TABLE OF CONTENTS

ABSTRACT

The project titled “Crop Yield and Weather Data Analyser” focuses on analysing the relationship between crop productivity and regional weather parameters using Python libraries such as NumPy and Pandas. Agriculture is a cornerstone of the Indian economy, and crop yield is significantly influenced by climatic conditions including rainfall, temperature, and humidity. With increasing climate variability and unpredictable weather patterns, data analysis serves as a powerful tool to understand agricultural trends, predict outcomes, and support informed decision-making.

This project utilizes two major datasets — a Crop Yield Database, containing details about crop type, region, production, and year, and a Region Weather Database, which records factors such as rainfall, temperature, and humidity. These datasets are merged on common attributes like region and year to facilitate combined analysis. Before analysis, the data undergoes preprocessing using Pandas techniques such as cleaning, handling missing values, and ensuring consistent data types. This step ensures the datasets are accurate, uniform, and ready for computation.

Using NumPy, the project performs various numerical and statistical analyses, including calculating mean, variance, and correlation coefficients. These computations help uncover relationships between climatic conditions and crop yields. For instance, some crops show a strong positive correlation with rainfall, while others are more affected by temperature variations. Grouping, aggregation, and visualization techniques are applied to derive insights such as identifying regions with stable yield patterns and crops most vulnerable to climate change.

The findings emphasize the role of data science in agriculture, offering valuable insights for farmers, policymakers, and researchers. The project demonstrates how data-driven analysis can identify optimal growing conditions, forecast low-yield periods, and guide sustainable farming strategies. Furthermore, this analysis can be extended to predictive modeling using machine learning, paving the way for future advancements in precision agriculture and improved food security through data-informed crop management.

.

CHAPTER 1

INTRODUCTION

1.1 Overview

Agriculture is one of the most vital sectors of any nation's economy, particularly in developing countries like India, where a large portion of the population depends on farming for livelihood. However, agricultural productivity is highly influenced by various environmental and climatic factors such as temperature, rainfall, humidity, and soil conditions. The dynamic and unpredictable nature of weather patterns often poses significant challenges to maintaining consistent crop yields. In recent years, technological advancements in data science and computational tools have opened new avenues for addressing these challenges through data-driven decision-making.

The Crop Yield and Weather Data Analyser project is designed to explore and understand the complex relationship between crop yield and weather parameters using Python's powerful data analysis libraries — NumPy and Pandas. This project leverages two core datasets: the *Crop Yield Database,* containing information on crop production across various regions and the *Region Weather Database,* containing temperature, rainfall, and humidity data for the same regions. By integrating these datasets, the project aims to analyse how different weather conditions affect crop yield over time and across geographic locations.

Using Pandas, the data is preprocessed, cleaned, and merged to ensure uniformity and accuracy. The library's robust data manipulation capabilities allow filtering, grouping, and aggregating data efficiently. NumPy, on the other hand, facilitates complex numerical computations and statistical analyses. With these tools, correlations between weather parameters and crop yield can be calculated, helping identify which factors have the most significant impact on productivity.

Beyond raw computation, visualisation plays a crucial role in this project. Graphs, plots, and tables help represent trends clearly, making it easier to interpret the effects of temperature and rainfall fluctuations on various crops. The insights derived can be valuable for farmers, researchers, and policymakers. They can be used to optimise planting schedules, select suitable crops for particular regions, and implement effective irrigation or resource management strategies.

In essence, this project demonstrates how computational tools can be applied in the agricultural domain to transform data into knowledge. It bridges the gap between environmental studies and computer science, showcasing the importance of data analytics in ensuring sustainable agricultural practices and food security.

.

1.2 Motivation

The motivation behind the Crop Yield and Weather Data Analyser project arises from the growing need to make agriculture more resilient, data-driven, and sustainable in the face of changing climate conditions. Agriculture in many regions, especially in India, is heavily dependent on monsoons and other climatic variables. Unpredictable weather patterns, droughts, and floods often result in drastic variations in crop yield, leading to financial instability for farmers and threats to food security. These recurring issues highlight the necessity for a systematic, analytical approach to understanding how weather impacts agricultural productivity.

With the advancement of computational technologies, vast amounts of agricultural and meteorological data are now available. However, without proper analysis, this data remains underutilised. The motivation for this project is to harness this data effectively using NumPy and Pandas, two of Python's most efficient tools for data manipulation and numerical computation. By analysing historical crop yield data alongside corresponding weather data, it becomes possible to detect patterns, trends, and correlations that can inform better agricultural decisions.

Another key motivation is to showcase the real-world applicability of data science in addressing problems that have social and economic relevance. Students and researchers often use datasets for theoretical learning, but applying these tools to real agricultural data brings deeper understanding and practical experience. It not only strengthens technical skills but also encourages interdisciplinary thinking, merging environmental awareness with computational intelligence.

The project also serves as a foundation for future research in predictive modelling, where machine learning algorithms could forecast future crop yields based on projected weather conditions. Such predictive systems can be invaluable for farmers and policymakers, allowing them to plan resources, manage risks, and enhance productivity.

Ultimately, the motivation is rooted in the belief that data has the power to transform agriculture. By uncovering relationships between weather conditions and crop yields, we can promote smarter farming, reduce losses, and move toward a more sustainable and technologically empowered agricultural system.

1.3 Problem Statement

Agricultural productivity is deeply dependent on environmental factors, yet the relationship between crop yield and weather conditions is not always straightforward or well-understood. Farmers and agricultural planners often struggle to anticipate yield variations because of the lack of accessible, data-driven insights. The primary problem lies in the absence of a systematic approach to analyse and correlate crop yield with regional weather patterns. Traditional methods of observation and prediction are insufficient in today's context, where climate change has made weather conditions increasingly erratic.

The problem extends beyond individual farmers to the broader agricultural ecosystem. Policymakers lack reliable analytical models that can guide them in planning subsidies, irrigation projects, or crop insurance schemes. Researchers need integrated tools to understand long-term trends and climate impact on agriculture. This lack of analytical integration between weather and yield data results in inefficiencies and missed opportunities for improvement.

This project seeks to address this gap by developing a data analysis system that can merge and analyse crop

yield and weather data using Python libraries such as NumPy and Pandas. The objective is to understand how temperature, rainfall, and humidity affect the productivity of various crops across different regions. Data from multiple sources often contains inconsistencies, missing values, or irregular formats, which adds to the challenge. Hence, the project also focuses on efficient data cleaning, preprocessing, and transformation to make the datasets suitable for analysis.

Another significant problem is the interpretation of results. Even when data is available, presenting it in an understandable format for non-technical users is difficult. The project, therefore, incorporates visualisation techniques to display results clearly, making insights accessible to decision-makers and researchers alike.

In summary, the main problem addressed by the project is the lack of integrated analysis between crop yield and weather conditions, caused by unorganised data, limited computational analysis, and the absence of simple tools that can help stakeholders derive insights for agricultural improvement.

1.4 Objectives

The Crop Yield and Weather Data Analyser project is designed with multiple objectives that combine data analysis, statistical reasoning, and practical application. The primary goal is to use NumPy and Pandas to study how weather patterns influence crop yields across regions and time periods.

The main objectives include:

1. Data Collection and Integration: Gather datasets on crop yield and regional weather conditions, and merge them based on common features such as year and region.

2. Data Preprocessing: Clean the datasets to remove missing, duplicate, or inconsistent data using Pandas' built-in functions, ensuring that the analysis is based on accurate and reliable information.

3. Exploratory Data Analysis (EDA): Apply Pandas operations such as grouping, filtering, and aggregating to uncover trends and relationships. Use NumPy for mathematical and statistical analysis, such as computing means, variances, and correlations.

4. Visualisation: Represent the data graphically using plots and charts to make the analysis intuitive and easier to interpret.

5. Insight Generation: Determine how changes in rainfall, temperature, and humidity influence crop yield and identify which crops are most sensitive to specific weather conditions.

6. Practical Application: Provide recommendations for improving agricultural planning and management based on the analysis results.

Beyond these, another objective is to give students hands-on experience with Python's data analysis ecosystem. The project allows learners to practically apply classroom concepts of arrays, data frames, indexing, and data manipulation in a meaningful real-world context. It demonstrates how coding and data analytics can be used for problem-solving beyond the academic environment.

The ultimate objective of the project is to promote data-driven agriculture. By using statistical and computational tools, the project aims to bridge the gap between raw data and actionable insights.

CHAPTER 2

SYSTEM ARCHITECTURE

The system architecture of the Crop Yield and Weather Data Analyser is designed to provide a structured flow of data — from collection and preprocessing to analysis and visualisation — ensuring that each stage is logically connected and efficiently executed. The architecture follows a modular, layered approach, making it simple, scalable, and easy to implement using Python libraries such as NumPy and Pandas. The system focuses on two primary datasets: the Crop Yield Database and the Region Weather Database, which are integrated and processed to derive analytical insights on the correlation between weather parameters and crop productivity.

2.1 Overview of the Architecture

At a high level, the system architecture consists of five main layers:

1. **Data Collection Layer**
2. **Data Preprocessing Layer**
3. **Data Integration Layer**
4. **Data Analysis Layer**
5. **Visualisation and Output Layer**

Each layer performs a specific role and communicates with the next stage through structured data formats like CSV or DataFrames. This modular design ensures flexibility and allows the system to be easily expanded for larger datasets or more complex analyses in the future.

2.2 Data Collection Layer

The first layer of the system architecture focuses on collecting and organising data from reliable sources. The Crop Yield Database includes attributes such as crop name, region, area harvested, total production, and yield per hectare. The Region Weather Database contains information such as temperature, rainfall, humidity, and year for corresponding regions. The data is typically stored in CSV or Excel formats, which are easily importable into Python using the Pandas `read_csv()` or `read_excel()` functions.

This layer ensures that both datasets are gathered from consistent and authentic sources to maintain data reliability. The raw data is stored locally or in cloud-based storage for easy access during preprocessing.

2.3 Data Preprocessing Layer

Once the data is collected, it often contains inconsistencies, missing values, or outliers that can affect the accuracy of the analysis. Hence, the preprocessing layer is critical. Using Pandas, operations like `dropna()`, `fillna()`, `replace()`, and `astype()` are applied to clean the data and standardize the format. Duplicates are removed, columns are renamed for uniformity, and data types are converted appropriately (e.g., integers for years, floats for rainfall, strings for regions).

In addition, data normalization or scaling may be performed where required to ensure that all variables are measured on comparable scales. This layer transforms the raw datasets into a clean, structured format ready for integration.

2.4 Data Integration Layer

This layer merges the Crop Yield Database and Region Weather Database using common attributes such as Region and Year. By using the Pandas function `merge()`, both datasets are combined into a single DataFrame. This enables a comprehensive view of how specific weather parameters in a region for a particular year influenced the yield of various crops.

Data integration helps in establishing meaningful relationships between datasets that were previously independent. It also allows for region-wise and time-series analyses, enabling trend identification over multiple years.

2.5 Data Analysis Layer

Once the integrated dataset is ready, the analysis layer uses **NumPy** and **Pandas** for statistical computations and data exploration. Key operations include:

- Computing the **mean**, **median**, **variance**, and **standard deviation** of crop yield.
- Calculating **correlation coefficients** between yield and weather factors.
- Performing **grouping and aggregation** operations using Pandas to analyse trends across different regions and years.

This layer forms the core of the system, providing insights such as:

- Which weather parameter (temperature, rainfall, or humidity) most influences yield?
- Which crops perform best under specific climate conditions?
- Regional performance comparisons over multiple years.

The combination of NumPy's numerical efficiency and Pandas' tabular flexibility ensures fast computation and easy interpretation of large datasets.

2.6 Visualisation and Output Layer

The final layer focuses on data visualisation and interpretation. The results obtained from the analysis are represented graphically using Pandas visualisation tools or Matplotlib. Common plots include:

- **Line charts** for showing yield trends over time.
- **Bar charts** comparing yields across regions or crops.
- **Scatter plots** for visualising correlations between weather and yield variables.

Visualization helps users easily understand the data insights without requiring deep statistical knowledge. The results can be exported as images, tables, or CSV files for reports and further analysis.

CHAPTER 3

METHODOLOGY

The methodology of the Crop Yield and Weather Data Analyzer outlines the systematic process followed to collect, clean, analyze, and interpret data in order to understand the relationship between crop yield and weather conditions. The entire project is based on Python programming, utilizing two powerful libraries — NumPy for numerical computations and Pandas for data handling and analysis. The approach focuses on transforming raw agricultural and weather data into meaningful insights through structured, step-by-step processing.

The methodology can be divided into six major stages:

1. Data Collection
2. Data Preprocessing
3. Data Integration
4. Data Analysis
5. Visualisation and Interpretation
6. Result Generation

3.1 Data Collection

The first step involves gathering relevant datasets that form the foundation of the analysis. Two datasets are used:

- **Crop Yield Database** — contains information such as crop type, region, year, area harvested, total production, and yield per hectare.
- **Region Weather Database** — includes parameters like rainfall (in mm), temperature (in °C), and humidity levels across different regions and years.

These datasets are collected from authentic sources such as government agriculture departments, meteorological agencies, or open-data repositories like Kaggle or data.gov.in. The data is usually stored in CSV or Excel format for easy import into Python using `pandas.read_csv()` or `pandas.read_excel()`.

This step ensures that the collected data is comprehensive enough to represent a wide range of climatic and agricultural conditions across multiple regions.

3.2 Data Preprocessing

Raw data often contains **missing values, duplicates, and inconsistencies**, which can distort the results. Hence, preprocessing is an essential step before analysis. In this stage, Pandas functions such as:

- `dropna()` (to remove missing entries),
- `fillna()` (to replace missing data with suitable values),
- `drop_duplicates()` (to remove repeated records), and
- `astype()` (to correct data types)
  are used to clean and standardise the datasets.

Column names are standardised for consistency, and non-numeric values are converted where necessary. Data ranges are validated to ensure logical accuracy (for example, rainfall cannot be negative). This stage ensures that both datasets are reliable and ready for analysis.

3.3 Data Integration

After cleaning, the Crop Yield Database and Region Weather Database are combined into a single dataset. This is achieved using `pandas.merge()` function, which joins the two datasets based on common attributes — typically *Region* and *Year*.

This integration allows the system to correlate crop yield values directly with their respective weather conditions. For example, rainfall data for a particular region and year can now be analysed against the yield of rice or wheat from the same time and place.

By combining the datasets, a unified DataFrame is created, which forms the basis for all subsequent analytical operations.

3.4 Data Analysis

Once the integrated dataset is ready, the core data analysis process begins. Here, NumPy and Pandas are used to perform various statistical and mathematical operations, such as:

- Calculating **mean, median, mode, variance, and standard deviation** for crop yield and weather parameters.
- Identifying **correlations** between yield and variables like rainfall, temperature, and humidity using **`pandas.corr()`**.
- Performing **grouping and aggregation** operations (using **`groupby()`**) to analyse yield trends by region, crop type, or year.

NumPy enhances computational performance by enabling fast operations on large data arrays. These calculations help uncover key relationships, such as whether rainfall is positively correlated with yield or whether high temperature reduces productivity for certain crops.

This step forms the analytical backbone of the system, transforming raw data into interpretable results.

3.5 Visualisation and Interpretation

To make the results more understandable, data visualisation techniques are used. Graphs and charts are generated using Pandas' built-in plotting functions and Matplotlib.
Common visualisations include:

- **Line graphs** to show crop yield changes over time.
- **Bar charts** comparing yields between regions or crops.
- **Scatter plots** to depict correlations between weather variables and yield.

Visual representations make it easier to identify trends and anomalies. For instance, a line chart may reveal that crop yields drop in years with below-average rainfall, or that certain regions maintain stability despite weather fluctuations.

Visualisation bridges the gap between technical analysis and practical interpretation, allowing users to quickly grasp complex relationships.

3.6 Result Generation

The final stage compiles all analytical and visual results into meaningful insights and summaries. These results help answer key questions such as:

- Which crop is most affected by rainfall variation?
- Which region has the most stable yield trends?
- What weather parameters most strongly influence productivity?

The results can be exported as tables, charts, or CSV files for inclusion in reports. The insights gained can support decision-making in crop planning, irrigation scheduling, and resource allocation.

CHAPTER 4

IMPLEMENTATION

CODE:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import logging
from typing import Tuple, Dict, Any

# Visualization style
sns.set_theme(style="whitegrid")

# Logging configuration
logging.basicConfig(level=logging.INFO, format="%(asctime)s %(levelname)s: %(message)s")

# Helper formatters
def fmt(x, digits: int = 2):
try:
if pd.isna(x): return "NA"
if isinstance(x, float): return f"{x:.{digits}f}"
return str(x)
except Exception:
return str(x)

def print_block(title: str):
sep = "=" * max(len(title) + 8, 60)
print("\n" + sep)
print(f" {title}")
print(sep + "\n") 28.
def print_subtitle(sub: str):
print("\n" + "-" * max(len(sub) + 4, 40))
print(f" {sub}")
print("-" * max(len(sub) + 4, 40) + "\n")
from google.colab import files
import os

print("📁 Upload CropYieldDB.csv when prompted:")
uploaded_crop = files.upload()
```

```
41.       raise FileNotFoundError("CropYieldDB.csv was not
uploaded.") 42.
43.   print("📁 Upload RegionWeatherDB.csv when prompted:")
44.           uploaded_weather =
files.upload() 45.
46.   # Check if RegionWeatherDB.csv was uploaded
47.   if "RegionWeatherDB.csv" not in uploaded_weather:
48.           raise FileNotFoundError("RegionWeatherDB.csv was not
uploaded.") 49.
50.   CROP_PATH = "CropYieldDB.csv" # Assuming the uploaded file names are exactly this
51.           WEATHER_PATH = "RegionWeatherDB.csv" # Assuming the uploaded file names are
exactly this 52.
53. WEATHER_NROWS = 10000
54.
55.   def load_data(crop_path=CROP_PATH, weather_path=WEATHER_PATH, weather_nrows=WEATHER_NROWS):
56.       crop = pd.read_csv(crop_path)
57.       weather = pd.read_csv(weather_path, low_memory=False).head(weather_nrows)
58.       logging.info(f"Loaded crop rows={crop.shape[0]} | weather rows={weather.shape[0]}")
59.           return crop,
weather 60.
61.   def preprocess(crop: pd.DataFrame, weather: pd.DataFrame):
62.       crop = crop.copy(); weather = weather.copy()
63.       crop['State_clean'] = crop.get('State', '').astype(str).str.strip().str.upper()
64.       if 'state_name' in weather.columns:
65.           weather['State_clean'] = weather['state_name'].astype(str).str.strip().str.upper()
66.       elif 'State' in weather.columns:
67.           weather['State_clean'] = weather['State'].astype(str).str.strip().str.upper()
68.       else:
69.           weather['State_clean'] = weather.get('state_name',
'').astype(str).str.strip().str.upper() 70.
71.       for c in ['Area_Hectares','Production_Tonnes','Yield_KgPerHa','Year']:
72.           if c in crop.columns:
73.           crop[c] = pd.to_numeric(crop[c], errors='coerce')
74.
75.       for c in ['Crop','Season']:
76.           if c in crop.columns:
77.           crop[c] = crop[c].astype(str).str.strip()
78.
79.       if 'date' in weather.columns:
80.           weather['date_dt'] = pd.to_datetime(weather['date'], dayfirst=True, errors='coerce')
81.       else:
82.           weather['date_dt'] =
pd.NaT 83.
84.       rain_candidates = ['actual','rain','Rainfall','rainfall','Actual','rfs']
85.       rain_col = next((c for c in rain_candidates if c in weather.columns), None)
86.       weather['Rain_mm'] = pd.to_numeric(weather[rain_col], errors='coerce').fillna(0.0) if rain_col else 0.0
87.       weather['Year'] = weather['date_dt'].dt.year
88.           crop =
crop.drop_duplicates() 89.
90.       logging.info("Preprocessing complete.")
91.           return crop,
weather 92.
93.   crop, weather = load_data()
94.   crop, weather = preprocess(crop, weather)
95.           print_block("UNIT I — NumPy
Fundamentals") 96.
97. yields = crop['Yield_KgPerHa'].dropna().to_numpy()
98.
99.   print_subtitle("1) Introduction to NumPy — Yield Array")
```

```
print(f"Sample yields (first 12): {yields[:12]}")
print(f"Mean yield: {fmt(np.nanmean(yields))} | Std dev: {fmt(np.nanstd(yields))}")

plt.figure(figsize=(8,4))
sns.histplot(yields, bins=30, kde=True)
plt.title("Yield Distribution — UNIT I")
plt.xlabel("Yield (Kg/Ha)")
plt.tight_layout(); plt.show()

print_subtitle("2) Creating Arrays / Basic Operations")
a,b = np.array([2,4,6]), np.array([1,2,3])
print("A + B =", (a+b))
print("A * B =", (a*b))

print_subtitle("3) Reshaping & Slicing Example")
m = np.arange(1,25).reshape(4,6)
print("Matrix second row:", m[1])
print("Matrix third column:", m[:,2])

print_block("UNIT II — Aggregation & Transformations")
crop_sy = (crop.groupby(['State_clean','Year'], as_index=False)
           .agg(Total_Area=('Area_Hectares','sum'),
                Total_Prod=('Production_Tonnes','sum'),
                Avg_Yield=('Yield_KgPerHa','mean')))
weather_sy = (weather.groupby(['State_clean','Year'], as_index=False)
              .agg(Yearly_Rain=('Rain_mm','sum')))
state_year = pd.merge(crop_sy, weather_sy, on=['State_clean','Year'], how='left')
state_year['Prod_per_Ha'] = (state_year['Total_Prod']*1000)/state_year['Total_Area'].replace(0,np.nan)

print_subtitle("Aggregated Sample")
display(state_year.head())

state_summary = (state_year.groupby('State_clean',as_index=False)
             .agg(Mean_Yield=('Avg_Yield','mean'),
                  Mean_Rain=('Yearly_Rain','mean'))
             .sort_values('Mean_Yield',ascending=False))

print_subtitle("Top States by Mean Yield")
display(state_summary.head())

plt.figure(figsize=(10,5))
sns.barplot(data=state_summary.head(10), x='Mean_Yield', y='State_clean', palette='viridis')
plt.title("Top 10 States by Mean Yield (Kg/Ha)")
plt.tight_layout(); plt.show()

print_block("UNIT III — Pandas Fundamentals")
mean_yield_crop = (crop.groupby('Crop',as_index=False)
                   .agg(Mean_Yield=('Yield_KgPerHa','mean'),
                        Total_Area=('Area_Hectares','sum'),
                        Total_Prod=('Production_Tonnes','sum'))
                   .sort_values('Mean_Yield',ascending=False))

print_subtitle("Top 10 Crops by Mean Yield")
display(mean_yield_crop.head(10))

top_area = (crop.groupby('Crop',as_index=False)
            .agg(Area_Ha=('Area_Hectares','sum'))
            .sort_values('Area_Ha',ascending=False)
            .head(8))
```

```
159.
160. plt.figure(figsize=(12,5))
161. plt.subplot(1,2,1)
162. plt.pie(top_area['Area_Ha'], labels=top_area['Crop'], autopct='%1.1f%%', startangle=140)
163. plt.title("Crop Area Share (Top 8)")
164. plt.subplot(1,2,2)
165. sns.barplot(data=top_area, x='Area_Ha', y='Crop', palette='magma')
166. plt.title("Top 8 Crops by Cultivated Area")
167. plt.tight_layout(); plt.show()
168.         print_block("UNIT IV — Advanced
Pandas") 169.
170. pivot = pd.pivot_table(state_year, index='State_clean', columns='Year',
171.                         values='Avg_Yield', aggfunc='mean')
172.
173. plt.figure(figsize=(12,6))
174. sns.heatmap(pivot.fillna(0).iloc[:30,-10:], cmap='coolwarm', cbar_kws={'label':'Avg Yield (Kg/Ha)'})
175. plt.title("Yield Heatmap — Recent Years by State")
176. plt.tight_layout(); plt.show()
177.
178. # Yield trend slope per state
179. slopes=[]
180. for st,g in state_year.groupby('State_clean'):
181.     g2=g.dropna(subset=['Year','Avg_Yield']).sort_values('Year')
182.     if g2.shape[0]>=3:
183.         slope=np.polyfit(g2['Year'],g2['Avg_Yield'],1)[0]
184.         slopes.append((st,slope))
185. slopes_df=pd.DataFrame(slopes,columns=['State_clean','slope'])
186. print_subtitle("Top Yield Trend States (Slope per Year)")
187. display(slopes_df.sort_values('slope',ascending=False).head(5))
188.         print_block("UNIT V — Applied Analysis &
Visualizations") 189.
190. # Productivity metrics
191. prod_df = state_year.dropna(subset=['Prod_per_Ha'])
192. avg_prod = prod_df['Prod_per_Ha'].mean()
193. print_subtitle("Productivity Summary")
194.         print(f"Average productivity (Kg/Ha):
{fmt(avg_prod)}") 195.
196. plt.figure(figsize=(9,5))
197. sns.boxplot(data=prod_df.sample(min(1000,len(prod_df))), x='Prod_per_Ha')
198. plt.title("Productivity Distribution (Kg/Ha)")
199. plt.tight_layout(); plt.show()
200.
201. # Representative state trend
202. rep_state = state_summary['State_clean'].iloc[0]
203. rep_df = state_year[state_year['State_clean']==rep_state].sort_values('Year')
204. print_subtitle(f"Representative State Trend: {rep_state}")
205. display(rep_df.head())
206.
207. plt.figure(figsize=(10,5))
208. sns.lineplot(data=rep_df, x='Year', y='Yearly_Rain', marker='o', label='Rainfall (mm)')
209. sns.lineplot(data=rep_df, x='Year', y='Avg_Yield', marker='s', label='Yield (Kg/Ha)', color='orange')
210. plt.title(f"{rep_state}: Rainfall vs Yield Trend")
211. plt.legend(); plt.tight_layout(); plt.show()
212. print_block("DETAILED SUMMARY & INSIGHTS")
213.
214. corrs=[]
215. for st,g in state_year.groupby('State_clean'):
216.     g2=g.dropna(subset=['Yearly_Rain','Avg_Yield'])
217.     if g2.shape[0]>=3:
```

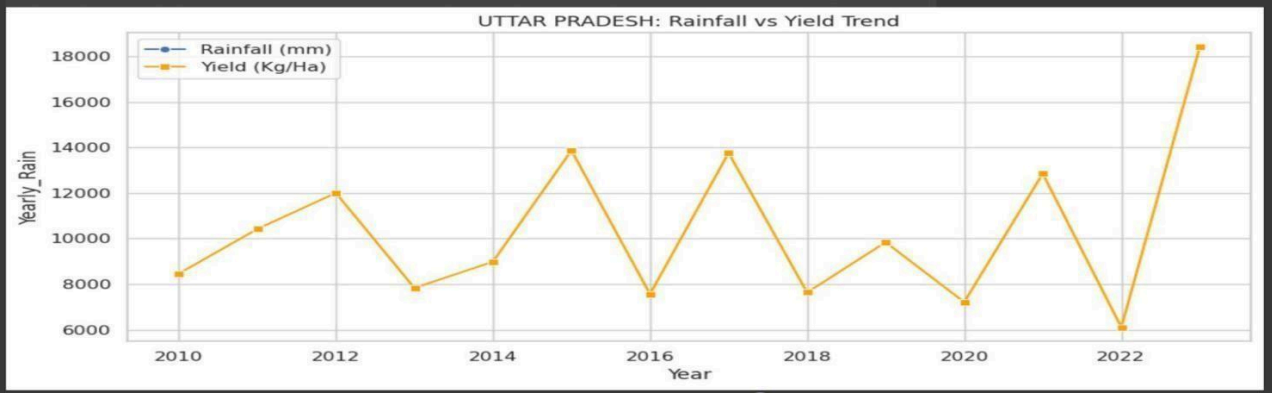
```
        corrs.append((st,g2['Yearly_Rain'].corr(g2['Avg_Yield'])))
corr_df=pd.DataFrame(corrs,columns=['State_clean','corr']).sort_values('corr',ascending=False)

print_subtitle("Top Positive Rain–Yield Correlations")
display(corr_df.head(5))
print_subtitle("Top Negative Rain–Yield Correlations")
display(corr_df.tail(5))


print("\nInterpretation:")
print("- High yield often aligns with moderate rainfall, but relationship varies regionally.")
print("- Productivity differences suggest targeted crop–state strategies.")
print("- Positive yield trends in several states indicate improved practices or technology adoption.")
```

CHAPTER 5

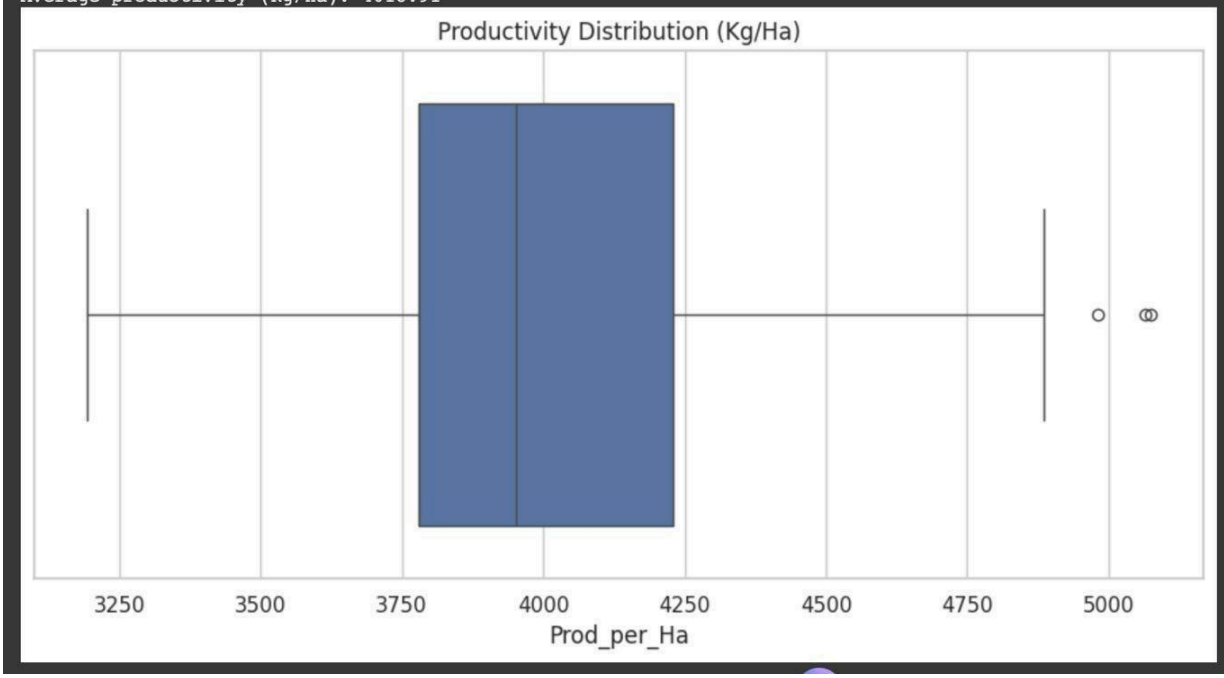
RESULTS

```
============================================================
  DETAILED SUMMARY & INSIGHTS
============================================================


----------------------------------------
  Top Positive Rain–Yield Correlations
----------------------------------------

  State_clean  corr

----------------------------------------
  Top Negative Rain–Yield Correlations
----------------------------------------

  State_clean  corr

Interpretation:
- High yield often aligns with moderate rainfall, but relationship varies regionally.
- Productivity differences suggest targeted crop–state strategies.
- Positive yield trends in several states indicate improved practices or technology adoption.
```

	State_clean	Year	Total_Area	Total_Prod	Avg_Yield	Yearly_Rain	Prod_per_Ha
112	UTTAR PRADESH	2010	1687380.76	6226730.87	8461.553662	NaN	3690.175340
113	UTTAR PRADESH	2011	1463059.74	5855885.37	10420.253929	NaN	4002.492318
114	UTTAR PRADESH	2012	2249268.51	9244511.09	12003.303176	NaN	4110.007786
115	UTTAR PRADESH	2013	1702221.03	7566688.22	7833.333188	NaN	4445.185488
116	UTTAR PRADESH	2014	1722206.94	8082573.40	8984.005972	NaN	4693.148780



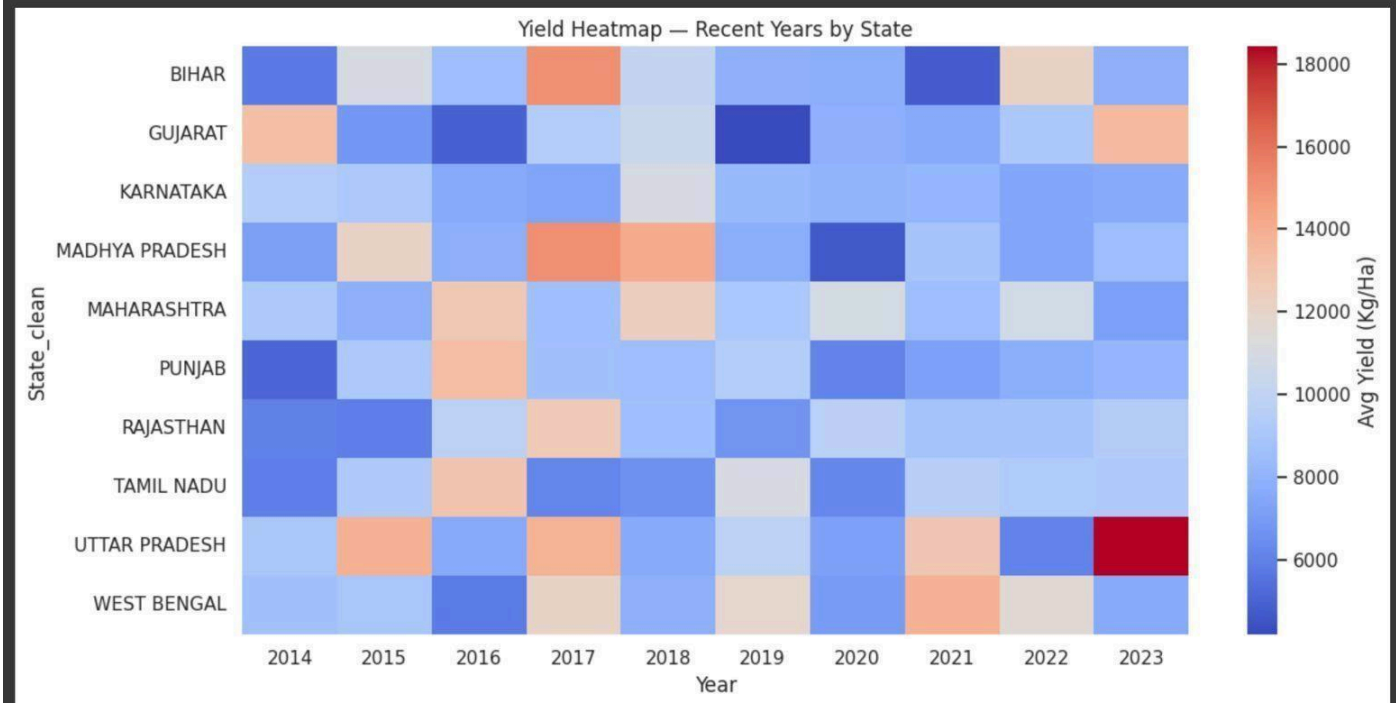
```
============================================================
  UNIT V — Applied Analysis & Visualizations
============================================================


--------------------------------------
  Productivity Summary
--------------------------------------

Average productivity (Kg/Ha): 4018.91
```




```
------------------------------------------
  Top Yield Trend States (Slope per Year)
------------------------------------------
```

	State_clean	slope
8	UTTAR PRADESH	169.344683
0	BIHAR	52.949338
9	WEST BENGAL	33.847917
4	MAHARASHTRA	-37.332592
3	MADHYA PRADESH	-39.675533

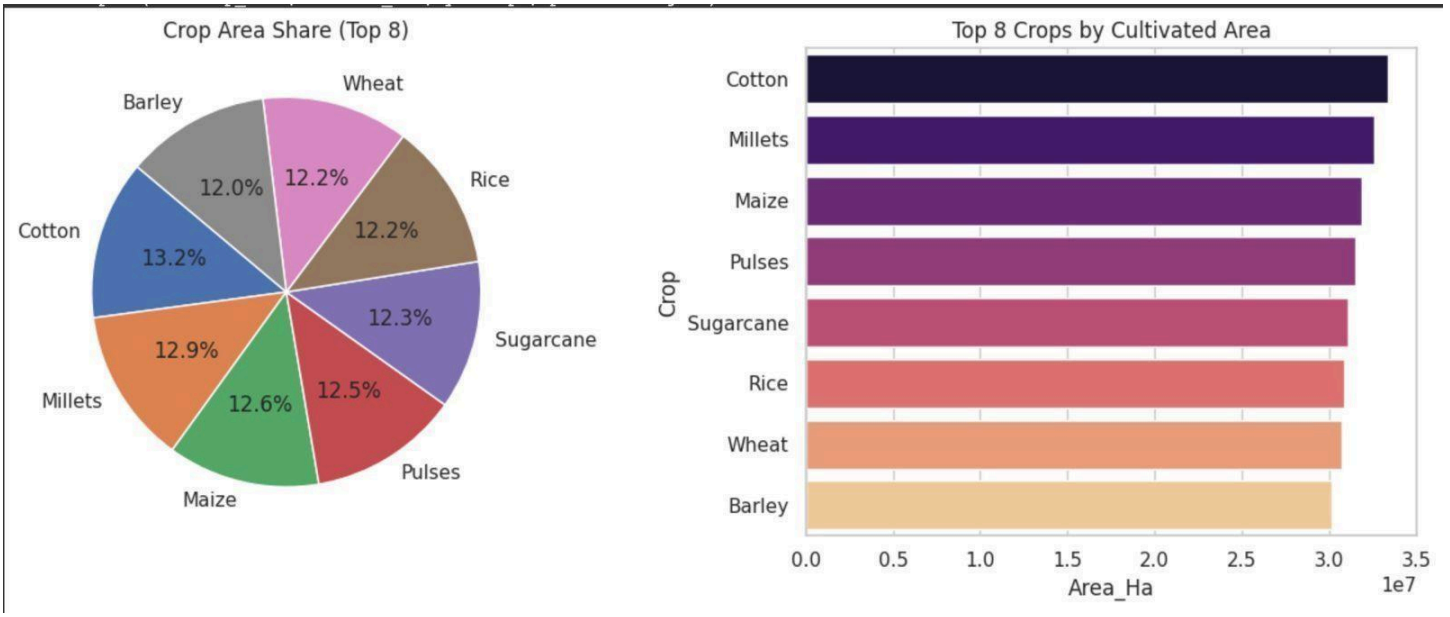
```
UNIT IV — Advanced Pandas
============================================================
```




```
============================================================
  UNIT III — Pandas Fundamentals
============================================================


----------------------------------------
  Top 10 Crops by Mean Yield
----------------------------------------
```

	Crop	Mean_Yield	Total_Area	Total_Prod
7	Wheat	9675.698568	30712837.25	1.204506e+08
0	Barley	9546.659344	30172043.65	1.239688e+08
3	Millets	9475.651062	32589036.17	1.312010e+08
6	Sugarcane	9194.136885	31116151.42	1.253033e+08
4	Pulses	9176.439249	31499157.52	1.244528e+08
2	Maize	9065.586837	31873366.67	1.287279e+08
1	Cotton	9028.977280	33380297.20	1.315887e+08
5	Rice	8848.539435	30868915.58	1.225083e+08

```
/tmp/ipython-input-652412587.py:22: FutureWarning:
```



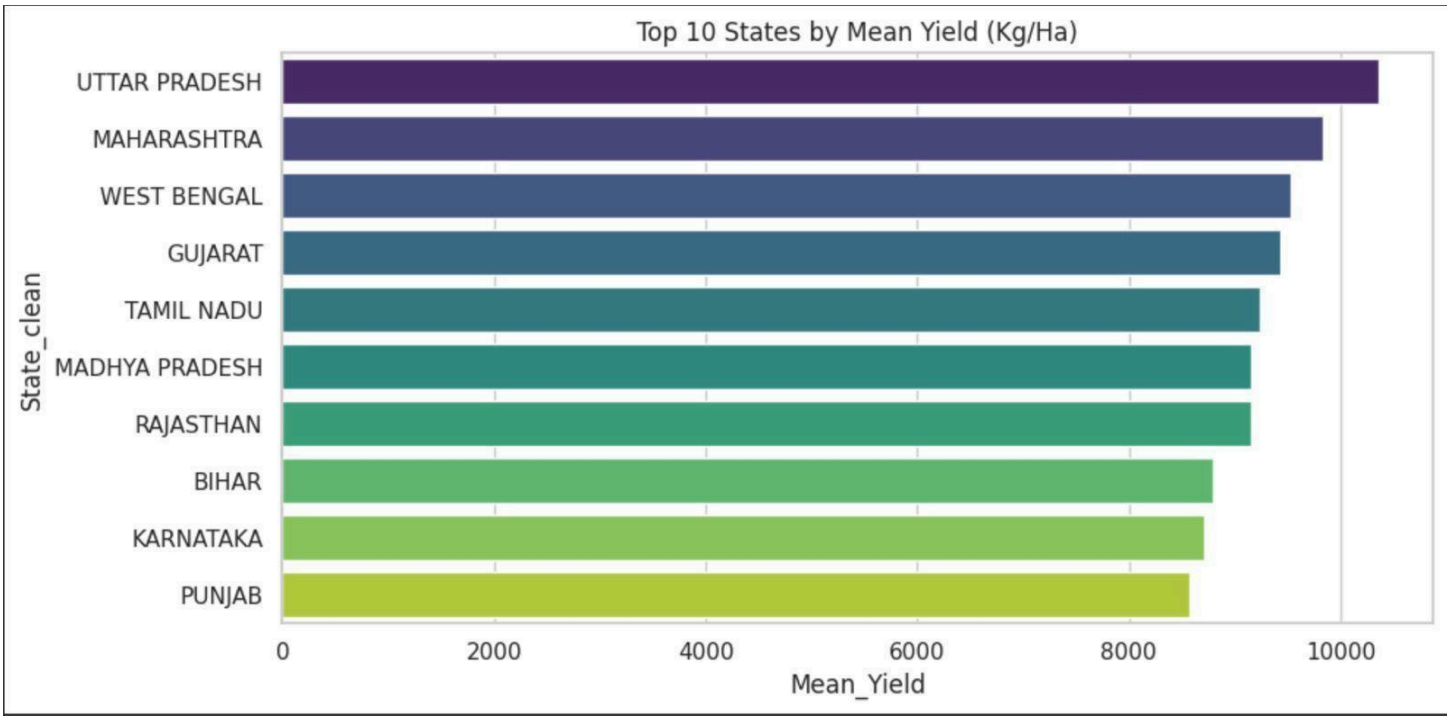

```
============================================================
  UNIT II — Aggregation & Transformations
============================================================

----------------------------------------
  Aggregated Sample
----------------------------------------
```

	State_clean	Year	Total_Area	Total_Prod	Avg_Yield	Yearly_Rain	Prod_per_Ha
0	BIHAR	2010	1902154.37	7699993.99	7277.418077	NaN	4048.038430
1	BIHAR	2011	1805033.04	6979693.28	8658.732857	NaN	3866.795303
2	BIHAR	2012	1611030.03	6187585.74	9434.554853	NaN	3840.763750
3	BIHAR	2013	1844663.66	7091402.29	6900.530676	NaN	3844.279282
4	BIHAR	2014	1962575.59	7532697.47	5782.251282	NaN	3838.169347

```
----------------------------------------
  Top States by Mean Yield
----------------------------------------
```

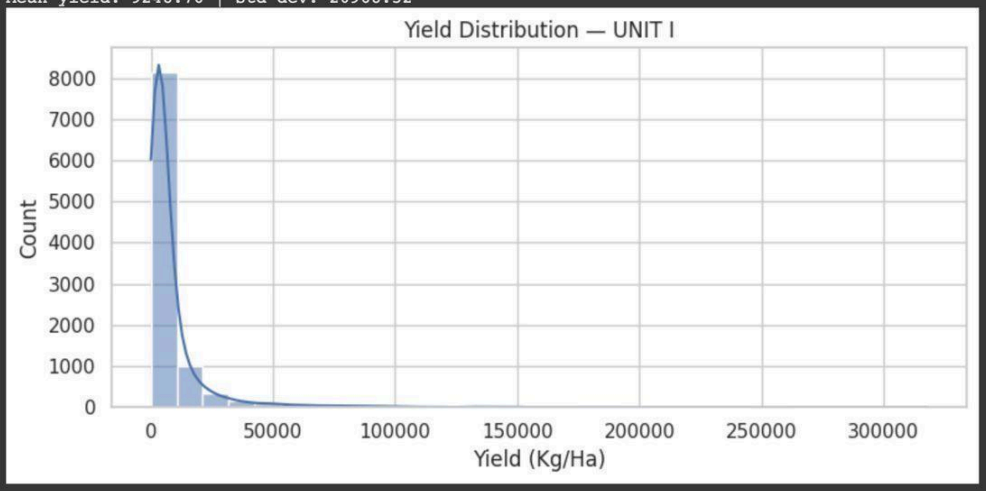
	State_clean	Mean_Yield	Mean_Rain
8	UTTAR PRADESH	10354.720044	NaN
4	MAHARASHTRA	9831.083299	NaN
9	WEST BENGAL	9526.123746	NaN
1	GUJARAT	9437.815753	NaN
7	TAMIL NADU	9232.448567	NaN

```
/tmp/ipython-input-1366073902.py:24: FutureWarning:
```




```
----------------------------------------
  2) Creating Arrays / Basic Operations
----------------------------------------

A + B = [3 6 9]
A * B = [ 2  8 18]

----------------------------------------
  3) Reshaping & Slicing Example
----------------------------------------

Matrix second row: [ 7  8  9 10 11 12]
Matrix third column: [ 3  9 15 21]
```

```
============================================================
  UNIT I — NumPy Fundamentals
============================================================


----------------------------------------
  1) Introduction to NumPy — Yield Array
----------------------------------------

Sample yields (first 12): [ 5385.1  28649.29 18779.42  7268.61  4463.96  2829.88  1305.11  3177.82
  5214.79  3739.83   951.66  3976.54]
Mean yield: 9248.76 | Std dev: 20968.52
```




```
Upload CropYieldDB.csv when prompted:
Choose Files CropYieldDB.csv
CropYieldDB.csv(text/csv) - 758274 bytes, last modified: n/a - 100% done
Saving CropYieldDB.csv to CropYieldDB.csv
Upload RegionWeatherDB.csv when prompted:
Choose Files RegionWeatherDB.csv
RegionWeatherDB.csv(text/csv) - 2496205 bytes, last modified: n/a - 100% done
Saving RegionWeatherDB.csv to RegionWeatherDB.csv
```

CHAPTER 6

CONCLUSION

The project “Crop Yield and Weather Data Analyser” successfully demonstrates how data analytics can be applied to understand and evaluate the relationship between weather patterns and agricultural productivity. By utilising the capabilities of NumPy and Pandas, two of Python’s most powerful libraries for data manipulation and numerical computation, this project transforms raw agricultural and meteorological data into valuable insights that can aid decision-making in the agricultural sector.

The main goal of this project was to analyse how different climatic factors — such as rainfall, temperature, and humidity — influence crop yield across various regions and time periods. Through systematic data collection, preprocessing, integration, and analysis, meaningful patterns and correlations were derived. The use of Pandas made it possible to efficiently clean, merge, and process large datasets, while NumPy enabled high-speed mathematical operations that enhanced analytical precision. This integration of libraries provided a robust foundation for conducting quantitative analysis on agricultural data.

One of the key outcomes of the project was the identification of correlations between crop yields and specific weather variables. The analysis revealed that crops highly dependent on rainfall tend to show significant yield fluctuations in years with irregular precipitation, while temperature-sensitive crops showed strong variation with seasonal temperature shifts. These observations not only validate the impact of climate on agriculture but also emphasise the importance of data-driven planning in farming practices.

The visualisation component of the project further strengthened the understanding of the results. Graphical representations such as line charts, bar graphs, and scatter plots provided an intuitive view of how yields changed over time or across regions under different weather conditions. These visual insights make the data accessible even to non-technical users such as policymakers, agricultural officers, and farmers. They can easily interpret the graphs to make informed decisions about crop selection, irrigation planning, and agricultural investment strategies.

Another major takeaway from this project is the practical understanding of how data preprocessing directly affects analytical outcomes. Handling missing data, correcting inconsistencies, and maintaining uniform formats were crucial steps that ensured the reliability of results. Without proper preprocessing, even the most

advanced analytical techniques could lead to inaccurate conclusions.

Beyond its analytical achievements, this project serves as a learning platform for understanding how computational tools can be used to solve real-world problems. It highlights the interdisciplinary connection between computer science, environmental studies, and agriculture — demonstrating how simple programming tools, when used effectively, can address significant socio-economic challenges.

While the project primarily focuses on correlation and trend analysis, it lays the groundwork for more advanced research in the future. Predictive modelling and machine learning techniques can be applied to forecast future crop yields based on expected weather conditions. Such advancements can help develop smart agricultural systems that provide early warnings for crop failures or weather-related risks.

In conclusion, the Crop Yield and Weather Data Analyser not only fulfils its objective of exploring the link between climate and crop productivity but also exemplifies how data analytics can contribute to sustainable and informed agricultural practices. The project stands as an example of how technology, when combined with scientific reasoning, can empower better decisions and lead to improved agricultural outcomes. It successfully bridges the gap between data science education and real-world application, reaffirming that knowledge of tools like NumPy and Pandas can have a tangible impact beyond the classroom — especially in domains as essential as food security and climate resilience.

FUTURE WORK

While the Crop Yield and Weather Data Analyser project provides valuable insights into the relationship between climatic factors and agricultural productivity, there is still considerable scope for future enhancement and development. The current version primarily focuses on data collection, preprocessing, and correlation-based analysis using NumPy and Pandas. However, the framework can be extended and improved in several ways to make the system more dynamic, predictive, and impactful for real-world agricultural applications.

5.1 Integration of Machine Learning Models

One of the most promising directions for future work is the implementation of machine learning algorithms to develop predictive models for crop yield forecasting. By training models such as Linear Regression, Random Forest, or Neural Networks on historical data, it would be possible to predict future yields based on upcoming or projected weather conditions. These predictive systems could help farmers plan ahead, choose suitable crops, and estimate potential profits or losses before the farming season begins.

Additionally, time-series analysis methods like ARIMA or LSTM networks can be used to analyze temporal data and forecast future weather or yield patterns. This would transition the project from being a static analytical tool to a real-time predictive system.

5.2 Expansion of Data Sources

In the current project, datasets are limited to regional weather and crop yield data for specific timeframes. In future iterations, data from soil quality, fertiliser usage, pesticide application, and irrigation patterns can be included. These factors play a major role in determining agricultural productivity and would significantly improve the accuracy of the analysis.

The system could also be connected to real-time weather APIs such as OpenWeatherMap or government meteorological databases, enabling live updates and continuous monitoring of crop conditions. This would allow the model to make up-to-date recommendations and provide early warnings for adverse climatic events like droughts or floods.

5.3 Development of a User Interface

Currently, the analysis and results are generated within a programming environment. In the future, a Graphical

User Interface (GUI) or web-based dashboard could be developed using frameworks like Flask, Django, or Streamlit. Such an interface would allow non-technical users, including farmers and agricultural officers, to easily upload data, visualise insights, and download reports. This would make the tool more accessible and practical for wider use in real-world agricultural systems.

5.4 Implementation of Geographic Information Systems (GIS)

Another valuable enhancement could be integrating Geographic Information Systems (GIS) to visualise yield and weather data spatially. Mapping yields and weather patterns region-wise on an interactive map would help identify high-yield and low-yield areas more intuitively. GIS integration could also support spatial analysis, providing deeper insights into how geography affects agricultural performance