CIND820 Capstone Project

Integrating Predictive Analytics and Anomaly Detection for Cyanobacterial Bloom Forecasting

This notebook details the development of predictive analytics for cyanobacterial blooms using ARIMA, LSTM, and anomaly detection models. It includes data preparation, exploratory analysis, and initial modeling insights.

```
# Mount Google Drive to permit colab Notebook access to project datasets

from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

1. Data Collection and Preprocessing

1.1 Import Necessary Libraries

Import libraries that will be required for data analysis, visualization, and handling.

```
# Import necessary libraries for data manipulation and visualization.

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

```

1.2 Source Identification and Data Acquisition

Define the file paths to the datasets that will be used in the analysis and in assessing the project research questions.

```
# Specify file paths for the datasets

bloom_indices_path = '/content/drive/MyDrive/CIND820/LakeErie_Daily_BloomIndices.csv'
water_quality_path = '/content/drive/MyDrive/CIND820/lake_erie_habs_field_sampling_results_2012_2018.csv'

```

1.3 Load Datasets

Read the Bloom Indices and Water Quality datasets into pandas DataFrames for manipulation.

```
# Load the Bloom Indices dataset
bloom_data = pd.read_csv(bloom_indices_path, encoding='ISO-8859-1')

# Load the Water Quality dataset
water_quality_data = pd.read_csv(water_quality_path, encoding='ISO-8859-1')

```

1.4 Data Inspection

Get a preliminary understanding of the data structure and content.

```
# Inspect the first few rows of the Bloom Indices dataset
print("Bloom Data Sample:")
print(bloom_data.head())

# Print a line of underscores to demarcate the outputs
print("_ " * 70)

# Inspect the first few rows of the Water Quality dataset
print("\nWater Quality Data Sample:")
print(water_quality_data.head())

```

```
3              41.8339              -83.3640              NaN
4              41.7625              -83.3286              NaN

   Wave Height (ft)  ... Soluble Reactive Phosphorus (µg P/L)  \
0               NaN  ...                                 3.23
1               NaN  ...                                 2.02
2               NaN  ...                                 2.97
3               NaN  ...                                 3.02
4               NaN  ...                                 5.15

  Ammonia (µg N/L)  Nitrate + Nitrite (mg N/L)  Urea (µg N/L)  \
0            10.22                        0.46            NaN
1            28.79                        0.32            NaN
2             35.7                        0.52            NaN
3            11.28                        0.51            NaN
4            14.68                        0.47            NaN

   Particulate Organic Carbon (mg/L) Particulate Organic Nitrogen (mg/L)  \
0                               NaN                                  NaN
1                              0.34                                 0.05
2                              0.40                                 0.08
3                              0.49                                 0.12
4                              0.50                                 0.07

  Dissolved Organic Carbon (mg/L)  \
0                            2.88
1                            1.80
2                            4.35
3                            2.53
4                            2.80

  Colored Dissolved Organic Material absorbance (m-1) at 400nm   \
0                                                NaN
1                                                NaN
2                                                NaN
3                                                NaN
4                                                NaN

   Total Suspended Solids (mg/L)  Volatile Suspended Solids (mg/L)
0                           3.47                              1.11
1                           3.15                              1.01
2                           2.90                              0.98
3                           4.32                              1.06
4                          21.42                              2.40

[5 rows x 36 columns]
```

1.5 Data Cleaning and Preparation

1.5.1 Parsing Dates

Convert date columns to datetime objects for time-series analysis.

2. Convert bloom_data Dates to Match water_quality_data Use the pd.to_datetime function to parse bloom_data dates into yyyy-mm-dd:

```
# Inspect the current date formats and data types
print("Initial Date Format and Data Type:")
print(bloom_data['Date'].head())
print(water_quality_data['Date'].head())

print("\nData Types:")
print(f"Bloom Data Date Type: {bloom_data['Date'].dtype}")
print(f"Water Quality Data Date Type: {water_quality_data['Date'].dtype}")

# Convert Date columns to datetime
bloom_data['Date'] = pd.to_datetime(bloom_data['Date'], format='%Y-%m-%d', errors='coerce')
water_quality_data['Date'] = pd.to_datetime(water_quality_data['Date'], format='%m/%d/%Y', errors='coerce')

# Check for invalid dates (after conversion)
print("\nPost Conversion - Check for Null Dates:")
print(bloom_data['Date'].isnull().sum())
print(water_quality_data['Date'].isnull().sum())

# Verify the new data types
print("\nPost Conversion - Data Types:")
print(f"Bloom Data Date Type: {bloom_data['Date'].dtype}")
print(f"Water Quality Data Date Type: {water_quality_data['Date'].dtype}")

# Preview the converted dates
print("\nConverted Date Formats:")
print(bloom_data['Date'].head())
print(water_quality_data['Date'].head())

```

```
Initial Date Format and Data Type:
0    2002-06-02
1    2002-06-15
2    2002-06-18
3    2002-06-23
4    2002-06-24
Name: Date, dtype: object
0    5/15/2012
1    5/15/2012
2    5/15/2012
3    5/15/2012
4    5/31/2012
Name: Date, dtype: object

Data Types:
Bloom Data Date Type: object
```

```
Water Quality Data Date Type: object

Post Conversion - Check for Null Dates:
0
0

Post Conversion - Data Types:
Bloom Data Date Type: datetime64[ns]
Water Quality Data Date Type: datetime64[ns]

Converted Date Formats:
0   2002-06-02
1   2002-06-15
2   2002-06-18
3   2002-06-23
4   2002-06-24
Name: Date, dtype: datetime64[ns]
0   2012-05-15
1   2012-05-15
2   2012-05-15
3   2012-05-15
4   2012-05-31
Name: Date, dtype: datetime64[ns]
```

1.5.2 Handling Column Names

Ensure consistency in column names for easier data manipulation. Standardize Column Names Normalize column names by stripping whitespace and other hidden characters.

2. Standardize Column Names Normalize column names by stripping whitespace and other hidden characters:

```
bloom_data.columns = bloom_data.columns.str.strip().str.replace('\n', '').str.replace('\r', '').str.replace('  ', ' ')
water_quality_data.columns = water_quality_data.columns.str.strip().str.replace('\n', '').str.replace('\r', '').str.replace('  ', ' ')

print("Normalized column names in bloom_data:", bloom_data.columns.tolist())
print("Normalized column names in water_quality_data:", water_quality_data.columns.tolist())
```

```
Normalized column names in bloom_data: ['Date', 'Lake', 'Satellite Sensor', 'Bloom Extent (KM2)', 'Bloom Extent (% of Lake Area)', 'Bloom Intensity (µg/L)',
Normalized column names in water_quality_data: ['Date', 'Site', 'Station Depth (m)', 'Sample Depth (m)', 'Sample Depth (category)', 'Local Time (Eastern Tim
```

1.5.3 Handling Missing Values

Address missing data to prevent errors in analysis.

```python
# Check for missing values in Bloom Data
print("\nMissing values in Bloom Data:")
print(bloom_data.isnull().sum())

```

```
Missing values in Bloom Data:
Date                              0
Lake                              0
Satellite Sensor                  0
Bloom Extent (KM2)                0
Bloom Extent (% of Lake Area)     0
Bloom Intensity (µg/L)            0
Bloom Severity (µg/L km2)         0
Valid Pixels (% of Lake Area)     0
dtype: int64
```

```python
# Check for missing values in Water Quality Data
print("\nMissing values in Water Quality Data:")
print(water_quality_data.isnull().sum())

```

```
Missing values in Water Quality Data:
Date                                                         0
Site                                                         0
Station Depth (m)                                          371
Sample Depth (m)                                            17
Sample Depth (category)                                      0
Local Time (Eastern Time Zone)                             363
Latitude (decimal deg)                                     376
Longitude (decimal deg)                                    376
Wind speed (knots)                                         650
Wave Height (ft)                                           655
Sky                                                        366
Secchi Depth (m)                                           405
Sample Temperature (°C)                                   1188
CTD Temperature (°C)                                       136
CTD Specific Conductivity (µS/cm)                          133
CTD Beam Attenuation (m-1)                                 151
CTD Tramission (%)                                         151
CTD Dissolved Oxygen (mg/L)                                132
CTD Photosynthetically Active Radiation (µE/m2/s)          134
Turbidity (NTU)                                            424
Particulate Microcystin (µg/L)                              19
Dissolved Microcystin (µg/L)                               251
Extracted Phycocyanin (µg/L)                                 4
Chlorophyll_a                                                5
Total Phosphorus (µg P/L)                                  371
Total Dissolved Phosphorus (µg P/L)                        369
Soluble Reactive Phosphorus (µg P/L)                       367
Ammonia (µg N/L)                                           366
```

```
Nitrate + Nitrite (mg N/L)                                   367
Urea (µg N/L)                                                925
Particulate Organic Carbon (mg/L)                            371
Particulate Organic Nitrogen (mg/L)                          371
Dissolved Organic Carbon (mg/L)                              524
Colored Dissolved Organic Material absorbance (m-1) at 400nm     568
Total Suspended Solids (mg/L)                                383
Volatile Suspended Solids (mg/L)                             381
dtype: int64
```

```
# Handle missing values in Water Quality Data
# For numeric columns, we can fill missing values with mean or median if appropriate
numeric_columns = water_quality_data.select_dtypes(include=[np.number]).columns.tolist()
water_quality_data[numeric_columns] = water_quality_data[numeric_columns].fillna(method='ffill')

```

1.6 Data Integration

1.6.1 Aligning Datasets Temporally

Combine datasets based on the date to facilitate joint analysis.

Final Check Rerun the merge operation after ensuring the Date columns are correctly formatted and aligned:

```
# Merge datasets on 'Date'
merged_data = pd.merge(bloom_data, water_quality_data, on='Date', how='inner')

```

```
merged_data = pd.merge(bloom_data, water_quality_data, on='Date', how='inner')
print(merged_data.head())

```

```
   Bloom Extent (% of Lake Area)  Bloom Intensity (µg/L)  \
0                           0.51                   49.26
1                           0.51                   49.26
2                           0.51                   49.26
3                           0.51                   49.26
4                           0.51                   49.26
```

```
4                    6703.63                              99.94  WE8

   Station Depth (m)  ...  Soluble Reactive Phosphorus (µg P/L)  \
0               5.50  ...                                   1.4
1                NaN  ...                                   NaN
2               8.71  ...                                   0.9
3               3.10  ...                                   2.7
4               4.67  ...                                   3.4

  Ammonia (µg N/L) Nitrate + Nitrite (mg N/L)  Urea (µg N/L)  \
0            17.9                       0.47           8.96
1             NaN                        NaN            NaN
2            56.1                       0.34          10.85
3             7.4                       1.73           8.37
4            38.9                       1.02          16.21

   Particulate Organic Carbon (mg/L) Particulate Organic Nitrogen (mg/L)  \
0                               0.40                                0.05
1                                NaN                                 NaN
2                               0.88                                0.16
3                               2.37                                0.37
4                               0.44                                0.07

   Dissolved Organic Carbon (mg/L)  \
0                             2.30
1                              NaN
2                             1.94
3                             4.53
4                             2.83

  Colored Dissolved Organic Material absorbance (m-1) at 400nm  \
0                                               0.37
1                                                NaN
2                                               0.22
3                                               1.68
4                                               0.68

  Total Suspended Solids (mg/L)  Volatile Suspended Solids (mg/L)
0                          2.02                              0.20
1                           NaN                               NaN
2                         12.27                              1.12
3                         24.53                              3.52
4                          2.42                              0.50

[5 rows x 43 columns]
```

```python
# Having already created the `merged_data`...

# Convert the merged data to a DataFrame
merged_df = pd.DataFrame(merged_data)

# Display the entire DataFrame in a scrollable table using Pandas
from IPython.display import display

```

```
# Display the DataFrame
display(merged_df)

```

	Date	Lake	Satellite Sensor	Bloom Extent (KM2)	Bloom Extent (% of Lake Area)	Bloom Intensity (µg/L)	Bloom Severity (µg/L km2)	Valid Pixels (% of Lake Area)	Site	Station Depth (m)	...	Soluble Reactive Phosphorus (µg P/L)	Ammonia (µg N/L)	Nitrate + Nitrite (mg N/L)	Urea (µg N/L)	Particulate Organic Carbon (mg/L)	Particulate Organic Nitrogen (mg/L)
0	2016-06-13	Lake Erie	OLCI-S3	136.08	0.51	49.26	6703.63	99.94	WE2	5.50	...	1.4	17.9	0.47	8.96	0.40	0.05
1	2016-06-13	Lake Erie	OLCI-S3	136.08	0.51	49.26	6703.63	99.94	WE2	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
2	2016-06-13	Lake Erie	OLCI-S3	136.08	0.51	49.26	6703.63	99.94	WE4	8.71	...	0.9	56.1	0.34	10.85	0.88	0.16
3	2016-06-13	Lake Erie	OLCI-S3	136.08	0.51	49.26	6703.63	99.94	WE6	3.10	...	2.7	7.4	1.73	8.37	2.37	0.37
4	2016-06-13	Lake Erie	OLCI-S3	136.08	0.51	49.26	6703.63	99.94	WE8	4.67	...	3.4	38.9	1.02	16.21	0.44	0.07
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
610	2018-10-01	Lake Erie	OLCI-S3	164.07	0.61	25.82	4236.66	99.91	WE13	9.07	...	6.17	1.72	0.27	NaN	0.50	0.10
611	2018-10-01	Lake Erie	OLCI-S3	164.07	0.61	25.82	4236.66	99.91	WE13	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
612	2018-10-01	Lake Erie	OLCI-S3	164.07	0.61	25.82	4236.66	99.91	WE16	6.30	...	18.33	22.69	0.11	NaN	0.72	0.12
613	2018-10-09	Lake Erie	OLCI-S3	157.86	0.59	24.85	3923.31	99.90	WE6	NaN	...	44.14	78.41	0.63	NaN	1.19	0.21
614	2018-10-09	Lake Erie	OLCI-S3	157.86	0.59	24.85	3923.31	99.90	WE9	NaN	...	59.19	145.34	0.98	NaN	1.45	0.27

615 rows × 43 columns

```
from google.colab import data_table
data_table.DataTable(merged_df)


```

```
Warning: Total number of columns (43) exceeds max_columns (20) limiting to first (20) columns.
```

1 to 25 of 615 entries Filter

index	Date	Lake	Satellite Sensor	Bloom Extent (KM2)	Bloom Extent (% of Lake Area)	Bloom Intensity (µg/L)	Bloom Severity (µg/L km2)	Valid Pixels (% of Lake Area)	Site	Stat
0	2016-06-13 00:00:00	Lake Erie	OLCI-S3	136.08	0.51	49.26	6703.63	99.94	WE2	
1	2016-06-13 00:00:00	Lake Erie	OLCI-S3	136.08	0.51	49.26	6703.63	99.94	WE2	
2	2016-06-13 00:00:00	Lake Erie	OLCI-S3	136.08	0.51	49.26	6703.63	99.94	WE4	
3	2016-06-13 00:00:00	Lake Erie	OLCI-S3	136.08	0.51	49.26	6703.63	99.94	WE6	
4	2016-06-13 00:00:00	Lake Erie	OLCI-S3	136.08	0.51	49.26	6703.63	99.94	WE8	
5	2016-06-13 00:00:00	Lake Erie	OLCI-S3	136.08	0.51	49.26	6703.63	99.94	WE8	
6	2016-06-13 00:00:00	Lake Erie	OLCI-S3	136.08	0.51	49.26	6703.63	99.94	WE9	
7	2016-06-13 00:00:00	Lake Erie	OLCI-S3	136.08	0.51	49.26	6703.63	99.94	WE12	
8	2016-06-13 00:00:00	Lake Erie	OLCI-S3	136.08	0.51	49.26	6703.63	99.94	WE12	
9	2016-06-13 00:00:00	Lake Erie	OLCI-S3	136.08	0.51	49.26	6703.63	99.94	WE13	
10	2016-06-13 00:00:00	Lake Erie	OLCI-S3	136.08	0.51	49.26	6703.63	99.94	WE13	
11	2016-06-13 00:00:00	Lake Erie	OLCI-S3	136.08	0.51	49.26	6703.63	99.94	WE15	
12	2016-06-27 00:00:00	Lake Erie	OLCI-S3	154.53	0.58	48.97	7567.06	99.93	WE2	
13	2016-06-27 00:00:00	Lake Erie	OLCI-S3	154.53	0.58	48.97	7567.06	99.93	WE2	
14	2016-06-27 00:00:00	Lake Erie	OLCI-S3	154.53	0.58	48.97	7567.06	99.93	WE4	
15	2016-06-27 00:00:00	Lake Erie	OLCI-S3	154.53	0.58	48.97	7567.06	99.93	WE6	

16	2016-06-27 00:00:00	Lake Erie	OLCI-S3	154.53	0.58	48.97	7567.06	99.93	WE8
17	2016-06-27 00:00:00	Lake Erie	OLCI-S3	154.53	0.58	48.97	7567.06	99.93	WE8
18	2016-06-27 00:00:00	Lake Erie	OLCI-S3	154.53	0.58	48.97	7567.06	99.93	WE9
19	2016-06-27 00:00:00	Lake Erie	OLCI-S3	154.53	0.58	48.97	7567.06	99.93	WE12
20	2016-06-27 00:00:00	Lake Erie	OLCI-S3	154.53	0.58	48.97	7567.06	99.93	WE12
21	2016-06-27 00:00:00	Lake Erie	OLCI-S3	154.53	0.58	48.97	7567.06	99.93	WE13
22	2016-06-27 00:00:00	Lake Erie	OLCI-S3	154.53	0.58	48.97	7567.06	99.93	WE13
23	2016-06-27 00:00:00	Lake Erie	OLCI-S3	154.53	0.58	48.97	7567.06	99.93	WE15
24	2016-07-05 00:00:00	Lake Erie	OLCI-S3	149.49	0.56	51.05	7631.85	99.94	WE2

Show 25 per page

1 2 10 20 25

```
# Define the file path for saving the merged dataframe
output_file_path = "/content/drive/MyDrive/CIND820/LakeErie_Merged_Data.csv"

# Save the merged dataframe to the specified location as a CSV file
merged_df.to_csv(output_file_path, index=False)

# Confirm the file has been saved
print(f"Merged DataFrame saved to: {output_file_path}")

```

```
Merged DataFrame saved to: /content/drive/MyDrive/CIND820/LakeErie_Merged_Data.csv
```

Summary of Section 1

- Handled missing values and ensured consistency across datasets.
- Ensured compatibility for downstream alignment and analysis steps.

2. Exploratory Data Analysis (EDA)

2.1 Statistical Summary and Visualization

2.1.1 Descriptive Statistics

Provide a statistical overview of the datasets.

```
# Summary statistics for Bloom Data
print("\nBloom Data Statistics:")
print(bloom_data.describe())

# Summary statistics for Water Quality Data
print("\nWater Quality Data Statistics:")
print(water_quality_data.describe())

```

```
Bloom Data Statistics:
                                Date  Bloom Extent (KM2)  \
count                           1369         1369.000000
mean   2010-10-09 18:57:03.813002240          443.157239
min              2002-06-02 00:00:00            0.270000
25%              2006-07-16 00:00:00          174.960000
50%              2009-08-26 00:00:00          256.320000
```

```
75%             2016-09-10 00:00:00         469.980000
max             2018-10-31 00:00:00        5256.810000
std                             NaN         594.930353

       Bloom Extent (% of Lake Area)  Bloom Intensity (µg/L)  \
count                    1369.000000             1369.000000
mean                        1.660767               33.203046
min                         0.000000               11.190000
25%                         0.660000               25.380000
50%                         0.960000               31.390000
75%                         1.760000               40.320000
max                        19.700000               65.620000
std                         2.229544               10.133595

       Bloom Severity (µg/L km2)  Valid Pixels (% of Lake Area)
count                1369.000000                    1369.000000
mean                13417.767480                      96.355566
min                     3.020000                       1.010000
25%                  7021.980000                      99.290000
50%                  8969.150000                      99.890000
75%                 11618.340000                      99.980000
max                176410.750000                     100.000000
std                 17402.623445                      12.535047

Water Quality Data Statistics:
                                Date  Station Depth (m)  Sample Depth (m)  \
count                           1244         873.000000       1227.000000
mean   2016-04-14 17:03:16.784566016           5.643860          1.930929
min              2012-05-15 00:00:00           1.900000          0.000000
25%              2015-06-22 00:00:00           4.000000          0.750000
50%              2016-07-18 00:00:00           5.330000          0.750000
75%              2017-08-21 00:00:00           7.800000          3.300000
max              2018-10-09 00:00:00          10.860000          8.700000
std                              NaN           2.202444          2.187762

       Latitude (decimal deg)  Longitude (decimal deg)  Wave Height (ft)  \
count              868.000000               868.000000        589.000000
mean                41.748882               -83.272040          0.879677
min                 41.012700               -83.708400          0.020000
25%                 41.705875               -83.363600          0.480000
50%                 41.743000               -83.328350          0.750000
75%                 41.826300               -83.193375          1.170000
max                 42.002000               -82.698300          2.970000
std                  0.080505                 0.118646          0.539996

       Sample Temperature (°C)  CTD Temperature (°C)  \
count                56.000000           1108.000000
mean                 23.823214             21.970036
min                  16.300000              2.400000
25%                  21.700000             20.300000
```

```
1 # Convert the bloom_data summary statistics output to a DataFrame
2 bloom_data_summary_statistics_df = pd.DataFrame(bloom_data.describe())
3
```

```
# Display the DataFrame using Pandas
from IPython.display import display

# Display the DataFrame
display(bloom_data_summary_statistics_df)

# Define the file path for saving the merged dataframe
output_file_path = "/content/drive/MyDrive/CIND820/bloom_data_summary_statistics_Output.csv"

# Save the merged dataframe to the specified location as a CSV file
bloom_data_summary_statistics_df.to_csv(output_file_path, index=False)
```

	Bloom Extent (KM2)	Bloom Extent (% of Lake Area)	Bloom Intensity (ug/L)	Bloom Severity (ug/L km2)	Valid Pixels (% of Lake Area)
count	1369.000000	1369.000000	1369.000000	1369.000000	1369.000000
mean	443.157239	1.660767	33.203046	13417.767480	96.355566
std	594.930353	2.229544	10.133595	17402.623445	12.535047
min	0.270000	0.000000	11.190000	3.020000	1.010000
25%	174.960000	0.660000	25.380000	7021.980000	99.290000
50%	256.320000	0.960000	31.390000	8969.150000	99.890000
75%	469.980000	1.760000	40.320000	11618.340000	99.980000
max	5256.810000	19.700000	65.620000	176410.750000	100.000000

Next steps: Generate code with `bloom_data_summary_statistics_df` View recommended plots New interactive sheet

```
# Convert the water_quality_data summary statistics output to a DataFrame
water_quality_data_summary_statistics_df = pd.DataFrame(water_quality_data.describe())

# Display the DataFrame using Pandas
from IPython.display import display

# Display the DataFrame
display(water_quality_data_summary_statistics_df)

# Define the file path for saving the merged dataframe
output_file_path = "/content/drive/MyDrive/CIND820/water_quality_data_summary_statistics_Output.csv"

# Save the merged dataframe to the specified location as a CSV file
water_quality_data_summary_statistics_df.to_csv(output_file_path, index=False)
```

	Date	Station Depth (m)	Sample Depth (m)	Latitude (decimal deg)	Longitude (decimal deg)	Wave Height (ft)	Sample Temperature (°C)	CTD Temperature (°C)	CTD Specific Conductivity (µS/cm)	CTD Photosynthetically Active Radiation (µE/m2/s)	Turbidity (NTU)
count	1244	873.000000	1227.000000	868.000000	868.000000	589.000000	56.000000	1108.000000	1111.000000	1110.000000	820.000000
mean	2016-04-14 17:03:16.784566016	5.643860	1.930929	41.748882	-83.272040	0.879677	23.823214	21.970036	289.774887	324.566315	19.161780
min	2012-05-15 00:00:00	1.900000	0.000000	41.012700	-83.708400	0.020000	16.300000	2.400000	1.500000	0.000000	0.680000
25%	2015-06-22 00:00:00	4.000000	0.750000	41.705875	-83.363600	0.480000	21.700000	20.300000	249.300000	14.000000	4.627500
50%	2016-07-18 00:00:00	5.330000	0.750000	41.743000	-83.328350	0.750000	24.200000	22.900000	274.000000	147.370000	9.710000
75%	2017-08-21 00:00:00	7.800000	3.300000	41.826300	-83.193375	1.170000	26.050000	24.500000	314.800000	460.372500	19.300000
max	2018-10-09 00:00:00	10.860000	8.700000	42.002000	-82.698300	2.970000	28.300000	29.700000	855.900000	1902.000000	1148.000000
std	NaN	2.202444	2.187762	0.080505	0.118646	0.539996	2.671261	3.635855	64.250139	418.007239	54.735476

Next steps: Generate code with `water_quality_data_summary_statistics_df` View recommended plots New interactive sheet

2. Exploratory Data Analysis (EDA)

Inspect Column Names to verify the exact column names. Standardize Column Names to remove unnecessary spaces and special characters like "µ" which may not render correctly and to simplify access for downstream analyses.

```
print(bloom_data.columns.tolist())

```

```
['Date', 'Lake', 'Satellite Sensor', 'Bloom Extent (KM2)', 'Bloom Extent (% of Lake Area)', 'Bloom Intensity (µg/L)', 'Bloom Severity (µg/L km2)', 'Valid Pi
```

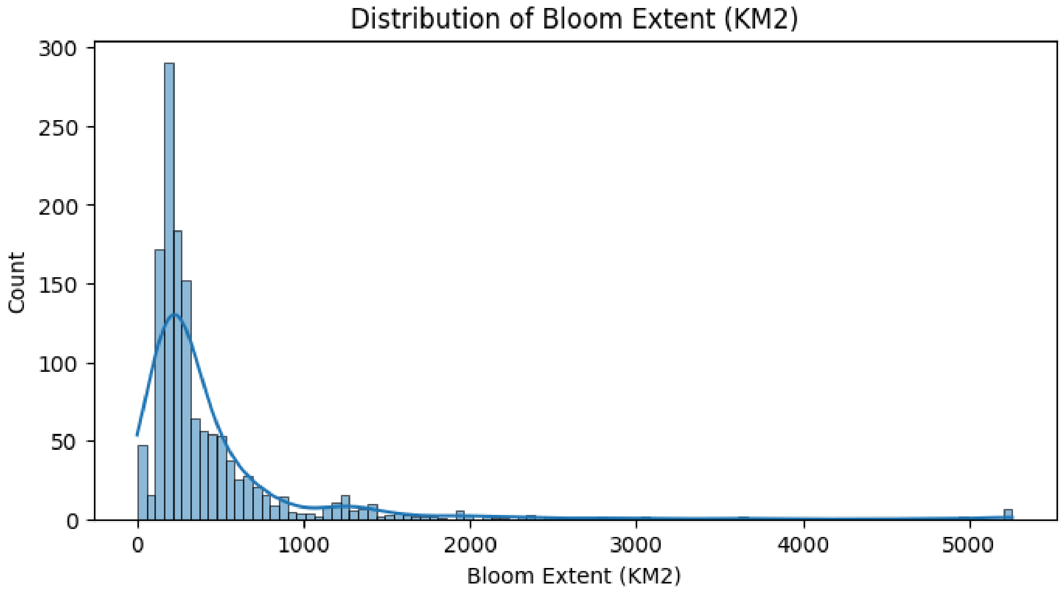
```
bloom_data.columns = bloom_data.columns.str.strip().str.replace('µ', 'u').str.replace('μ', 'u')
print(bloom_data.columns.tolist())

```

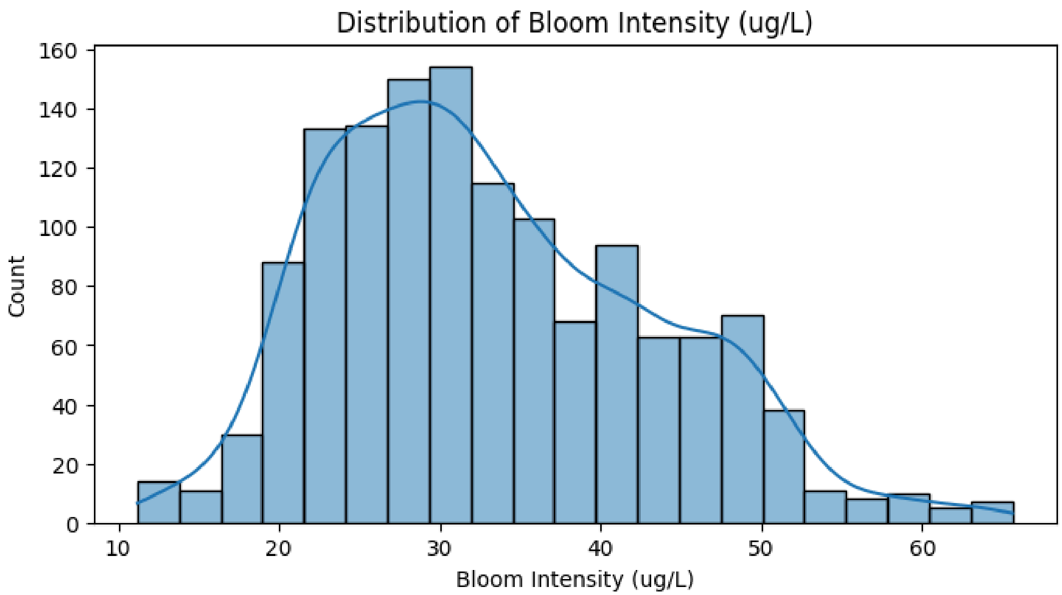
```
['Date', 'Lake', 'Satellite Sensor', 'Bloom Extent (KM2)', 'Bloom Extent (% of Lake Area)', 'Bloom Intensity (ug/L)', 'Bloom Severity (ug/L km2)', 'Valid Pi
```

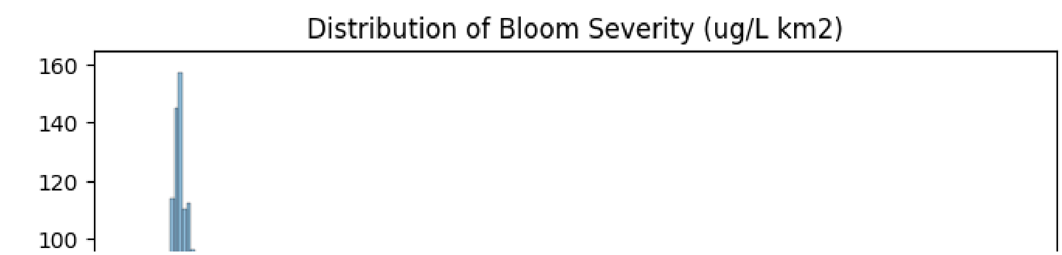
2.1.2 Visualization of Distributions

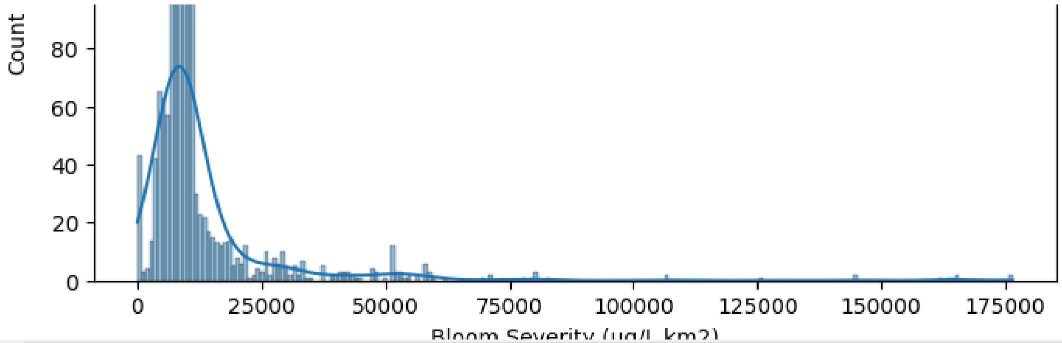
Visualize the distribution of key bloom indices to understand data variability.

```
# Plot histograms of key Bloom Indices
bloom_features = ['Bloom Extent (KM2)', 'Bloom Intensity (ug/L)', 'Bloom Severity (ug/L km2)']

for feature in bloom_features:
    plt.figure(figsize=(8, 4))
    sns.histplot(bloom_data[feature], kde=True)
    plt.title(f'Distribution of {feature}')
    plt.show()

```


Distribution of Bloom Extent (KM2)
Count
Bloom Extent (KM2)
0
50
100
150
200
250
300
0
1000
2000
3000
4000
5000


Distribution of Bloom Intensity (ug/L)
Count
Bloom Intensity (ug/L)
0
20
40
60
80
100
120
140
160
10
20
30
40
50
60

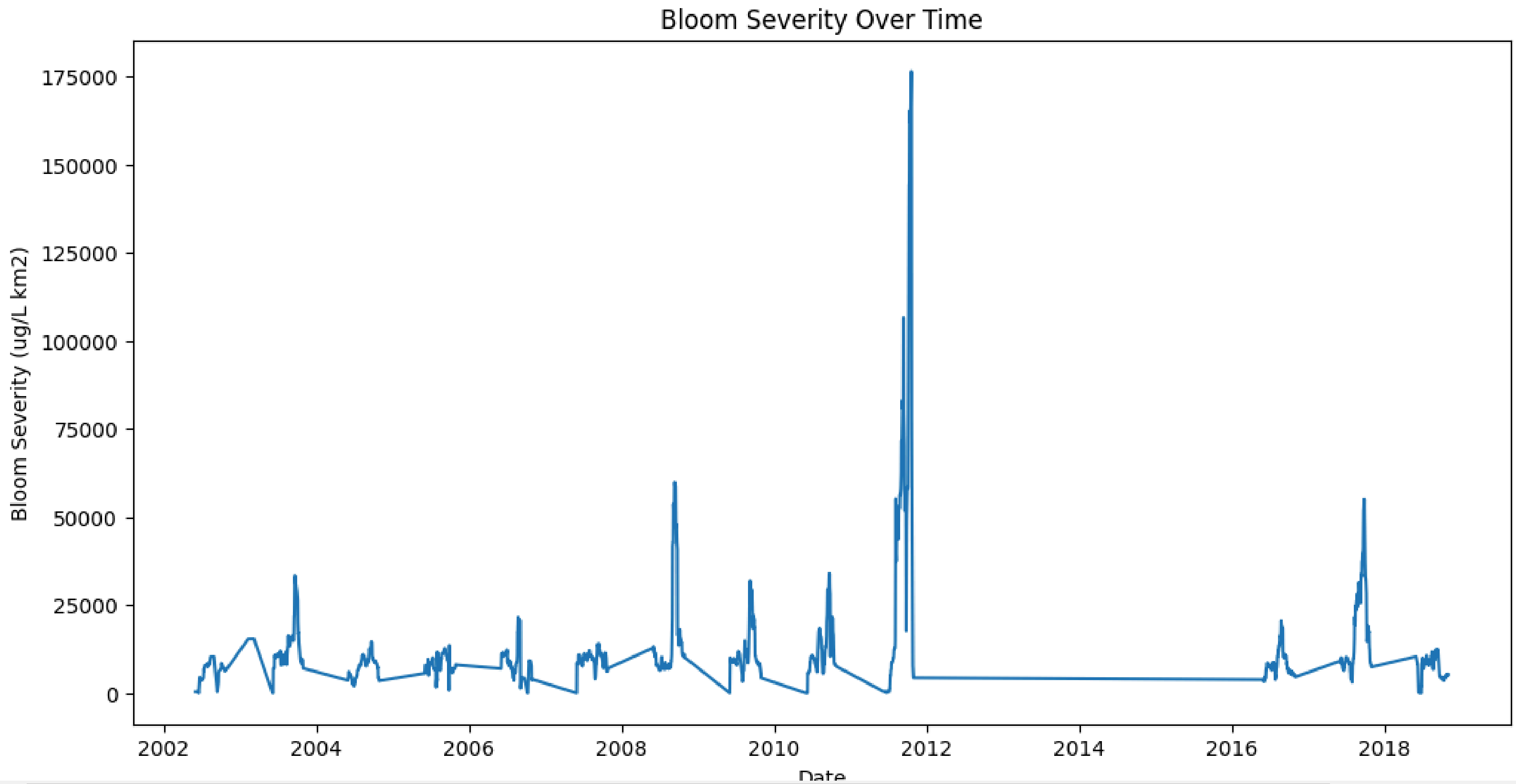

Distribution of Bloom Severity (ug/L km2)
100
120
140
160



2.1.3 Time-Series Plots

Identify trends and patterns over time in bloom severity.

```
# Time-series plot of Bloom Severity
plt.figure(figsize=(12, 6))
sns.lineplot(x='Date', y='Bloom Severity (ug/L km2)', data=bloom_data)
plt.title('Bloom Severity Over Time')
plt.xlabel('Date')
plt.ylabel('Bloom Severity (ug/L km2)')
plt.show()

```



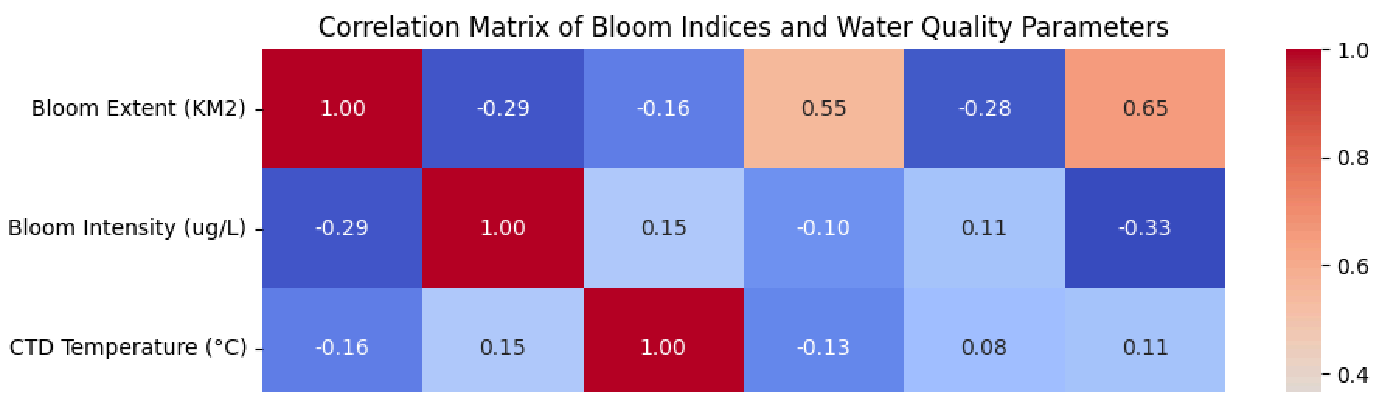
2.2 Correlation Analysis

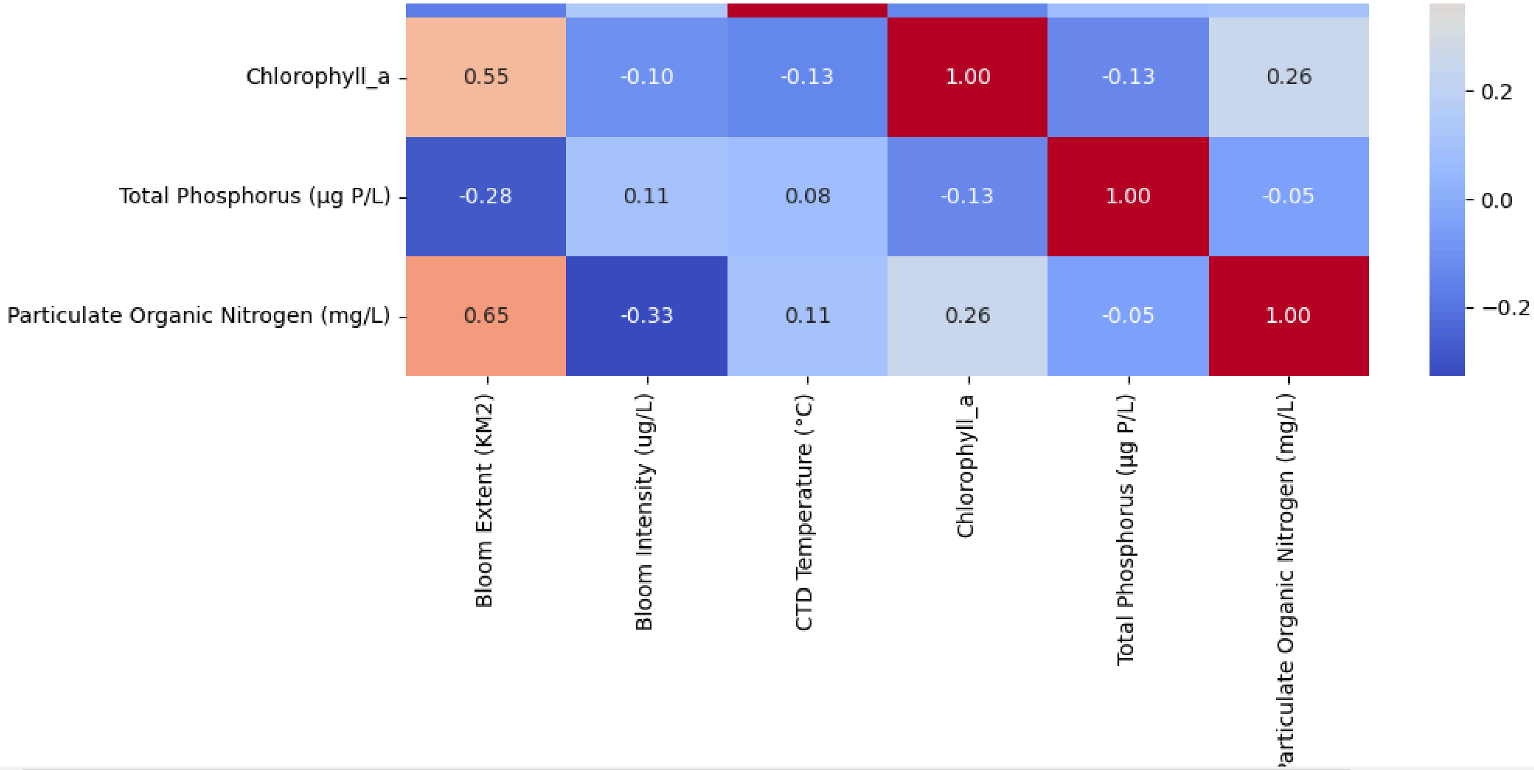
Identify relationships between environmental variables and bloom indices.

```
# Filter for numeric columns in the water quality dataset
numeric_columns = water_quality_data.select_dtypes(include=['float64', 'int64']).columns

# Aggregate water quality data by date (compute daily mean values for numeric columns)
water_quality_daily = water_quality_data.groupby('Date')[numeric_columns].mean().reset_index()

# Merge aggregated water quality data with bloom data on 'Date'
merged_daily_data = pd.merge(bloom_data, water_quality_daily, on='Date', how='inner')

# Compute correlation matrix for selected attributes
selected_columns = ['Bloom Extent (KM2)', 'Bloom Intensity (ug/L)', 'CTD Temperature (°C)', 'Chlorophyll_a', 'Total Phosphorus (µg P/L)', 'Particulate Organi
correlation_matrix = merged_daily_data[selected_columns].corr()

```

```
# Display the correlation matrix
print("Correlation Matrix:")
print(correlation_matrix)

# Plot the correlation heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix of Bloom Indices and Water Quality Parameters')
plt.show()
```

```
Correlation Matrix:
                                    Bloom Extent (KM2)  \
Bloom Extent (KM2)                            1.000000
Bloom Intensity (ug/L)                       -0.289382
CTD Temperature (°C)                         -0.163121
Chlorophyll_a                                 0.549689
Total Phosphorus (µg P/L)                    -0.282915
Particulate Organic Nitrogen (mg/L)           0.652454

                                    Bloom Intensity (ug/L)  \
Bloom Extent (KM2)                               -0.289382
Bloom Intensity (ug/L)                            1.000000
CTD Temperature (°C)                              0.147104
Chlorophyll_a                                    -0.101966
Total Phosphorus (µg P/L)                         0.105888
Particulate Organic Nitrogen (mg/L)              -0.328346

                                    CTD Temperature (°C)  Chlorophyll_a  \
Bloom Extent (KM2)                             -0.163121       0.549689
Bloom Intensity (ug/L)                          0.147104      -0.101966
CTD Temperature (°C)                            1.000000      -0.133796
Chlorophyll_a                                  -0.133796       1.000000
Total Phosphorus (µg P/L)                       0.083726      -0.128680
Particulate Organic Nitrogen (mg/L)             0.109280       0.256347

                                    Total Phosphorus (µg P/L)  \
Bloom Extent (KM2)                                  -0.282915
Bloom Intensity (ug/L)                               0.105888
CTD Temperature (°C)                                 0.083726
Chlorophyll_a                                       -0.128680
Total Phosphorus (µg P/L)                            1.000000
Particulate Organic Nitrogen (mg/L)                 -0.046774

                                    Particulate Organic Nitrogen (mg/L)
Bloom Extent (KM2)                                             0.652454
Bloom Intensity (ug/L)                                        -0.328346
CTD Temperature (°C)                                           0.109280
Chlorophyll_a                                                  0.256347
Total Phosphorus (µg P/L)                                     -0.046774
Particulate Organic Nitrogen (mg/L)                            1.000000
```




Chlorophyll_a
Total Phosphorus (µg P/L)
Particulate Organic Nitrogen (mg/L)
0.55 -0.10 -0.13 1.00 -0.13 0.26
-0.28 0.11 0.08 -0.13 1.00 -0.05
0.65 -0.33 0.11 0.26 -0.05 1.00
Bloom Extent (KM2)
Bloom Intensity (ug/L)
CTD Temperature (°C)
Chlorophyll_a
Total Phosphorus (µg P/L)
Particulate Organic Nitrogen (mg/L)
0.2
0.0
−0.2

```python
print(bloom_data.columns)
print(water_quality_daily.columns)

```

```
Index(['level_0', 'index', 'Date', 'Lake', 'Satellite Sensor',
       'Bloom Extent (KM2)', 'Bloom Extent (% of Lake Area)',
       'Bloom Intensity (ug/L)', 'Bloom Severity (ug/L km2)',
       'Valid Pixels (% of Lake Area)'],
      dtype='object')
Index(['level_0', 'index', 'Date', 'Station Depth (m)', 'Sample Depth (m)',
       'Latitude (decimal deg)', 'Longitude (decimal deg)', 'Wave Height (ft)',
       'Sample Temperature (°C)', 'CTD Temperature (°C)',
       'CTD Specific Conductivity (µS/cm)',
       'CTD Photosynthetically Active Radiation (µE/m2/s)', 'Turbidity (NTU)',
       'Chlorophyll_a', 'Total Phosphorus (µg P/L)',
       'Total Dissolved Phosphorus (µg P/L)',
       'Particulate Organic Carbon (mg/L)',
       'Particulate Organic Nitrogen (mg/L)',
       'Dissolved Organic Carbon (mg/L)',
       'Colored Dissolved Organic Material absorbance (m-1) at 400nm',
       'Total Suspended Solids (mg/L)', 'Volatile Suspended Solids (mg/L)'],
      dtype='object')
```

```python
# Drop unnecessary columns
bloom_data = bloom_data.drop(columns=['level_0', 'index'], errors='ignore')
water_quality_daily = water_quality_daily.drop(columns=['level_0', 'index'], errors='ignore')

# Ensure 'Date' is a column, not the index
bloom_data.reset_index(drop=True, inplace=True)
water_quality_daily.reset_index(drop=True, inplace=True)

# Merge the two datasets on 'Date' to ensure alignment
aligned_data = pd.merge(
    bloom_data[['Date', 'Bloom Extent (KM2)', 'Bloom Severity (ug/L km2)']],
    water_quality_daily[['Date', 'Chlorophyll_a', 'CTD Temperature (°C)', 'Total Phosphorus (µg P/L)', 'Particulate Organic Nitrogen (mg/L)']],
    on='Date', how='inner'
)

# Drop rows with missing data
aligned_data = aligned_data.dropna()

# Display the first few rows of the aligned dataset
print(aligned_data.head())

```

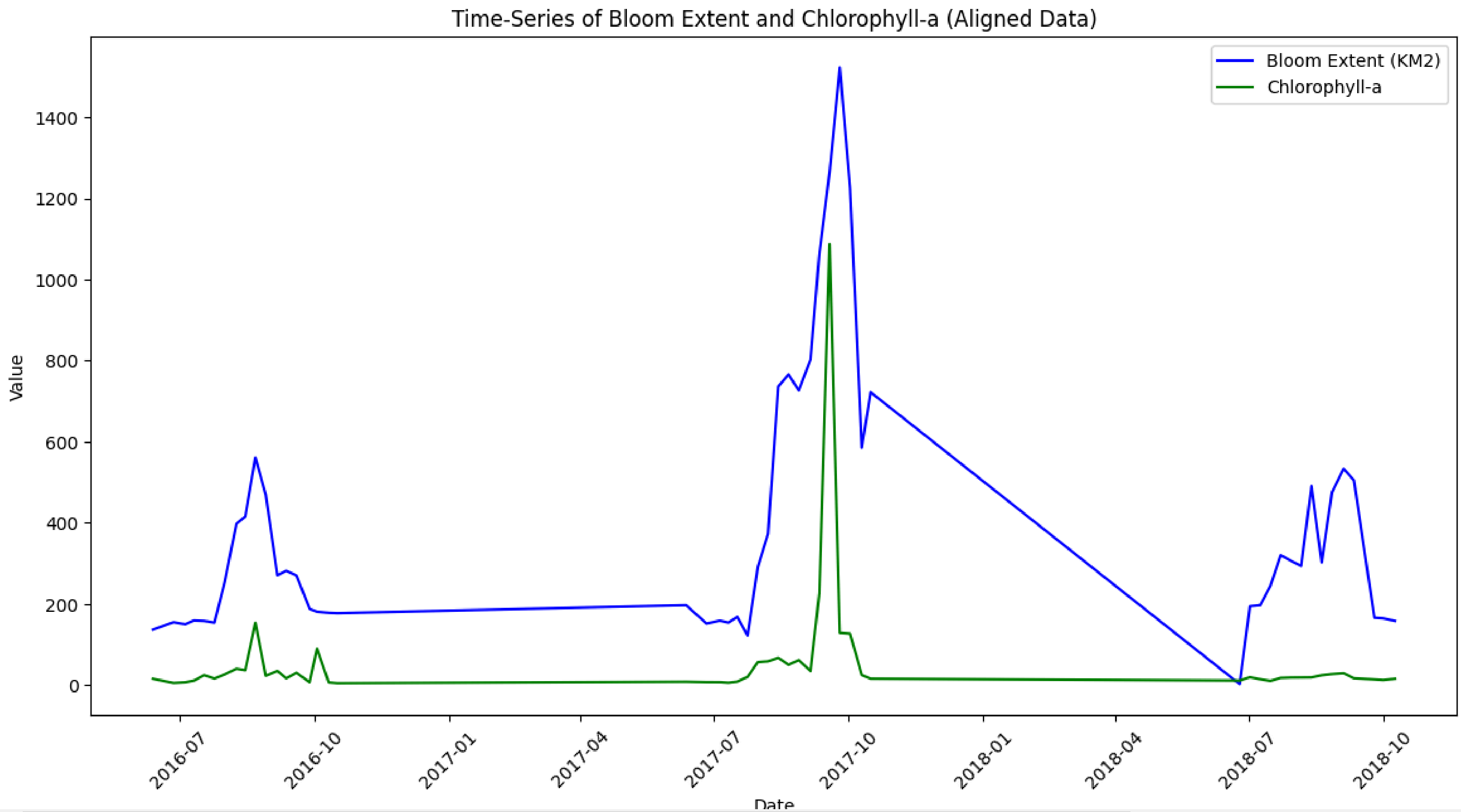
```
         Date  Bloom Extent (KM2)  Bloom Severity (ug/L km2)  Chlorophyll_a  \
0 2016-06-13              136.08                    6703.63      14.980000
1 2016-06-27              154.53                    7567.06       4.651667
2 2016-07-05              149.49                    7631.85       6.260833
3 2016-07-11              158.85                    7946.17      10.446667
```

```
4 2016-07-18              157.77                   7864.66      23.877500

   CTD Temperature (°C)  Total Phosphorus (µg P/L)  \
0             21.575000                    87.5625
1             24.191667                    77.9875
2             23.450000                    75.5875
3             24.933333                    60.5875
4             24.766667                    61.7000

   Particulate Organic Nitrogen (mg/L)
0                              0.18625
1                              0.14250
2                              0.11625
3                              0.16375
4                              0.18125
```

```
print(aligned_data.columns)
print(aligned_data.isnull().sum())

```

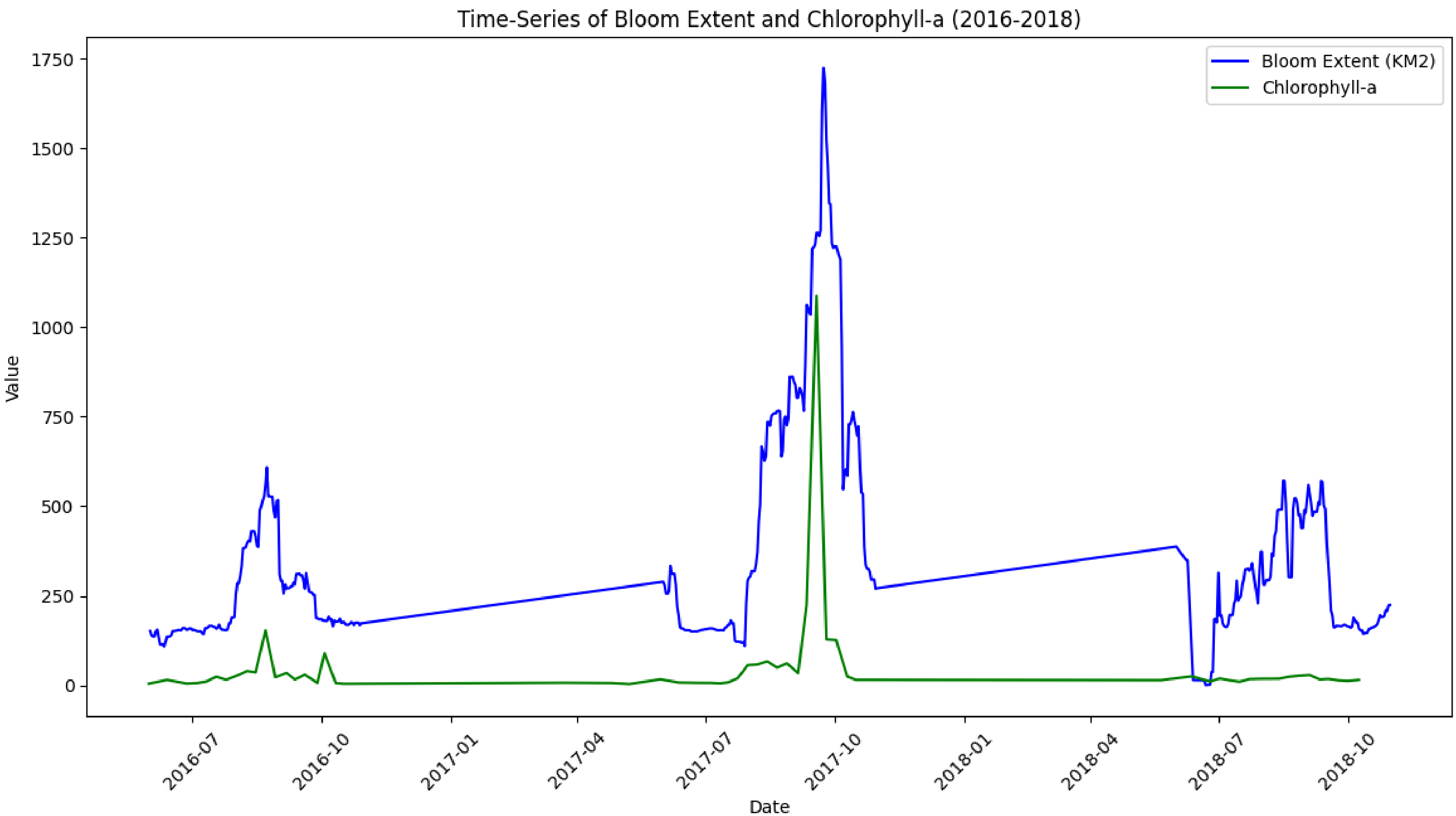
```
Index(['Date', 'Bloom Extent (KM2)', 'Bloom Severity (ug/L km2)',
       'Chlorophyll_a', 'CTD Temperature (°C)', 'Total Phosphorus (µg P/L)',
       'Particulate Organic Nitrogen (mg/L)'],
      dtype='object')
Date                                   0
Bloom Extent (KM2)                     0
Bloom Severity (ug/L km2)              0
Chlorophyll_a                          0
CTD Temperature (°C)                   0
Total Phosphorus (µg P/L)              0
Particulate Organic Nitrogen (mg/L)    0
dtype: int64
```

```
# Plot the aligned time-series data
plt.figure(figsize=(14, 7))
sns.lineplot(x='Date', y='Bloom Extent (KM2)', data=aligned_data, label='Bloom Extent (KM2)', color='blue')
sns.lineplot(x='Date', y='Chlorophyll_a', data=aligned_data, label='Chlorophyll-a', color='green')
plt.title('Time-Series of Bloom Extent and Chlorophyll-a (Aligned Data)')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.xticks(rotation=45)
plt.show()
```




```python
# Filter data for the range 2016 to 2018 inclusive
bloom_data_filtered = bloom_data[(bloom_data['Date'] >= '2016-01-01') & (bloom_data['Date'] <= '2018-12-31')]
water_quality_filtered = water_quality_daily[(water_quality_daily['Date'] >= '2016-01-01') & (water_quality_daily['Date'] <= '2018-12-31')]

# Merge the filtered datasets (outer join to include all dates in the range)
merged_filtered_data = pd.merge(bloom_data_filtered[['Date', 'Bloom Extent (KM2)']],
                                water_quality_filtered[['Date', 'Chlorophyll_a']],
                                on='Date', how='outer')

# Sort by date for consistent plotting
merged_filtered_data = merged_filtered_data.sort_values(by='Date')

# Plot the full range data
```

```
plt.figure(figsize=(14, 7))
sns.lineplot(x='Date', y='Bloom Extent (KM2)', data=merged_filtered_data, label='Bloom Extent (KM2)', color='blue')
sns.lineplot(x='Date', y='Chlorophyll_a', data=merged_filtered_data, label='Chlorophyll-a', color='green')
plt.title('Time-Series of Bloom Extent and Chlorophyll-a (2016-2018)')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.xticks(rotation=45)
plt.show()

```



2.3 Anomaly Detection

Detect unusual bloom severity events for further investigation.

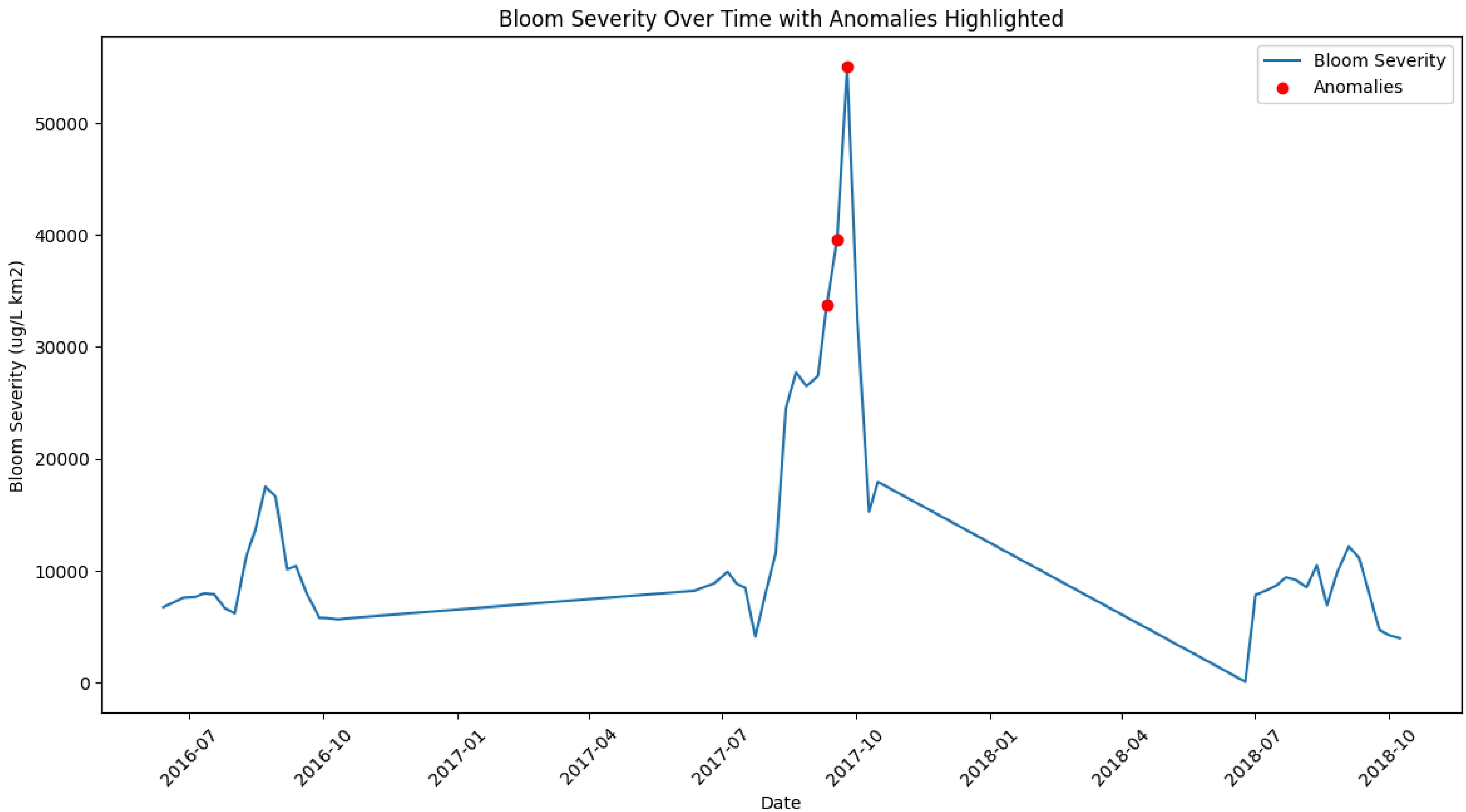
```python
# Compute Z-scores for Bloom Severity
aligned_data['Bloom_Severity_Z'] = (
    (aligned_data['Bloom Severity (ug/L km2)'] - aligned_data['Bloom Severity (ug/L km2)'].mean()) /
    aligned_data['Bloom Severity (ug/L km2)'].std()
)

# Identify anomalies where Z-score > 2 or < -2
anomalies = aligned_data[(aligned_data['Bloom_Severity_Z'] > 2) | (aligned_data['Bloom_Severity_Z'] < -2)]

# Display anomalies
print("\nAnomalies in Bloom Severity:")
print(anomalies[['Date', 'Bloom Severity (ug/L km2)', 'Bloom_Severity_Z']])

```

```
Anomalies in Bloom Severity:
         Date  Bloom Severity (ug/L km2)  Bloom_Severity_Z
30 2017-09-11                   33721.23          2.042731
31 2017-09-18                   39543.10          2.606389
32 2017-09-25                   55014.21          4.104259
```

Visualize the Anomalies:

Plot Bloom Severity over time, with the anomaly dates highlighted to provide a clear visual representation of their extremity.

```python
# Plot Bloom Severity with Anomalies Highlighted
plt.figure(figsize=(14, 7))
sns.lineplot(x='Date', y='Bloom Severity (ug/L km2)', data=aligned_data, label='Bloom Severity')
plt.scatter(anomalies['Date'], anomalies['Bloom Severity (ug/L km2)'], color='red', label='Anomalies', zorder=5)
plt.title('Bloom Severity Over Time with Anomalies Highlighted')
plt.xlabel('Date')
plt.ylabel('Bloom Severity (ug/L km2)')
plt.legend()
plt.xticks(rotation=45)
plt.show()

```



Summary of Section 2

- Visualized distributions and correlations.
- Key Insight: Chlorophyll-a strongly correlates with bloom severity.

3. Model Development and Training

3.1 Time-Series Forecasting Model

3.1.1 Preparing Data for Time-Series Modeling

Prepare the data in the correct format for time-series forecasting.

```
# Ensure 'Date' is present and converted to datetime
if 'Date' not in bloom_data.columns:
    bloom_data.reset_index(inplace=True)  # Reset index if 'Date' is currently the index

bloom_data['Date'] = pd.to_datetime(bloom_data['Date'], errors='coerce')  # Convert to datetime

# Drop any rows where 'Date' could not be parsed
bloom_data.dropna(subset=['Date'], inplace=True)

# Set 'Date' as the index
bloom_data.set_index('Date', inplace=True)

# Resample Bloom Severity data to monthly frequency and fill missing values
monthly_bloom = bloom_data['Bloom Severity (ug/L km2)'].resample('ME').mean().ffill()  # Updated syntax

# Display the first few rows of the resampled data
print(monthly_bloom.head())

```

```
Date
2002-06-30    2295.754286
2002-07-31    6197.530000
2002-08-31    9033.583333
2002-09-30    3141.056667
2002-10-31    7056.688824
Freq: ME, Name: Bloom Severity (ug/L km2), dtype: float64
```

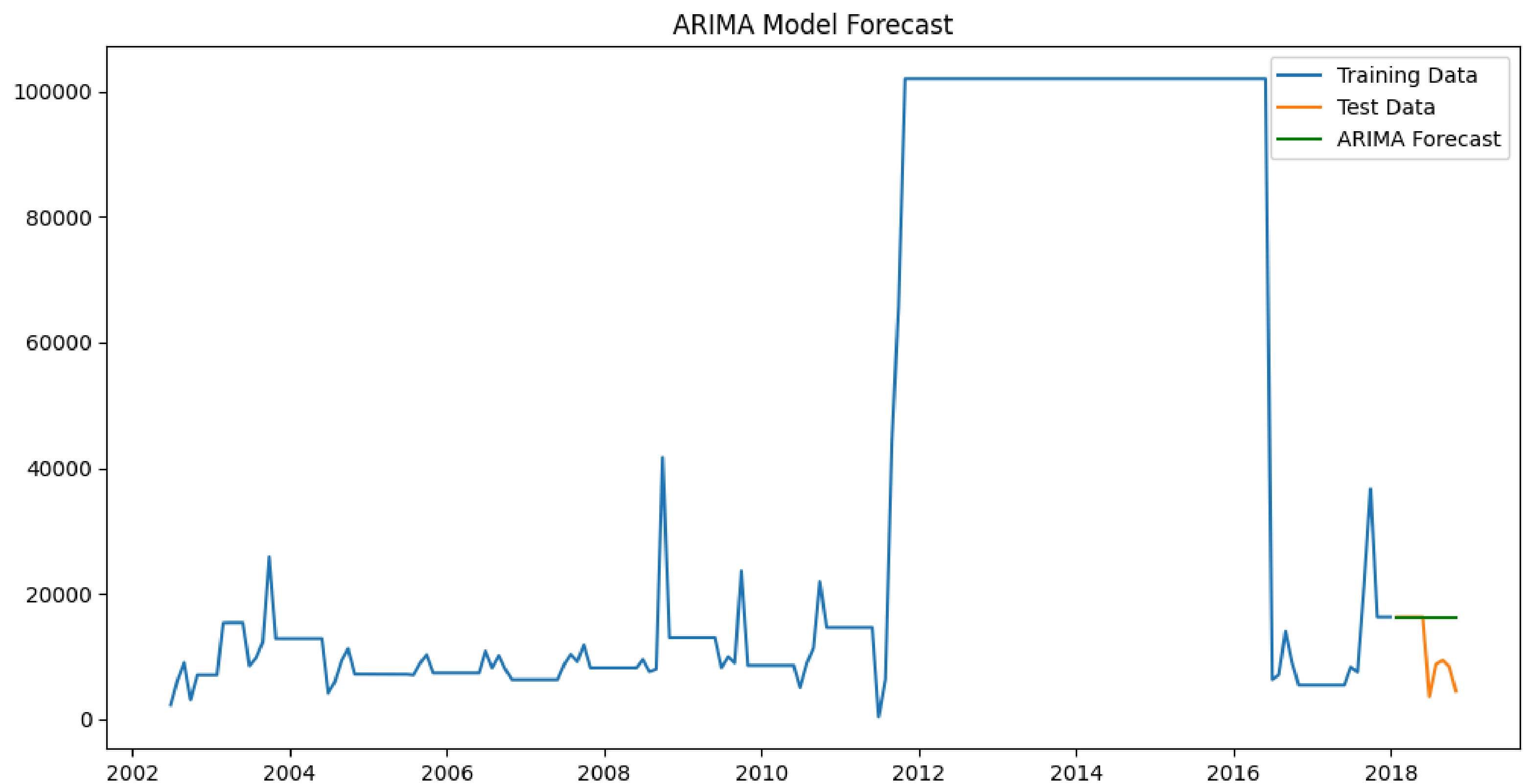
3.1.2 ARIMA Model

Develop a statistical time-series model to forecast bloom severity.

```
# Import ARIMA model
from statsmodels.tsa.arima.model import ARIMA

# Split data into training and testing sets
train_data = monthly_bloom[:'2017']
test_data = monthly_bloom['2018':]

# Fit ARIMA model
model_arima = ARIMA(train_data, order=(1, 1, 1))
arima_result = model_arima.fit()
```

```
# Forecast
forecast_arima = arima_result.predict(start=test_data.index[0], end=test_data.index[-1], typ='levels')

# Plot forecasts
plt.figure(figsize=(12, 6))
plt.plot(train_data, label='Training Data')
plt.plot(test_data, label='Test Data')
plt.plot(forecast_arima, label='ARIMA Forecast', color='green')
plt.legend()
plt.title('ARIMA Model Forecast')
plt.show()


```

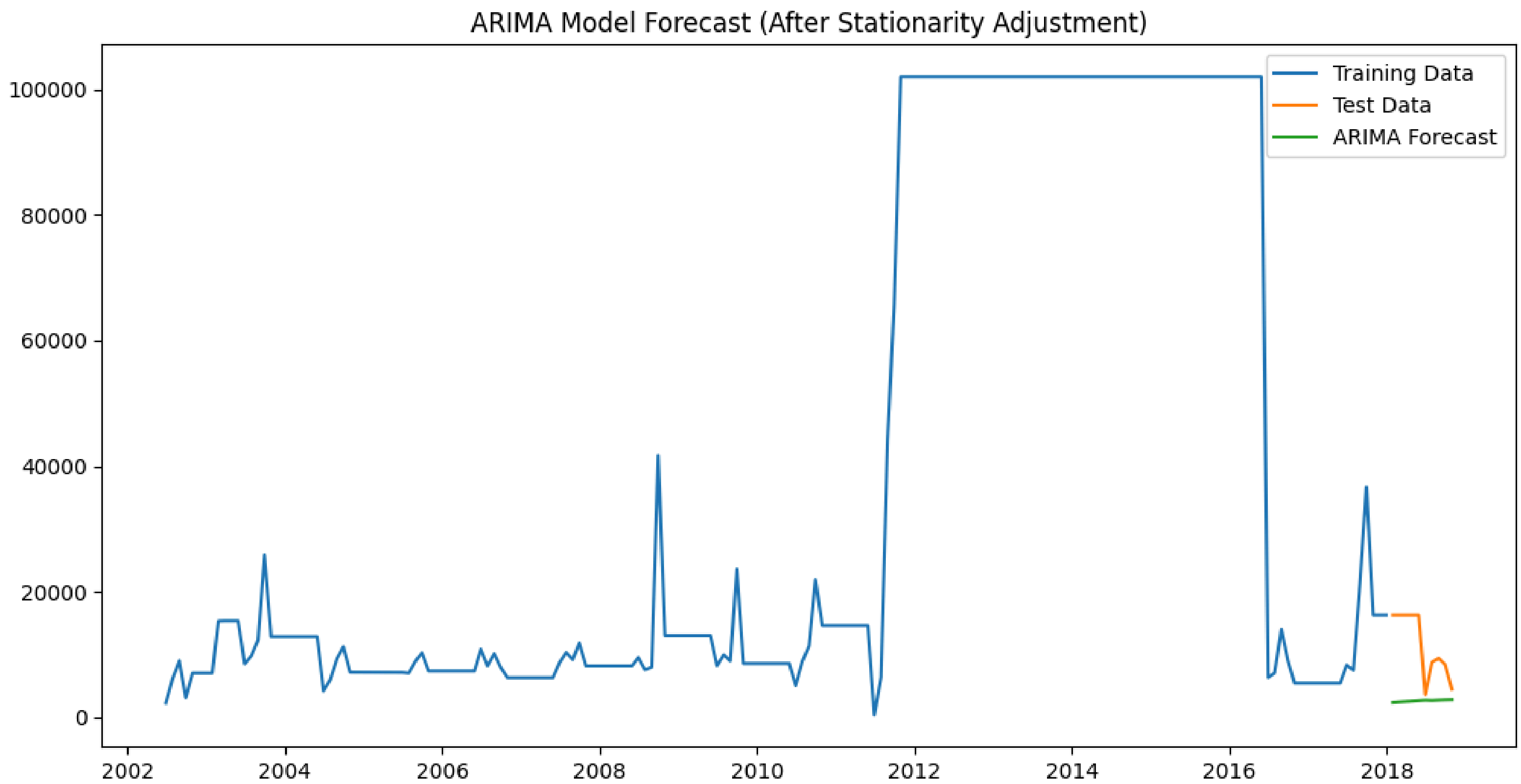
```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/statespace/sarimax.py:966: UserWarning: Non-stationary starting autoregressive parameters found. Usi
  warn('Non-stationary starting autoregressive parameters'
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/statespace/sarimax.py:978: UserWarning: Non-invertible starting MA parameters found. Using zeros as
  warn('Non-invertible starting MA parameters found.'
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/statespace/representation.py:374: FutureWarning: Unknown keyword arguments: dict_keys(['typ']).Passi
  warnings.warn(msg, FutureWarning)
```



```
# Apply differencing to make the data stationary
monthly_bloom_diff = monthly_bloom.diff().dropna()

# Reassess stationarity with Augmented Dickey-Fuller test
from statsmodels.tsa.stattools import adfuller

adf_test = adfuller(monthly_bloom_diff)
print("ADF Statistic:", adf_test[0])
print("p-value:", adf_test[1])
if adf_test[1] < 0.05:
    print("The data is stationary.")
else:
    print("The data is not stationary. Further transformations may be required.")

```

```
ADF Statistic: -13.877731330116458
p-value: 6.293234184492074e-26
The data is stationary.
```

```
# Refit ARIMA model after data transformation
model_arima = ARIMA(monthly_bloom_diff, order=(1, 1, 1))  # Adjust parameters based on ACF/PACF
arima_result = model_arima.fit()

# Forecast
forecast_arima_diff = arima_result.predict(start=test_data.index[0], end=test_data.index[-1])

# Revert differencing to interpret predictions
forecast_arima = forecast_arima_diff.cumsum() + monthly_bloom.iloc[0]  # Add back the first value for interpretation

# Plot updated forecasts
plt.figure(figsize=(12, 6))
plt.plot(train_data, label='Training Data')
plt.plot(test_data, label='Test Data')
plt.plot(forecast_arima, label='ARIMA Forecast')
plt.legend()
plt.title('ARIMA Model Forecast (After Stationarity Adjustment)')
plt.show()

```



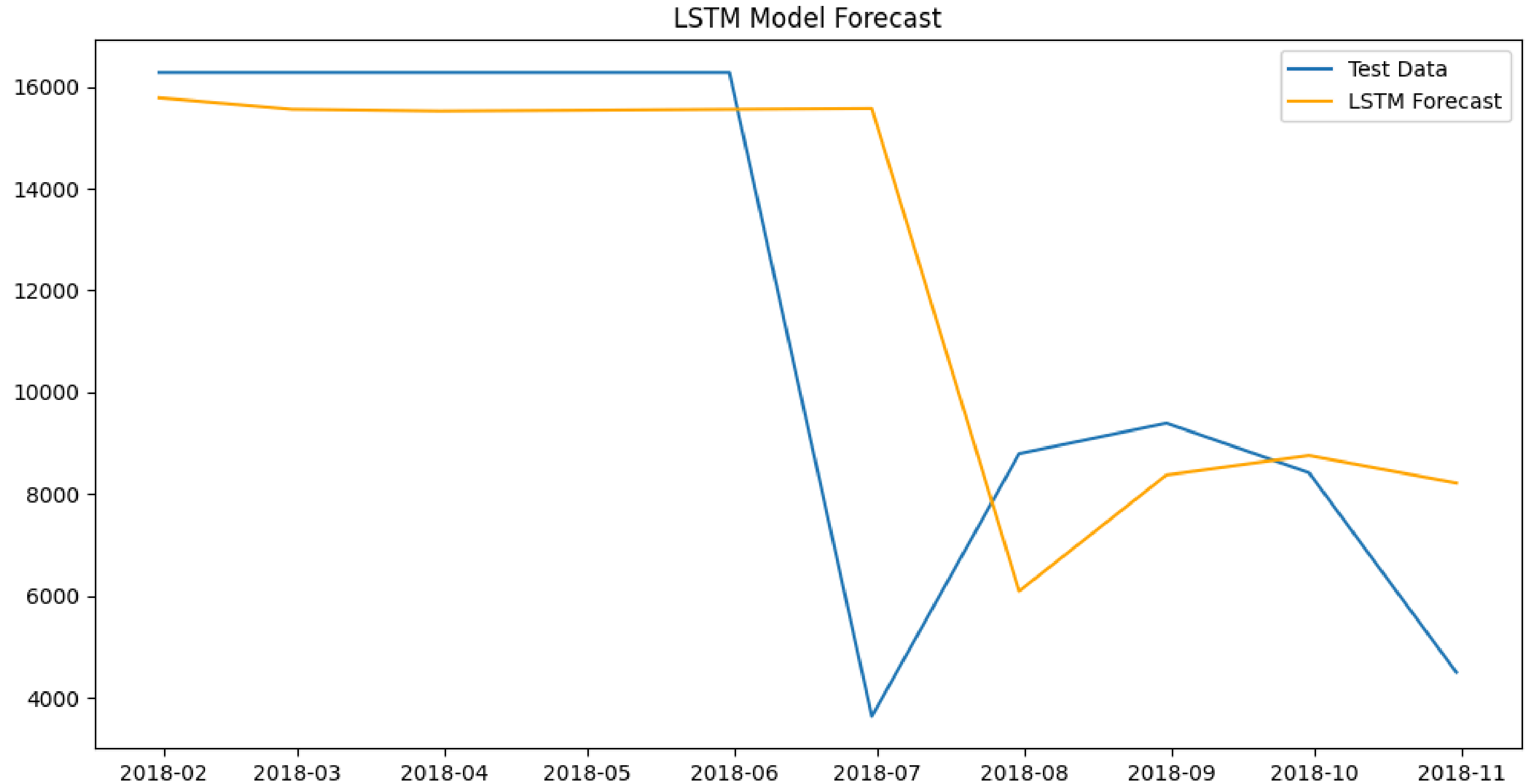
3.1.3 LSTM Model

Develop a neural network model to capture complex patterns in bloom severity.

```
# Import libraries for LSTM
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Prepare data for LSTM
scaler = MinMaxScaler()
monthly_bloom_scaled = scaler.fit_transform(monthly_bloom.values.reshape(-1, 1))

# Create sequences
def create_sequences(data, seq_length):
    X = []
    y = []
```

```
    for i in range(len(data) - seq_length):
        X.append(data[i:i + seq_length])
        y.append(data[i + seq_length])
    return np.array(X), np.array(y)

seq_length = 12
X, y = create_sequences(monthly_bloom_scaled, seq_length)

# Split into training and testing sets
train_size = len(train_data)
X_train, X_test = X[:train_size - seq_length], X[train_size - seq_length:]
y_train, y_test = y[:train_size - seq_length], y[train_size - seq_length:]

# Build LSTM model
model_lstm = Sequential()
model_lstm.add(LSTM(50, activation='relu', input_shape=(seq_length, 1)))
model_lstm.add(Dense(1))
model_lstm.compile(optimizer='adam', loss='mse')

# Train the model
model_lstm.fit(X_train, y_train, epochs=50, batch_size=1, verbose=0)

# Forecast
lstm_predictions = model_lstm.predict(X_test)
lstm_predictions = scaler.inverse_transform(lstm_predictions)

# Plot forecasts
plt.figure(figsize=(12, 6))
plt.plot(test_data.index, test_data.values, label='Test Data')
plt.plot(test_data.index, lstm_predictions, label='LSTM Forecast', color='orange')
plt.legend()
plt.title('LSTM Model Forecast')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When usi
  super().__init__(**kwargs)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 340ms/step
```




```
from sklearn.metrics import mean_squared_error, mean_absolute_error

# Evaluate LSTM Model
mse_lstm = mean_squared_error(y_test, lstm_predictions)
mae_lstm = mean_absolute_error(y_test, lstm_predictions)
rmse_lstm = mse_lstm ** 0.5

print(f"LSTM Model MSE: {mse_lstm}")
print(f"LSTM Model MAE: {mae_lstm}")
print(f"LSTM Model RMSE: {rmse_lstm}")

```

```
LSTM Model MSE: 170808569.37927246
LSTM Model MAE: 12492.524370115412
LSTM Model RMSE: 13069.375248238626
```

3.2 Anomaly Detection Model

3.2.1 Isolation Forest

Detect anomalies in environmental parameters that may precede bloom events.

```
# Prepare data (exclude Total Phosphorus since it's unavailable)
features = aligned_data[['CTD Temperature (°C)', 'Chlorophyll_a']].ffill()

# Fit Isolation Forest
from sklearn.ensemble import IsolationForest

iso_forest = IsolationForest(contamination=0.05, random_state=42)
iso_forest.fit(features)

# Add anomaly scores and labels to the aligned dataset
aligned_data['Anomaly_Score'] = iso_forest.decision_function(features)
aligned_data['Anomaly'] = iso_forest.predict(features)

# Filter for anomalies (label = -1)
anomalies_if = aligned_data[aligned_data['Anomaly'] == -1]

# Display anomalies
print("\nAnomalies detected by Isolation Forest:")
print(anomalies_if[['Date', 'Anomaly_Score']])

```

```
Anomalies detected by Isolation Forest:
          Date  Anomaly_Score
9   2016-08-22      -0.014817
30  2017-09-11      -0.048949
31  2017-09-18      -0.204188
```

```
# Convert the Isolation forest anomalies results output to a DataFrame
anomalies_if_df = pd.DataFrame(anomalies_if)

# Display the DataFrame using Pandas
from IPython.display import display

# Display the DataFrame
display(anomalies_if_df)

# Define the file path for saving the merged dataframe
output_file_path = "/content/drive/MyDrive/CIND820/AnomaliesDetected_Isolation_Forest_Output.csv"

```

```
# Save the merged dataframe to the specified location as a CSV file
anomalies_if_df.to_csv(output_file_path, index=False)

```

	Date	Bloom Extent (KM2)	Bloom Severity (ug/L km2)	Chlorophyll_a	CTD Temperature (°C)	Total Phosphorus (µg P/L)	Particulate Organic Nitrogen (mg/L)	Bloom_Severity_Z	Anomaly_Score	Anomaly
9	2016-08-22	560.88	17480.22	153.321429	25.033333	117.81250	0.3375	0.470322	-0.014817	-1
30	2017-09-11	1062.99	33721.23	225.735714	18.675000	51.80750	0.8025	2.042731	-0.048949	-1
31	2017-09-18	1264.59	39543.10	1087.804667	20.975000	42.47375	0.4375	2.606389	-0.204188	-1

Next steps: Generate code with `anomalies_if_df` View recommended plots New interactive sheet

3.2.2 DBSCAN Clustering

Use clustering to identify outliers in the data.

```
# Import DBSCAN
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

# Standardize features
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Fit DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan.fit(features_scaled)

# Add cluster labels to aligned data
aligned_data['Cluster'] = dbscan.labels_

# Identify noise points (anomalies)
anomalies_dbscan = aligned_data[aligned_data['Cluster'] == -1]

print("\nAnomalies detected by DBSCAN:")
print(anomalies_dbscan[['Date', 'Cluster']])

```

```
Anomalies detected by DBSCAN:
         Date  Cluster
```

```
9  2016-08-22       -1
15 2016-10-03       -1
16 2016-10-11       -1
17 2016-10-17       -1
30 2017-09-11       -1
31 2017-09-18       -1
33 2017-10-02       -1
35 2017-10-16       -1
```

```
# Convert the DBSCAN Clustering anomalies results output to a DataFrame
anomalies_dbscan_df = pd.DataFrame(anomalies_dbscan)

# Display the DataFrame using Pandas
from IPython.display import display

# Display the DataFrame
display(anomalies_dbscan_df)

# Define the file path for saving the merged dataframe
output_file_path = "/content/drive/MyDrive/CIND820/AnomaliesDetected_DBSCAN_Clustering_Output.csv"

# Save the merged dataframe to the specified location as a CSV file
anomalies_dbscan_df.to_csv(output_file_path, index=False)
```

	Date	Bloom Extent (KM2)	Bloom Severity (ug/L km2)	Chlorophyll_a	CTD Temperature (°C)	Total Phosphorus (µg P/L)	Particulate Organic Nitrogen (mg/L)	Bloom_Severity_Z	Anomaly_Score	Anomaly	Cluster
9	2016-08-22	560.88	17480.22	153.321429	25.033333	117.812500	0.337500	0.470322	-0.014817	-1	-1
15	2016-10-03	180.36	5748.81	89.309231	18.666667	81.375000	0.155000	-0.665481	0.059830	1	-1
16	2016-10-11	177.75	5623.83	5.950833	17.116667	61.275000	0.157500	-0.677581	0.048634	1	-1
17	2016-10-17	176.85	5716.61	4.143333	16.791667	46.387500	0.092500	-0.668598	0.029412	1	-1
30	2017-09-11	1062.99	33721.23	225.735714	18.675000	51.807500	0.802500	2.042731	-0.048949	-1	-1
31	2017-09-18	1264.59	39543.10	1087.804667	20.975000	42.473750	0.437500	2.606389	-0.204188	-1	-1
33	2017-10-02	1226.61	32501.73	126.275385	19.433333	56.975000	0.353750	1.924663	0.048883	1	-1
35	2017-10-16	722.07	17918.82	15.100000	16.609091	50.082857	0.237143	0.512786	0.014817	1	-1

Next steps: Generate code with anomalies_dbscan_df View recommended plots New interactive sheet

3.3 Model Evaluation and Comparison

Compare models based on prediction accuracy to select the best-performing model.

```
# Evaluate ARIMA Model
from sklearn.metrics import mean_squared_error
mse_arima = mean_squared_error(test_data, forecast_arima)
print(f"ARIMA Model MSE: {mse_arima}")

# Evaluate LSTM Model
mse_lstm = mean_squared_error(test_data, lstm_predictions)
print(f"LSTM Model MSE: {mse_lstm}")

```

```
ARIMA Model MSE: 106344223.51937824
LSTM Model MSE: 16706222.720046317
```

Summary of Section 3

Model Development and Training:

- ARIMA and LSTM models compared; LSTM showed better handling of non-linear patterns.
- Key Insight: LSTM effectively forecasts bloom severity.

Anomaly Detection:

- Isolation Forest and DBSCAN flagged environmental anomalies.
- Key Insight: Detected anomalies often align with extreme bloom events.

4. Integration and Deployment

4.1 Hybrid System Development

Integrate forecasting and anomaly detection to create an early warning system.

```
print(aligned_data['Forecasted_Bloom_Severity'].dropna().head())

```

```
Series([], Name: Forecasted Bloom Severity, dtype: float64)
```

```
print(aligned_data[
    (aligned_data['Forecasted_Bloom_Severity'] > warning_threshold) |
    (aligned_data['Anomaly'] == -1)
])

```

```
            Bloom Extent (KM2)  Bloom Severity (ug/L km2)  Chlorophyll_a  \
Date
2016-08-22              560.88                   17480.22     153.321429
2017-09-11             1062.99                   33721.23     225.735714
2017-09-18             1264.59                   39543.10    1087.804667

            CTD Temperature (°C)  Total Phosphorus (µg P/L)  \
Date
2016-08-22             25.033333                  117.81250
2017-09-11             18.675000                   51.80750
2017-09-18             20.975000                   42.47375

            Particulate Organic Nitrogen (mg/L)  Bloom_Severity_Z  \
Date
2016-08-22                               0.3375          0.470322
2017-09-11                               0.8025          2.042731
2017-09-18                               0.4375          2.606389

            Anomaly_Score  Anomaly  Cluster  Forecasted_Bloom_Severity  \
Date
2016-08-22      -0.014817       -1       -1                        NaN
2017-09-11      -0.048949       -1       -1                        NaN
2017-09-18      -0.204188       -1       -1                        NaN

             Alert
Date
```