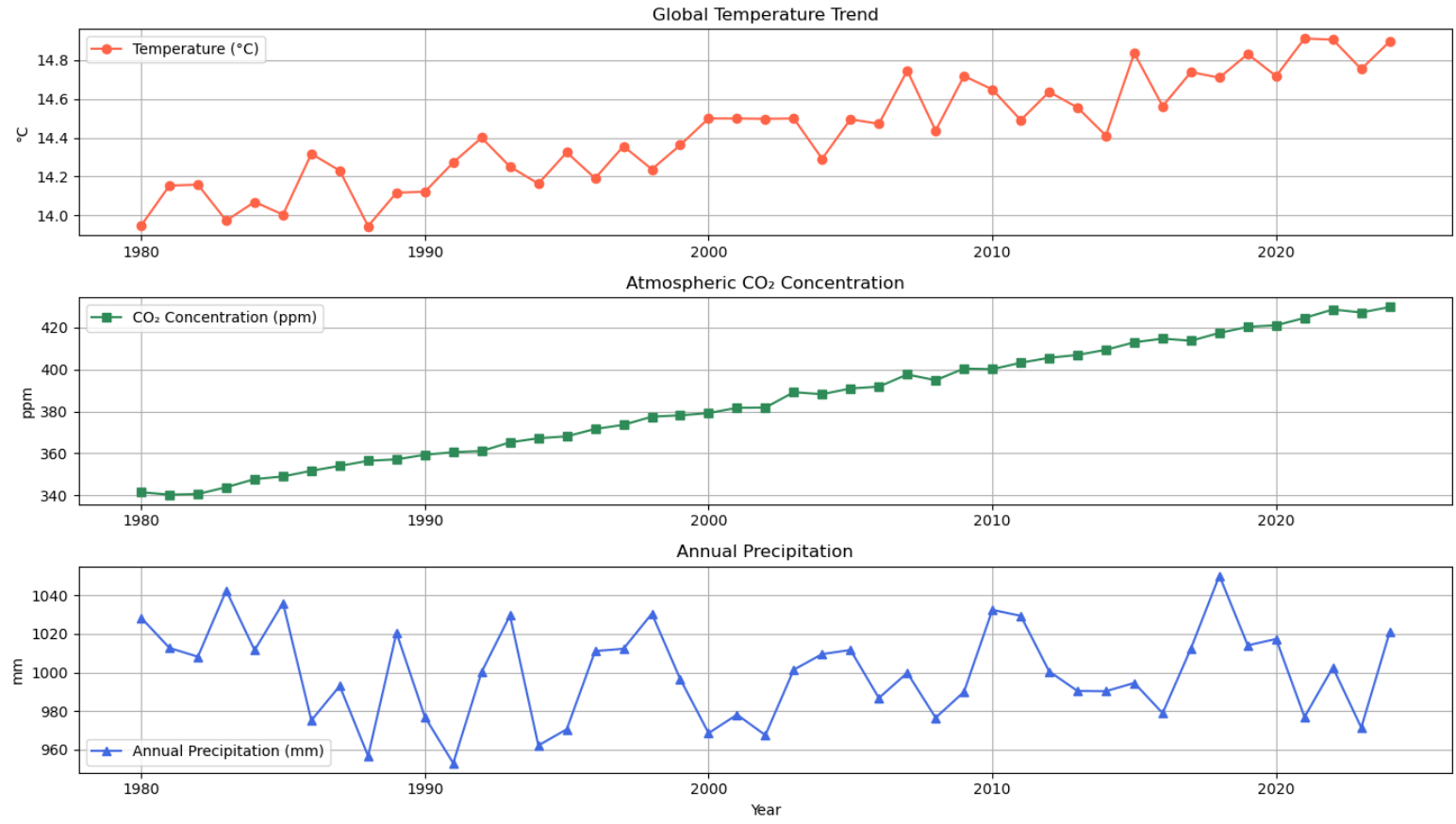
4️⃣ 어떤 시사점이 있나요?

- 데이터와 모델이 결합되면 기후변화가 **단순한 경고**가 아니라
  현실적인 농업 계획 수립의 기준이 될 수 있습니다.
- 변화하는 환경 속에서도 **데이터로 결정하는 힘**이 있음을 배웁니다.

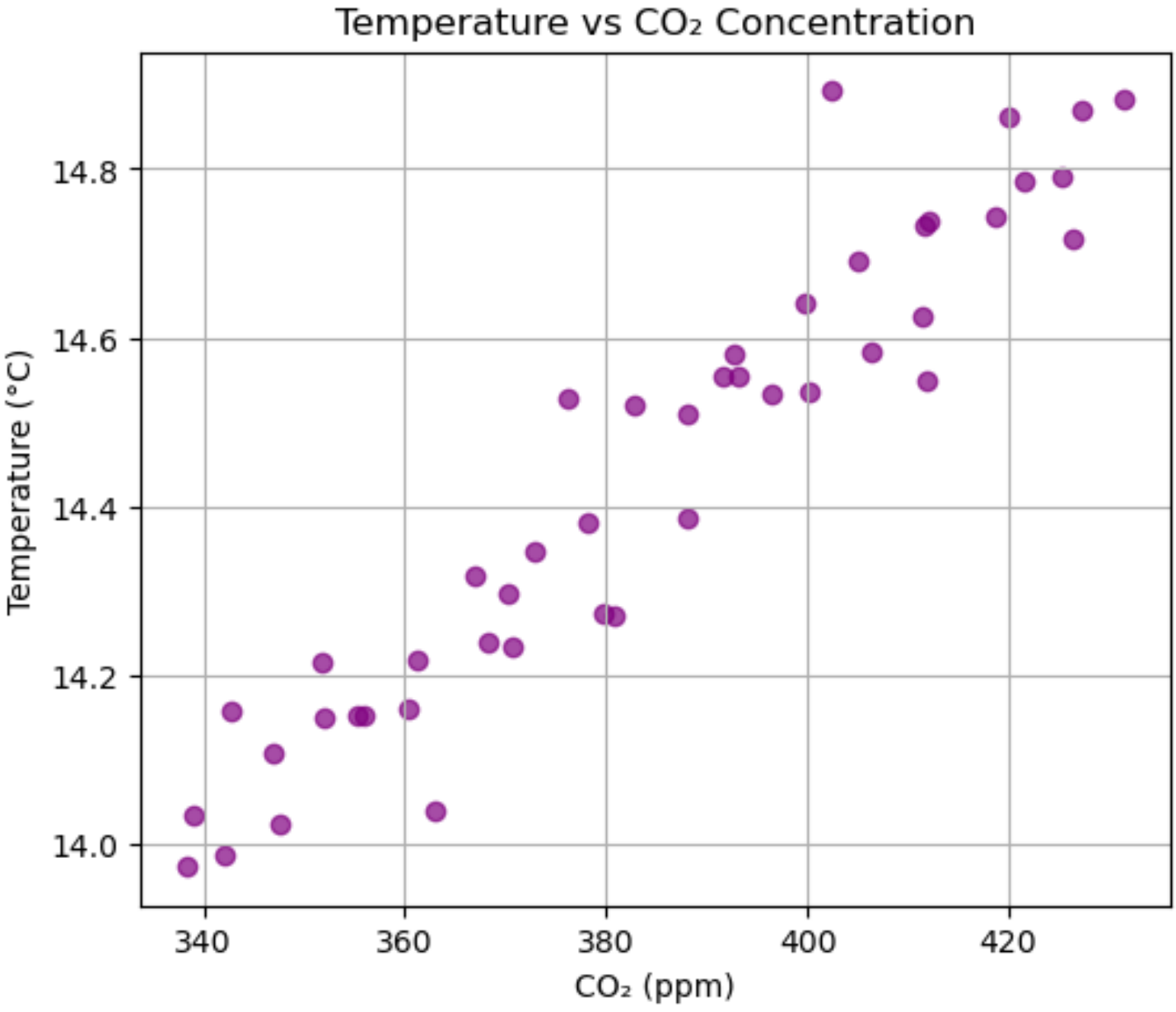
[1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 연도: 1980~2024
years = np.arange(1980, 2025)

# 모의 기온 (약간의 상승 추세 + 노이즈)
temperature = 14.0 + 0.02 * (years - 1980) + np.random.normal(0, 0.1, len(years))

# 모의 CO₂ 농도 (상승 추세 + 노이즈)
co2 = 338 + 2.1 * (years - 1980) + np.random.normal(0, 1.5, len(years))

# 모의 강수량 (큰 추세 없이 약간의 주기적 패턴)
precipitation = 1000 + 10 * np.sin((years - 1980) * 0.5) + np.random.normal(0, 20, len(years))

# DataFrame 구성
df = pd.DataFrame({
    'Year': years,
    'Temperature': temperature,
    'CO2': co2,
    'Precipitation': precipitation
})

df.head()
```

	Year	Temperature	CO2	Precipitation
0	1980	13.947628	341.493231	1028.333411
1	1981	14.154047	340.302471	1012.670364
2	1982	14.158693	340.605319	1008.130502
3	1983	13.973630	343.864594	1042.445004
4	1984	14.069647	347.716859	1011.508647

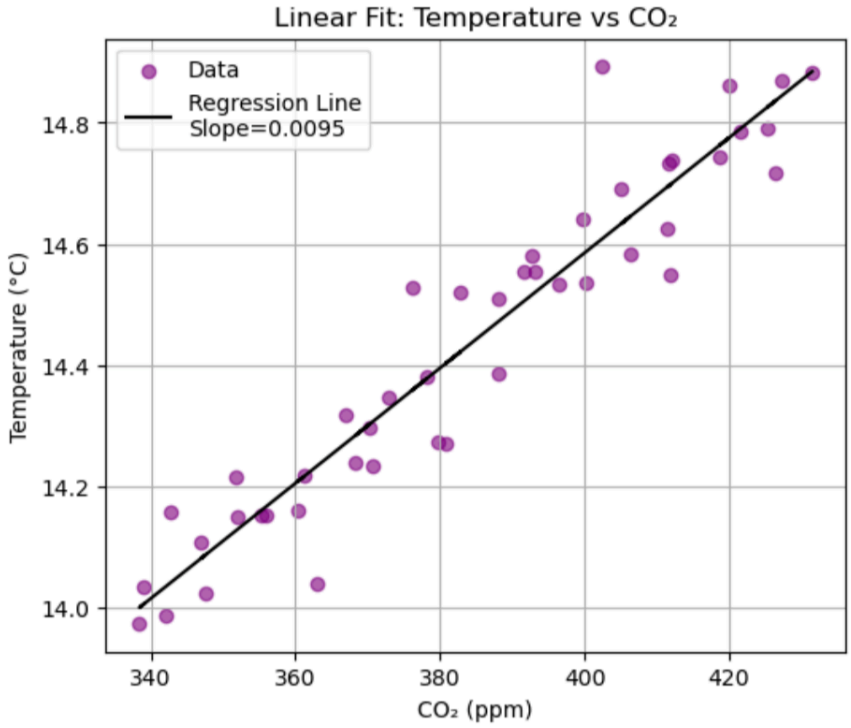
[2]:
```
plt.figure(figsize=(14, 8))

# 기온
plt.subplot(3, 1, 1)
plt.plot(df['Year'], df['Temperature'], marker='o', color='tomato', label='Temperature (°C)')
plt.title('Global Temperature Trend')
plt.ylabel('°C')
plt.grid(True)
plt.legend()

# CO2
plt.subplot(3, 1, 2)
plt.plot(df['Year'], df['CO2'], marker='s', color='seagreen', label='CO₂ Concentration (ppm)')
plt.title('Atmospheric CO₂ Concentration')
plt.ylabel('ppm')
plt.grid(True)
plt.legend()

# 강수량
plt.subplot(3, 1, 3)
plt.plot(df['Year'], df['Precipitation'], marker='^', color='royalblue', label='Annual Precipitation (mm)')
plt.title('Annual Precipitation')
plt.xlabel('Year')
plt.ylabel('mm')
plt.grid(True)
plt.legend()

plt.tight_layout()
plt.show()
```



[3]:
```
plt.figure(figsize=(6, 5))
plt.scatter(df['CO2'], df['Temperature'], c='purple', alpha=0.7)
plt.title('Temperature vs CO₂ Concentration')
plt.xlabel('CO₂ (ppm)')
plt.ylabel('Temperature (°C)')
plt.grid(True)
plt.show()
```



[5]:
```
# 회귀선 계산
slope, intercept = np.polyfit(df['CO2'], df['Temperature'], 1)

# 시각화
plt.figure(figsize=(6, 5))
plt.scatter(df['CO2'], df['Temperature'], c='purple', alpha=0.6, label='Data')
plt.plot(df['CO2'], slope * df['CO2'] + intercept, color='black', label=f'Regression Line\nSlope={slope:.4f}')
plt.title('Linear Fit: Temperature vs CO₂')
plt.xlabel('CO₂ (ppm)')
plt.ylabel('Temperature (°C)')
plt.grid(True)
plt.legend()
plt.show()
```



📈 결과 해석¶

- 연도별 **기온은 꾸준히 상승**하고 있으며,
- 이와 함께 **CO_2 농도도 꾸준히 증가**하고 있습니다.
- 기온과 CO_2 간에는 **선형적 양의 상관관계**가 보입니다.
- 이는 **지구 온난화의 원인으로 CO_2 증가가 중요한 역할**을 함을 보여주는 시뮬레이션 예입니다.

🥕 경남 시군단위 작물 추천 시뮬레이션

이 노트북은 경상남도 내 5개 시군에 대해 기온, 강수량 등의 기후 데이터를 기반으로
작물(밭작물, 과수류)에 적합한 지역을 선형회귀 모델로 예측합니다.

- 기후 조건은 가상 데이터이며,
- 작물별로 '선호 조건'을 정의하여 선호 점수를 계산합니다.

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 시군 이름
cities = ['Changwon', 'Jinju', 'Miryang', 'Hamyang', 'Geoje']

# 연도
years = np.arange(2010, 2025)

# 지역별 기온 및 강수량 생성 (도시별 편차 존재)
np.random.seed(42)
data = []
for city in cities:
    temp_base = 13.5 + np.random.rand() * 1.5  # 도시마다 온도 베이스 다름
    rain_base = 1000 + np.random.rand() * 200

    for year in years:
        temperature = temp_base + 0.03 * (year - 2010) + np.random.normal(0, 0.1)
        rainfall = rain_base + 5 * np.sin((year - 2010) * 0.4) + np.random.normal(0, 20)
        data.append([city, year, round(temperature, 2), round(rainfall, 2)])

df = pd.DataFrame(data, columns=["City", "Year", "Temperature", "Rainfall"])
df.head()
```

	City	Year	Temperature	Rainfall
0	Changwon	2010	14.13	1220.60
1	Changwon	2011	14.07	1187.41
2	Changwon	2012	14.28	1209.08
3	Changwon	2013	14.10	1205.65
4	Changwon	2014	14.14	1185.83

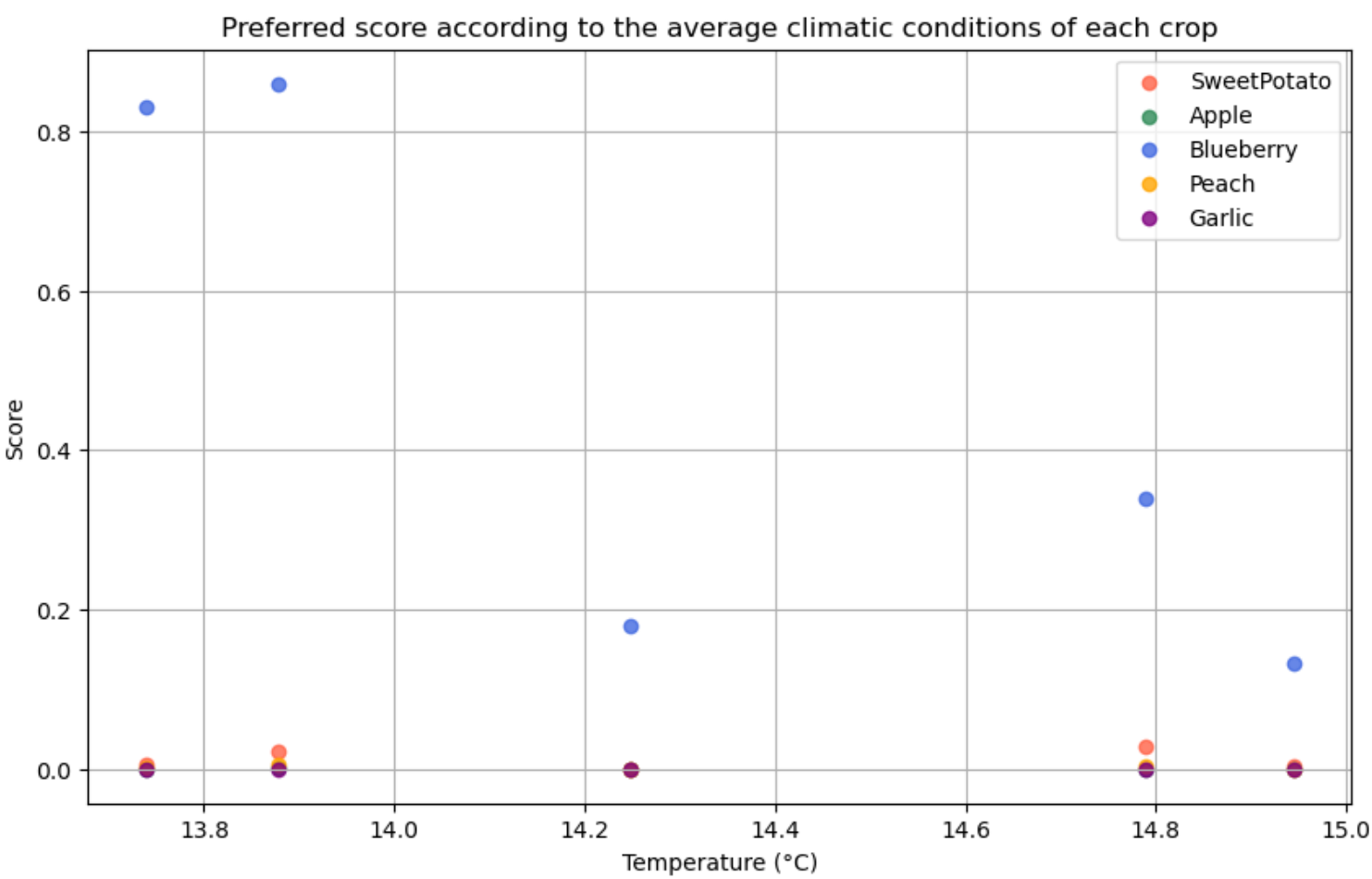
[3]:
```python
# 작물 종류와 선호 조건 (기온, 강수량 범위)
crops = {
    'SweetPotato':  {'T_opt': 16,  'R_opt': 1000},
    'Apple':        {'T_opt': 14,  'R_opt': 900},
    'Blueberry':    {'T_opt': 13.5,'R_opt': 1100},
    'Peach':        {'T_opt': 15.5,'R_opt': 950},
    'Garlic':       {'T_opt': 13,  'R_opt': 850}
}
```

[4]:
```python
# 작물 선호 점수 계산 함수 (온도와 강수량에서의 거리 기반)
def crop_score(temp, rain, T_opt, R_opt):
    t_score = np.exp(-((temp - T_opt)**2) / 2)
    r_score = np.exp(-((rain - R_opt)**2) / 5000)
    return t_score * r_score

# 작물별 점수 계산
for crop, params in crops.items():
    df[crop] = df.apply(lambda row: crop_score(row['Temperature'], row['Rainfall'], params['T_opt'], params['R_opt']), axis=1)

df.head()
```

[4]:

	City	Year	Temperature	Rainfall	SweetPotato	Apple	Blueberry	Peach	Garlic
0	Changwon	2010	14.13	1220.60	0.000010	1.171100e-09	0.044721	1.707062e-07	6.210766e-13
1	Changwon	2011	14.07	1187.41	0.000138	6.668061e-08	0.184418	4.573867e-06	7.292777e-11
2	Changwon	2012	14.28	1209.08	0.000036	4.845142e-09	0.068297	7.024516e-07	2.784677e-12
3	Changwon	2013	14.10	1205.65	0.000035	7.643697e-09	0.089601	7.898802e-07	5.632969e-12
4	Changwon	2014	14.14	1185.83	0.000178	7.933727e-08	0.186718	5.856504e-06	8.350139e-11

[5]:
```
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# 도시별 평균값 및 작물별 예측
grouped = df.groupby('City').mean(numeric_only=True).reset_index()

plt.figure(figsize=(10, 6))

colors = ['tomato', 'seagreen', 'royalblue', 'orange', 'purple']
for i, crop in enumerate(crops.keys()):
    plt.scatter(grouped['Temperature'], grouped[crop], label=crop, alpha=0.8, color=colors[i]

plt.title("Preferred score according to the average climatic conditions of each crop")
plt.xlabel("Temperature (°C)")
plt.ylabel("Score")
plt.legend()
plt.grid(True)
plt.show()
```



[6]:
```
# 최신년도 데이터에서 작물 선호도가 가장 높은 작물 추천
latest_year = df['Year'].max()

recommend = df[df['Year'] == latest_year].copy()
recommend['BestCrop'] = recommend[crops.keys()].idxmax(axis=1)
recommend[['City', 'Year', 'Temperature', 'Rainfall', 'BestCrop']]
```

	City	Year	Temperature	Rainfall	BestCrop
14	Changwon	2024	14.42	1224.03	Blueberry
29	Jinju	2024	13.99	1071.95	Blueberry
44	Miryang	2024	14.98	1095.08	Blueberry
59	Hamyang	2024	15.05	1157.66	Blueberry
74	Geoje	2024	13.86	1128.15	Blueberry

탐구 질문: 기후와 농업의 연결 고리를 찾아라!

❓ 1. CO_2는 정말 기온 상승의 원인일까?

데이터를 보면 CO_2가 증가하면서 온도도 증가했어요.
그런데 **"함께 증가했다"는 사실이 꼭 "원인-결과"를 의미하진 않죠.**

Q.
어떤 추가 정보가 있다면 "CO_2가 기온 상승의 원인이다"라고 더 확신할 수 있을까요?

❓ 2. 선형 회귀 없이도 가능한 작물 추천?

우리는 작물 추천을 위해 복잡한 회귀모델 없이
그냥 '선호 조건과의 거리'를 계산했어요.

Q.
이 방식이 어떤 점에서 유리할까요?
또, 어떤 상황에서는 **회귀 모델이 더 적절할 수 있을까요?**

❓ 3. 기후 변화에 대응하는 농부가 되어보자!

2050년, 기온이 지금보다 1.5도 더 올라가고
강수량은 지역마다 10%씩 줄어들 전망입니다.

🧬 유전자 분석과 시각화

🎯 목표

- DNA 염기서열 데이터를 분석하고 **시각적으로 표현하는 방법**을 배웁니다.
- 코딩을 통해 생명과학의 핵심인 **염기, RNA, 단백질 번역 과정**을 직접 체험합니다.
- 데이터 기반 생물학 탐구에 필요한 **사고력과 표현력**을 키우는 것이 목적입니다.

1️⃣ DNA 염기서열, 그게 뭐예요?

DNA는 `A`, `T`, `C`, `G` 4가지 염기로 구성된 **정보의 코드**입니다.
사람의 몸, 식물의 뿌리, 바이러스의 증식 방식까지 모두 이 코드를 바탕으로 결정됩니다.

이번 실습에서는 아래와 같은 작업을 합니다:

- `ATCGATCG...` 처럼 생긴 염기서열을 시각화 (색으로 표현)
- 각 염기의 빈도(등장 횟수)를 그래프로 분석
- 코돈(3개씩 짝지은 염기) → 아미노산으로 번역하는 과정 확인

2️⃣ 무엇을 시각화하나요?

- **염기 막대 시각화**: A는 초록, T는 빨강... 염기서열이 시각적으로 보입니다.
- **빈도 히스토그램**: 어떤 염기가 얼마나 나왔는지 직관적으로 확인
- **RNA로 전사, 단백질로 번역**: DNA 정보가 실제 생명 활동으로 바뀌는 과정을 살펴봅니다.
- **코돈-아미노산 매핑**: `ATG → M`, `GCC → A` 같이 변환 과정을 눈으로 확인
-

3️⃣ 이걸 어디에 쓸 수 있나요?

- **생물학적 유사성 비교**: 특정 염기 조합이 있으면 특정 병에 취약할 수도 있습니다.
- **유전 기반 질병 예측**: 암, 유전병, 알러지 등과 관련된 유전자 패턴을 예측
- **개인 맞춤 의학**: 약을 맞춤 추천하려면 유전자 해석이 필수입니다.

🧪 파이썬 없이 표현하는 알고리즘 예시

✅ 예시 1: 로지스틱 회귀로 유전자 변이 예측

```
# 유전자에 특정 염기 조합이 있는지(1) 없는지(0) → 이진 분류
# 입력: [A,T,C,G] 출현 수 → 출력: 병 위험도 (0~1)

score = 0.2*A_count + 0.5*G_count - 0.3*T_count
probability = 1 / (1 + exp(-score))  # 시그모이드 함수
```

➡️ 유전자 정보로 특정 질병에 걸릴 확률을 예측

✅ 예시 2: KNN으로 유사 유전자 분류

```
# 새로운 유전자 시퀀스가 기존의 어떤 그룹(예: 질병 보유/비보유)에 가까운지 비교

if distance_to_group_A < distance_to_group_B:
    label = "Group A"
else:
    label = "Group B"
```

➡️ 데이터 기반으로 **가장 가까운 이웃**을 기준으로 분류

✅ 예시 3: 염기서열의 경사하강법 적용 (고급)

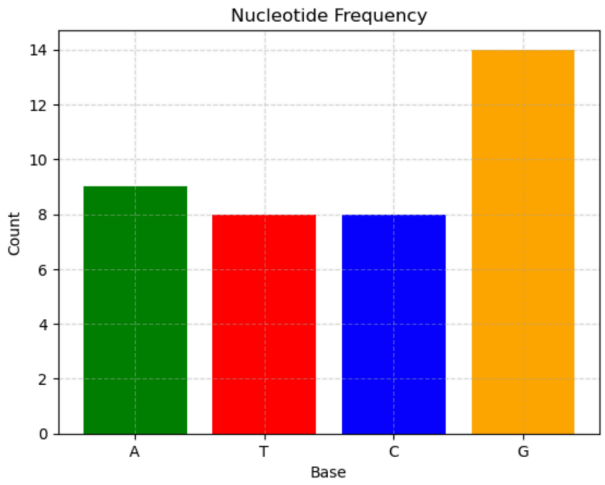
```
# 목표: 특정 염기 비율에서 예측 정확도가 높아지도록 가중치 조정

w_A = w_A - alpha * dL/dw_A  # A 염기의 영향력 업데이트
```

➡️ 오차를 줄이기 위한 최적화 (머신러닝 모델 훈련 방식)

📌 핵심 정리

주제	적용
염기 빈도 시각화	생물학적 특성 이해
RNA 전사, 단백질 번역	생명 정보 흐름 이해
머신러닝 개념 응용	질병 예측, 유전자 분류

[2]:
```python
# 염기서열 데이터
sequence = "ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG"

# 염기 빈도 분석 및 시각화
from collections import Counter
import matplotlib.pyplot as plt

count = Counter(sequence)
bases = ['A', 'T', 'C', 'G']
values = [count.get(b, 0) for b in bases]

plt.bar(bases, values, color=['green', 'red', 'blue', 'orange'])
plt.title("Nucleotide Frequency")
plt.xlabel("Base")
plt.ylabel("Count")
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
```



🧬 유전자 염기서열을 이용한 질병 예측

– 로지스틱 회귀, 선형 회귀, 경사하강법의 관계 정리

선형 회귀 (Linear Regression)

선형 회귀는 입력값 x에 대해 연속적인 숫자 y를 예측하는 모델입니다. 예측 수식은 다음과 같습니다:

$$\hat{y} = w \cdot x + b$$

여기서 w는 기울기(가중치), b는 절편입니다. 예를 들어 $x = 5, w = 10, b = 20$일 때, 예측값은 $\hat{y} = 70$이 됩니다. 이 모델은 결과값이 실수 범위 전체를 가질 수 있어 예측값이 제한 없이 나옵니다.

로지스틱 회귀 (Logistic Regression)

로지스틱 회귀는 선형 회귀의 결과를 확률로 변환하여 분류 문제(예/아니오)를 해결합니다. 우선 선형 결합을 계산합니다:

$$z = w \cdot x + b$$

그다음 시그모이드 함수로 확률로 바꿉니다:

$$\hat{y} = \frac{1}{1 + e^{-z}}$$

이때 $\hat{y}$는 항상 0과 1 사이의 값이 되며, 확률처럼 해석할 수 있습니다. 일반적으로 $\hat{y} \geq 0.5$면 "예"로 분류하고, 그렇지 않으면 "아니오"로 간주합니다.

시그모이드 함수 (Sigmoid Function)

시그모이드 함수는 입력값 z를 확률값으로 변환해 주는 함수입니다:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

z가 커질수록 출력은 1에 가까워지고, 작아질수록 0에 가까워지며, 항상 0과 1 사이의 값을 갖습니다. 이로 인해 로지스틱 회귀는 연속적인 출력값을 확률로 해석할 수 있습니다.

경사하강법 (Gradient Descent)

경사하강법은 주어진 데이터에 가장 잘 맞는 w와 b를 찾기 위한 최적화 알고리즘입니다. 손실 함수(loss)를 계산하고, 그 기울기를 따라 손실이 작아지는 방향으로 파라미터를 조금씩 갱신합니다. 이 과정을 반복하여 오차가 최소가 되는 지점을 찾습니다. 로지스틱 회귀의 손실 함수는 다음과 같습니다:

$$L = -\frac{1}{n} \sum_{i=1}^{n} \left[y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$$

업데이트 방식은 다음과 같습니다:

$$w \leftarrow w - \alpha \cdot \frac{\partial L}{\partial w}, \quad b \leftarrow b - \alpha \cdot \frac{\partial L}{\partial b}$$

전체 흐름 요약

1. 선형 회귀는 수치 예측 모델이다.
2. 로지스틱 회귀는 선형 회귀의 출력을 확률로 바꾸고, 기준값(예: 0.5)을 기준으로 분류를 수행한다.
3. 경사하강법은 이 모델의 가중치 w, 절편 b를 최적화하여 학습하게 만든다.

전체 예측 과정은 다음과 같다:
입력값 $x \rightarrow z = w \cdot x + b \rightarrow \hat{y} = \sigma(z) \rightarrow$ 확률 해석 $\rightarrow$ 분류(예: 질병 있음/정상)

유전자 예측과의 연결

입력으로 A, T, C, G 염기의 개수를 받고, 각 염기에 대응하는 가중치 w_A, w_T, w_C, w_G와 절편 b를 사용하여 점수를 계산한다:

$$z = w_A \cdot x_A + w_T \cdot x_T + w_C \cdot x_C + w_G \cdot x_G + b$$

$$\hat{y} = \frac{1}{1 + e^{-z}}$$

$\hat{y} \geq 0.5$이면 질병 유전자일 가능성이 높다고 판단할 수 있다.

학생 확인 질문

1. 선형 회귀와 로지스틱 회귀의 차이점은?
2. 시그모이드 함수는 어떤 역할을 하나요?
3. 경사하강법이 없다면 모델 학습이 가능한가요?
4. 가중치가 양수일 때와 음수일 때 결과 해석은 어떻게 다른가요?
5. 절편 b가 크거나 작으면 예측에 어떤 영향을 주나요?
6. 예측 확률이 0.82일 때 어떤 판단을 내릴 수 있나요?

✅ 예시 1: 로지스틱 회귀로 유전자 변이 예측

이 활동에서는 간단한 **로지스틱 회귀 모델**을 사용하여 유전자 염기서열이 **질병과 관련 있는지 없는지**를 예측해봅니다.
우리는 **A, T, C, G 염기의 등장 횟수**를 기준으로 질병의 위험도를 계산하고, **시그모이드 함수**를 통해 결과를 확률로 바꿉니다.

📌 1. 로지스틱 회귀

로지스틱 회귀는 아래처럼 생긴 수학 공식으로 분류 문제(예: 질병인지 아닌지)를 해결합니다.

$$\hat{y} = \frac{1}{1 + e^{-(w_1x_1 + w_2x_2 + \cdots + w_nx_n + b)}}$$

- (x_1, x_2, \dots): 입력값 (여기서는 염기의 개수)
- (w_1, w_2, \dots): 가중치 (각 염기의 중요도)
- (b): 절편 (bias)
- ({y}_hat): 결과 확률 (0 ~ 1 사이)
- 0.5보다 크면 "질병 유전자", 작으면 "정상 유전자"

🧪 2. 예제 코드

```
from collections import Counter
import math
# 🌱 1. 학습된 가중치 (임의로 지정된 값)
w = {'A': 0.4, 'T': -0.6, 'C': 0.3, 'G': 0.5}
bias = -2.0  # 절편
# 🧬 2. 염기서열 (입력값)
sequence = "ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG"
count = Counter(sequence.upper())  # 각 염기의 개수 계산
# 📊 3. 선형 결합 계산: w*x + b
score = sum(w[base] * count.get(base, 0) for base in 'ATCG') + bias
# 🔁 4. 시그모이드 함수 적용 → 0~1 사이 확률로 변환
prob = 1 / (1 + math.exp(-score))
# 🧾 5. 결과 출력
print(f"예측된 질병 위험도 (0~1): {prob:.3f}")
if prob > 0.5:
    print("⚠️ 질병 관련 유전자일 가능성이 높습니다.")
else:
    print("✅ 정상 유전자일 가능성이 높습니다.")
```

🧪 3. 예제의 실행 결과:

```
예측된 질병 위험도 (0~1): 0.998
⚠️ 질병 관련 유전자일 가능성이 높습니다.
```

🧠 실습 개념 + 핵심 코드 이해 질문¶

✅ 실습 1: 염기서열 빈도 시각화¶

Q1. 그래프에서 가장 많이 등장한 염기는 무엇이었나요?
→ 이 염기가 많다는 건 무슨 의미일까요?

Q2. `Counter(sequence)`라는 코드의 역할은 무엇인가요?
→ 왜 직접 세지 않고 이걸 썼을까요?

Q3. `plt.bar(bases, values, ...)` 명령은 어떤 시각화를 위한 코드인가요?

✅ 실습 2: 로지스틱 회귀 스타일 유전자 예측¶

Q4. `score = ...` 줄에서 계산한 값은 무엇을 의미하나요?
→ 어떤 염기는 위험도를 높이고, 어떤 염기는 낮추는 데 기여했나요?

Q5. `probability = 1 / (1 + math.exp(-score))` 이 수식은 어떤 개념을 나타내나요?
→ 왜 이렇게 복잡한 식을 써야 할까요?

Q6. `if prob > 0.5:` 아래에 조건문이 있어요.
→ 어떤 판단을 기준으로 '질병 위험'이라고 결정하나요?

✅ 실습 3: KNN 스타일 유전자 분류¶

Q7. `distance(seq_count, group_A)`라는 함수는 무엇을 계산하나요?
→ 어떤 방식으로 유전자 간 유사도를 판단하나요?

Q8. `sum((a - b)**2 for ...)` 부분은 왜 제곱을 하나요?
→ 단순한 차이 말고 제곱을 쓰는 이유는 무엇인가요?

Q9. 만약 A 그룹과 B 그룹 모두 거리 차이가 비슷하다면, 어떤 방식으로 결정하는 것이 좋을까요?
→ KNN 알고리즘을 조금 더 복잡하게 바꿔보자면?

🧩 여러분은 지금 데이터 분석과 머신러닝 개념을
유전자라는 진짜 생물학 정보에 적용해보고 있습니다.
코드의 흐름, 수식의 의미, 결과의 해석 모두 함께 생각해보세요!

🧠 탐구 질문: 유전자의 비밀을 풀어라!¶

❓ 1. 왜 세 글자씩 끊어서 읽을까?¶

DNA는 4글자인데, **왜 코돈은 항상 3개씩 묶어서 해석**할까요?

- Q. 2글자씩 읽으면 안 되는 이유는 무엇일까요?

❓ 2. 같은 단백질을 만드는 다른 코돈이 존재하는 이유는?¶

`GCU`, `GCC`, `GCA`는 모두 **Alanine**이라는 단백질로 번역돼요.
이건 **코돈이 달라도 같은 결과가 나올 수 있음**을 뜻하죠.

- Q. 이 유연한 구조는 어떤 장점이 있을까요?

❓ 3. 염기 빈도를 보고 유전자 기능을 유추할 수 있을까?¶

어떤 유전자 염기서열에서 G와 C가 유난히 많다면?
어떤 패턴이 반복된다면?

- Q. 염기 비율과 기능 간의 관련성을 어떻게 찾을 수 있을까요?

“호기심을 놓지 않는 사람은 언제나 배우는 사람입니다.”

🌾 농지 환경과 작물 생산량 시뮬레이션

🎯 목표

우리는 가상의 농지 데이터를 바탕으로
환경 요인이 작물 생산량에 어떤 영향을 미치는지를 시각적으로 확인하고,
수학적 모델을 이용해 생산량을 예측하는 활동을 수행합니다.

1️⃣ 왜 이런 실험을 하나요?

- 농업은 환경에 크게 영향을 받는 산업입니다.
- 데이터를 기반으로 하면 **감이 아니라 계산으로** 작물 선택과 배치를 결정할 수 있습니다.
- 우리는 실제 측정값이 아닌 **가상의 데이터**로 실험하지만,
  여기에 사용된 방법은 **현실 문제에도 적용 가능**합니다.

2️⃣ 첫 번째 실험: 기온과 강수량이 생산량에 미치는 영향

- 10개의 가상 농지에 대해 경도(x), 위도(y), 온도, 강수량을 무작위로 생성
- 생산량은 다음과 같은 수식으로 계산:

```
Yield = 2 * Temperature + 0.5 * Rainfall + noise
```

- 이를 지도 위 점으로 시각화하여, 어느 농지가 더 생산적인지 직관적으로 확인

3️⃣ 두 번째 실험: 산지 밭의 작물별 특성과 환경 요인

- 5곳의 '산지 밭'에 작물 종류를 할당 (고구마, 사과, 블루베리 등)
- 각 밭마다 환경 조건 설정:
  - 일조 시간, 토양 수분, 경사도, 고도
- 작물마다 가중치를 다르게 설정하고, 아래와 같은 모델로 생산량 예측:

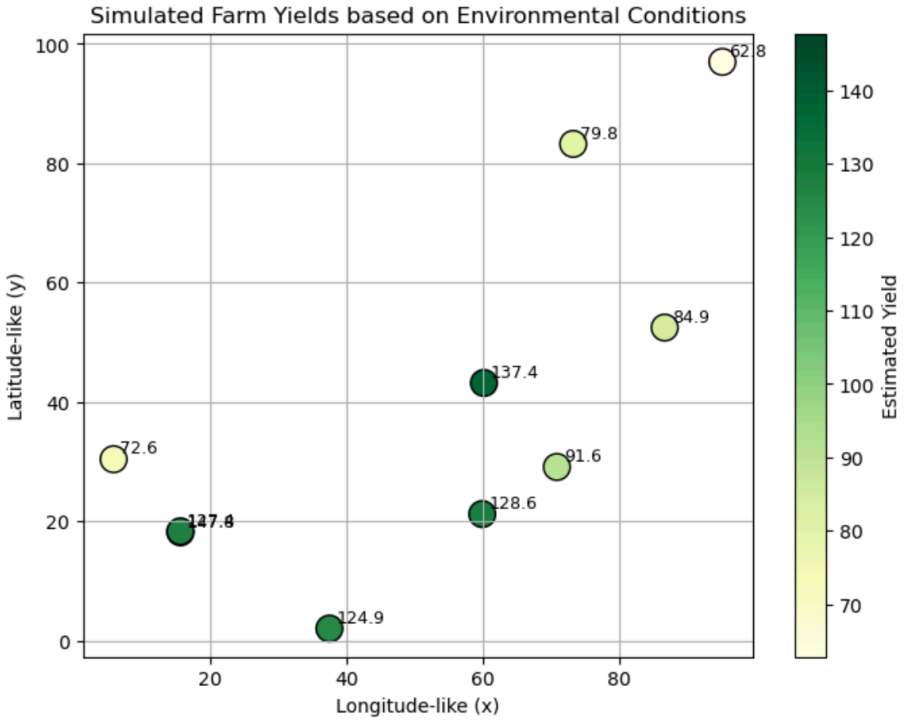
```
Yield = 10 * Sunlight + 0.5 * Moisture
- 1.0 * Slope - 0.1 * Altitude
+ CropBonus + noise
```

- 예측값은 지도에 색상(빨강~초록)으로 표시되어 시각적 이해를 돕습니다.

4️⃣ 이 활동을 통해 배우는 것

- **현실 세계 문제도 수학적 모델로 표현할 수 있다**는 감각
- 단순한 변수 조합이 작물 생산량을 설명해 줄 수 있음
- 기후 변화나 환경 조건 변화가 **농업 생산에 어떻게 영향을 주는지** 탐색 가능
- 데이터 + 지도 + 시각화의 결합은 매우 강력한 도구임

[1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 1. 가상의 농지 데이터 생성
np.random.seed(42)
num_farms = 10
x = np.random.uniform(0, 100, size=num_farms)   # 경도 유사값
y = np.random.uniform(0, 100, size=num_farms)   # 위도 유사값
temperature = np.random.uniform(15, 35, size=num_farms)  # 섭씨
rainfall = np.random.uniform(50, 200, size=num_farms)    # mm

# 2. 작물 생산량: 임의의 선형 모델 + noise
# 생산량 = 2*온도 + 0.5*강수량 + noise
true_weights = np.array([2.0, 0.5])
features = np.column_stack((temperature, rainfall))
noise = np.random.normal(0, 10, size=num_farms)
yield_est = features @ true_weights + noise

# 3. DataFrame 생성
df = pd.DataFrame({
    'x': x,
    'y': y,
    'Temperature': temperature,
    'Rainfall': rainfall,
    'Yield': yield_est
})

# 4. 시각화
plt.figure(figsize=(8, 6))
sc = plt.scatter(df['x'], df['y'], c=df['Yield'], cmap='YlGn', s=200, edgecolor='k')
plt.colorbar(sc, label='Estimated Yield')
plt.xlabel('Longitude-like (x)')
plt.ylabel('Latitude-like (y)')
plt.title('Simulated Farm Yields based on Environmental Conditions')

# 값 출력 (디버깅용)
for i in range(len(df)):
    plt.text(df['x'][i]+1, df['y'][i]+1, f"{df['Yield'][i]:.1f}", fontsize=9)

plt.grid(True)
plt.show()
```



👩‍🌾 농장의 위치와 환경조건이 수확량에 어떤 영향을 줄까?

이 실험에서는 가상의 농장 10개를 만들어서, 각 농장의 위치와 환경 조건(온도, 강수량)에 따라 수확량(**Yield**)이 어떻게 달라지는지를 시각적으로 분석해 봅니다.

📌 코드 설명 (한 줄씩 이해하기)

1. 데이터 만들기: 농장의 위치와 환경

python 코드
```
x = np.random.uniform(0, 100, size=num_farms)  # 농장의 경도(가로축 위치)
y = np.random.uniform(0, 100, size=num_farms)  # 농장의 위도(세로축 위치)
```
👉 각 농장의 위치를 지도로 표현할 수 있도록 무작위로 지정합니다.

python 코드
```
temperature = np.random.uniform(10, 35, size=num_farms)  # 온도(°C)
rainfall = np.random.uniform(50, 200, size=num_farms)    # 강수량(mm)
```
🌡️ 농장 주변의 기온과 비의 양을 무작위로 생성합니다.

2. 수확량 계산: 환경 조건의 영향

python 코드
```
true_weights = np.array([2.0, 0.5])  # 온도는 2배, 강수는 0.5배 중요
features = np.column_stack((temperature, rainfall))
noise = np.random.normal(0, 10, size=num_farms)
yield_est = features @ true_weights + noise
```
📈 실제 수확량은 다음과 같이 계산합니다:

$$\text{수확량} = 2 \cdot T + 0.5 \cdot R + \varepsilon$$

🔍 **기호 설명:**

- T: 온도 (Temperature)
- R: 강수량 (Rainfall)
- ε: 잡음 (noise), 예측할 수 없는 오차 항

> 이 식은 수확량이 **온도와 강수량의 가중합(linear combination)**으로 결정되며, 현실적인 요소를 반영하기 위해 **오차항** ε을 포함하고 있다는 점에서 **선형 회귀 모델**의 전형적인 형태입니다.