Statistical Analysis of Lettuce Growth Using Environmental Data

This report provides a comprehensive summary of the Kaggle dataset titled Lettuce_Growth_Days, which documents daily environmental conditions, including temperature, humidity, pH level, and total dissolved solids (TDS), for 70 lettuce samples from crop establishment to full growth. The dataset offers valuable insights into the relationships between environmental factors and the number of days required for lettuce to reach maturity.

To explore these relationships, the analysis follows a structured approach. As the compelling dataset contains three files which are "lettuce_dataset.csv", "lettuce_dataset_updated.csv", and "unseen_data.csv" we've proceded to verify the data structure in each dataset.

Datasets Summarize

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import drive
from IPython.display import display, Markdown

drive.mount('/content/drive')

file_path_dt1 = '/content/drive/My Drive/DataAnalysis/Ask_Questions/lettuce_dataset.csv'
file_path_dt2 = '/content/drive/My Drive/DataAnalysis/Ask_Questions/lettuce_dataset_upda
file_path_dt3 = '/content/drive/My Drive/DataAnalysis/Ask_Questions/unseen_data.csv'
df_original = pd.read_csv(file_path_dt1, encoding='latin-1')
df_updated = pd.read_csv(file_path_dt2, encoding='latin-1')
df_unseen = pd.read_csv(file_path_dt3, encoding='latin-1')
display(Markdown("<br><br>"))
print("Original Dataset:")
display(df_original.head())
display(Markdown("<br><br>"))
print(df_original.info())
display(Markdown("<br><br>"))
print("Updated Dataset:")
display(df_updated.head())
display(Markdown("<br><br>"))
print(df_updated.info())
display(Markdown("<br><br>"))
print("Unseen Dataset:")
display(df_unseen.head())
display(Markdown("<br><br>"))
```

```
print(df_unseen.info())
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mo

Original Dataset:

	Plant_ID	Date	Temperature (°C)	Humidity (%)	TDS Value (ppm)	pH Level	Growth Days
0	1	8/3/2023	33.4	53	582	6.4	1
1	1	8/4/2023	33.5	53	451	6.1	2
2	1	8/5/2023	33.4	59	678	6.4	3
3	1	8/6/2023	33.4	68	420	6.4	4
4	1	8/7/2023	33.4	74	637	6.5	5

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3169 entries, 0 to 3168
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Plant_ID          3169 non-null   int64
 1   Date              3169 non-null   object
 2   Temperature (°C)  3169 non-null   float64
 3   Humidity (%)      3169 non-null   int64
 4   TDS Value (ppm)   3169 non-null   int64
 5   pH Level          3169 non-null   float64
 6   Growth Days       3169 non-null   int64
dtypes: float64(2), int64(4), object(1)
memory usage: 173.4+ KB
None
```

Updated Dataset:

	Plant_ID	Date	Temperature (°C)	Humidity (%)	TDS Value (ppm)	pH Level	Growth Days	Temperature (F)	Humidi
0	1	8/3/2023	33.4	53	582	6.4	1	92.12	0.
1	1	8/4/2023	33.5	53	451	6.1	2	92.30	0.
2	1	8/5/2023	33.4	59	678	6.4	3	92.12	0.
3	1	8/6/2023	33.4	68	420	6.4	4	92.12	0.
4	1	8/7/2023	33.4	74	637	6.5	5	92.12	0.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3169 entries, 0 to 3168
Data columns (total 9 columns):
```

```
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Plant_ID          3169 non-null   int64
 1   Date              3169 non-null   object
 2   Temperature (°C)  3169 non-null   float64
 3   Humidity (%)      3169 non-null   int64
 4   TDS Value (ppm)   3169 non-null   int64
 5   pH Level          3169 non-null   float64
 6   Growth Days       3169 non-null   int64
 7   Temperature (F)   3169 non-null   float64
 8   Humidity          3169 non-null   float64
dtypes: float64(4), int64(4), object(1)
memory usage: 222.9+ KB
None
```

Unseen Dataset:

	Plant_ID	Date	Temperature (°C)	Humidity (%)	pH Level	TDS Value (ppm)
0	1	9/15/2023	30	60	6.5	500
1	1	9/16/2023	31	62	6.6	505
2	1	9/17/2023	26	58	6.4	495
3	1	9/18/2023	32	57	6.7	490
4	1	9/19/2023	25	59	6.5	500

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Plant_ID          30 non-null     int64
 1   Date              30 non-null     object
 2   Temperature (°C)  30 non-null     int64
 3   Humidity (%)      30 non-null     int64
 4   pH Level          30 non-null     float64
 5   TDS Value (ppm)   30 non-null     int64
dtypes: float64(1), int64(4), object(1)
memory usage: 1.5+ KB
None
```

Seems to be the updated dataset looks like the original dataset with slightly difference in total number of columns. The original dataset contains 7 columns and the updated dataset contains 9 columns instead.

On the other hand, the unseen_dataset, contains 6 columns but only 30 records in total. This dataset absecenses column "Growth Days". As the main purpose of the datasets from Lettuce datasets is to perform certain predictive model using either the original lettuce_dataset or the updated dataset and assess this model against the unseen dataset, we've decided to start performing cross-validation to verify any discrepancies that allow us select the correct dataset.

As the updated dataset expose two additional columns: Temperature in Fahrenheit degrees and Humidity in decimal values, and it is likely these columns doesn't provide additional value to the dataset, so we've proceeded to drop those columns to follow to the cross-validation process.

```
df_lettuce_updated = df_updated.drop(columns=["Temperature (F)", "Humidity"])
display(Markdown("<br><br>"))
print("Lettuce Dataset Updated without two dropped columns")
print(df_lettuce_updated.head())
display(Markdown("<br><br>"))
print(df_lettuce_updated.info())

```

```
Lettuce Dataset Updated without two dropped columns
   Plant_ID      Date  Temperature (°C)  Humidity (%)  TDS Value (ppm)  \
0         1  8/3/2023              33.4            53              582
1         1  8/4/2023              33.5            53              451
2         1  8/5/2023              33.4            59              678
3         1  8/6/2023              33.4            68              420
4         1  8/7/2023              33.4            74              637

   pH Level  Growth Days
0       6.4            1
1       6.1            2
2       6.4            3
3       6.4            4
4       6.5            5
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3169 entries, 0 to 3168
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Plant_ID          3169 non-null   int64
 1   Date              3169 non-null   object
 2   Temperature (°C)  3169 non-null   float64
 3   Humidity (%)      3169 non-null   int64
 4   TDS Value (ppm)   3169 non-null   int64
 5   pH Level          3169 non-null   float64
 6   Growth Days       3169 non-null   int64
dtypes: float64(2), int64(4), object(1)
memory usage: 173.4+ KB
None
```

Once both datasets contains similar structures we proceeded to the cross-validation process.

Cross-Validation Datasets

```
comparison = df_original.eq(df_lettuce_updated)
mismatches = ~comparison

# Count mismatches per column
mismatch_counts = mismatches.sum(axis=0)
print("Mismatch counts per column:")
print(mismatch_counts)

# Output mismatched rows
mismatch_rows = df_original[~comparison.all(axis=1)]
display(Markdown("<br><br>"))
```

```
print("Rows with mismatches:")
print(mismatch_rows)

```

```
Mismatch counts per column:
Plant_ID            0
Date                0
Temperature (°C)    0
Humidity (%)        0
TDS Value (ppm)     0
pH Level            0
Growth Days         3
dtype: int64



Rows with mismatches:
      Plant_ID       Date  Temperature (°C)  Humidity (%)  TDS Value (ppm)  \
2489        55  9/17/2023              31.6            69              539
2490        55  9/18/2023              29.4            55              554
2491        55  9/19/2023              31.5            51              527

      pH Level  Growth Days
2489       6.6           45
2490       6.6           46
2491       6.2           47
```

The cross-validation process indicated discrepancies in three rows between the original and the updated dataset. These discrepancies steam in the sample 55 contained repeated values in the column Growth Days. Once visualy validating this discrepancies we've verified that updated dataset contains the correct records and we decided to use it to address the current analysis.

Next step includes Exploratory Data Analysis - EDA.

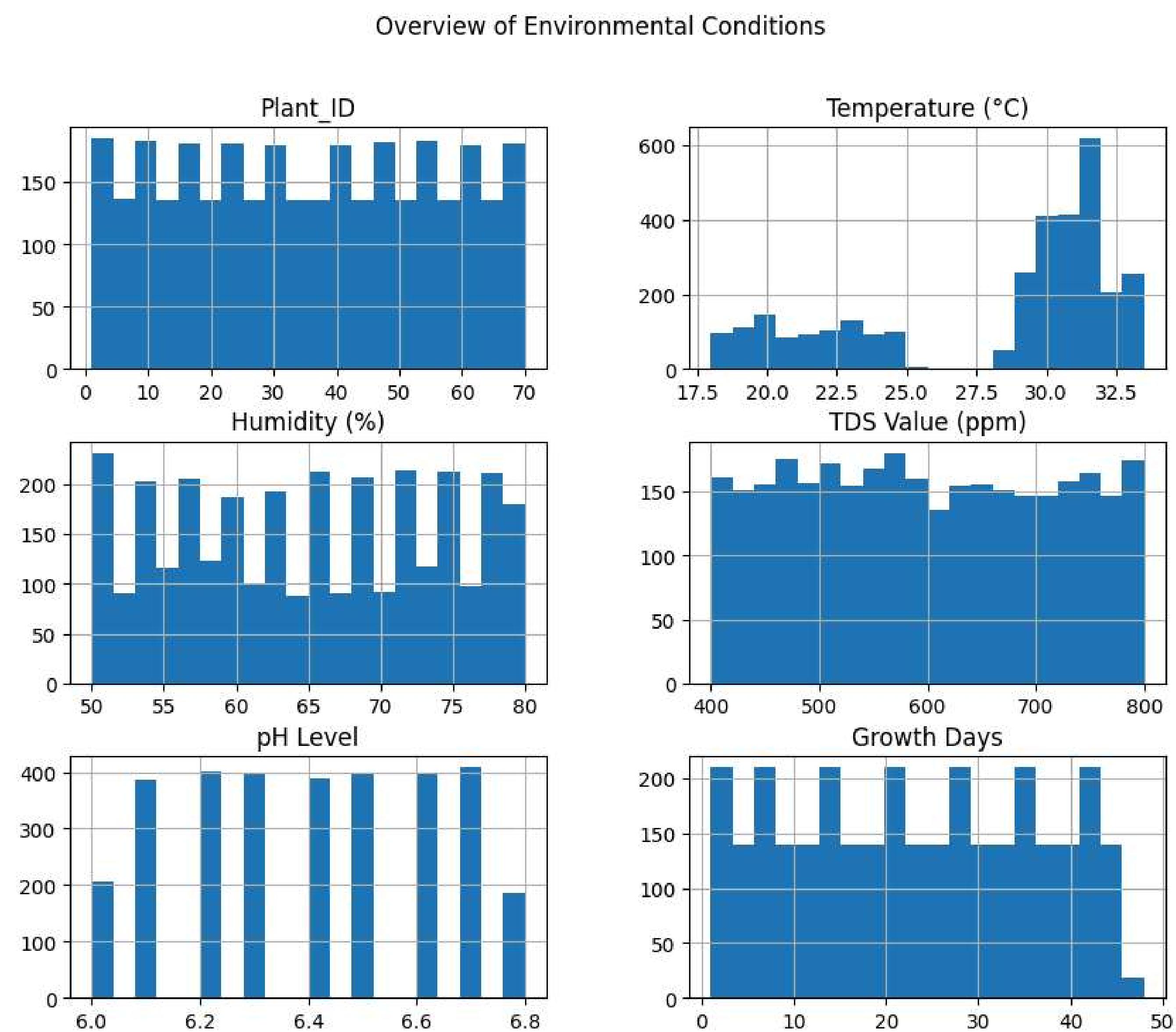
Exploratory Data Analysis

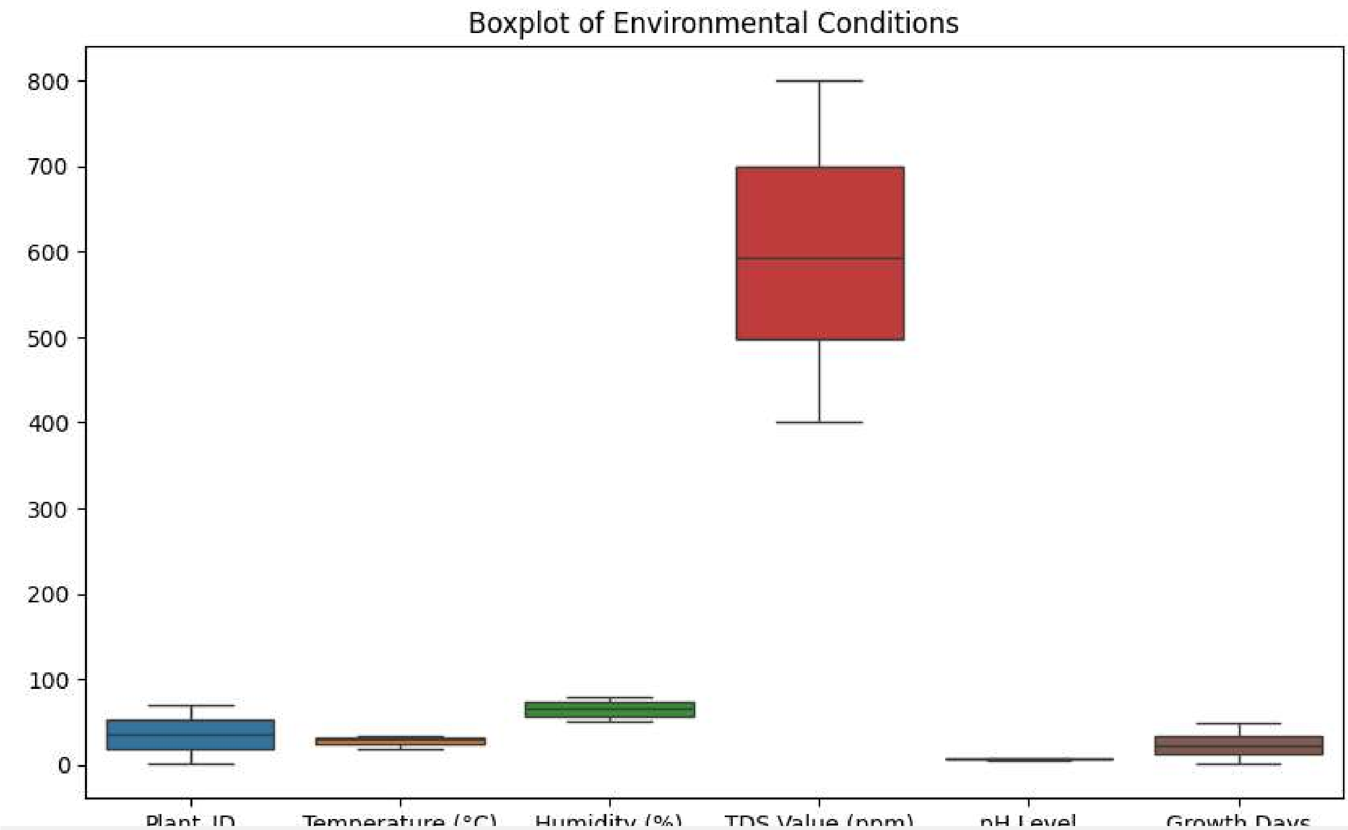
```
data = {"Variable": [], "Mean": [], "Median": [], "Standard Deviation": [], "Minimum": [
for col in ['Temperature (°C)', 'Humidity (%)', 'TDS Value (ppm)', 'pH Level']:
  data["Variable"].append(col)
  data["Mean"].append(df_lettuce_updated[col].mean())
  data["Median"].append(df_lettuce_updated[col].median())
  data["Standard Deviation"].append(df_lettuce_updated[col].std())
  data["Minimum"].append(df_lettuce_updated[col].min())
  data["Maximum"].append(df_lettuce_updated[col].max())
  data["25th Percentile"].append(df_lettuce_updated[col].quantile(0.25))
  data["50th Percentile"].append(df_lettuce_updated[col].quantile(0.5))
  data["75th Percentile"].append(df_lettuce_updated[col].quantile(0.75))

df_stats = pd.DataFrame(data)

display(Markdown("<br><br>"))
```

```
styled_table =df_stats.style.set_caption("Descriptive Statistics").set_table_styles(
    [{'selector': 'caption', 'props': [('text-align', 'center'), ('font-size', '16px'),

display(styled_table)

display(Markdown("<br><br>"))

df_lettuce_updated.hist(figsize=(10, 8), bins=20)
plt.suptitle('Overview of Environmental Conditions')
plt.show()

display(Markdown("<br><br>"))

plt.figure(figsize=(10, 6))
sns.boxplot(data=df_lettuce_updated.select_dtypes(include=[np.number]), orient='v')
plt.title('Boxplot of Environmental Conditions')
plt.show()
```

Descriptive Statistics

	Variable	Mean	Median	Standard Deviation	Minimum	Maximum	25th Percentile	Per
0	Temperature (°C)	28.142222	30.200000	4.670521	18.000000	33.500000	23.600000	30
1	Humidity (%)	64.873462	65.000000	8.988985	50.000000	80.000000	57.000000	65
2	TDS Value (ppm)	598.045440	593.000000	115.713047	400.000000	800.000000	498.000000	593
3	pH Level	6.399211	6.400000	0.234418	6.000000	6.800000	6.200000	6



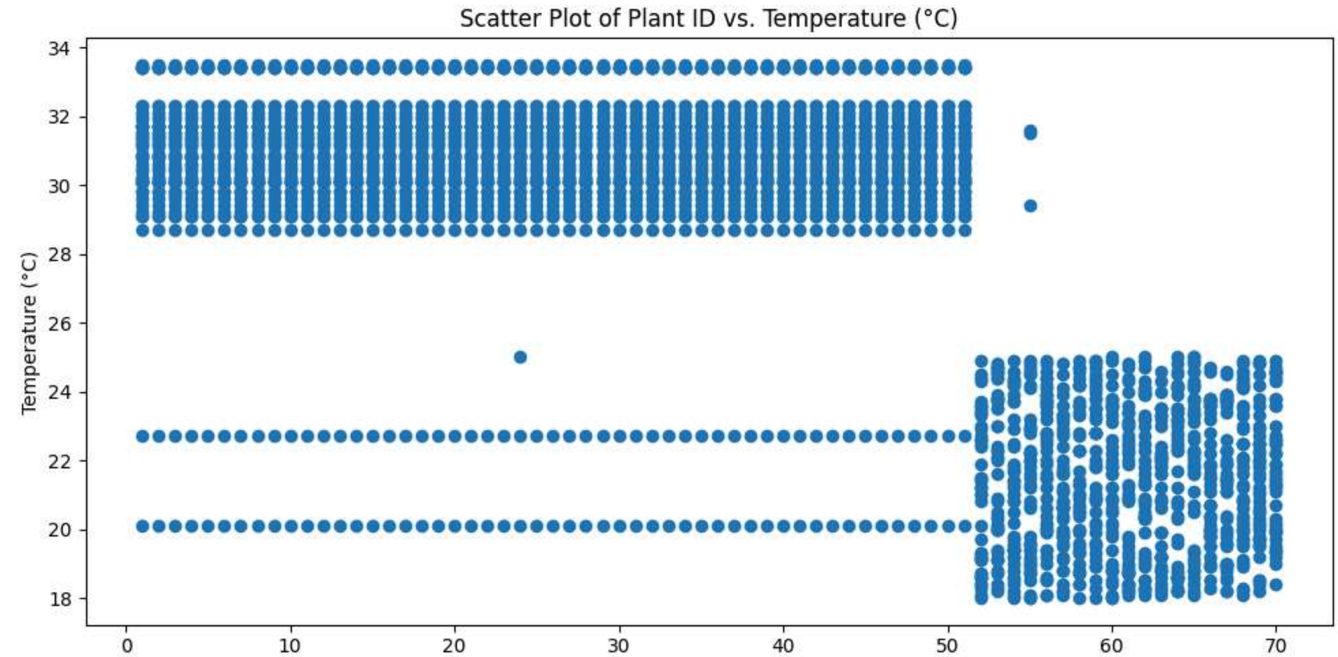


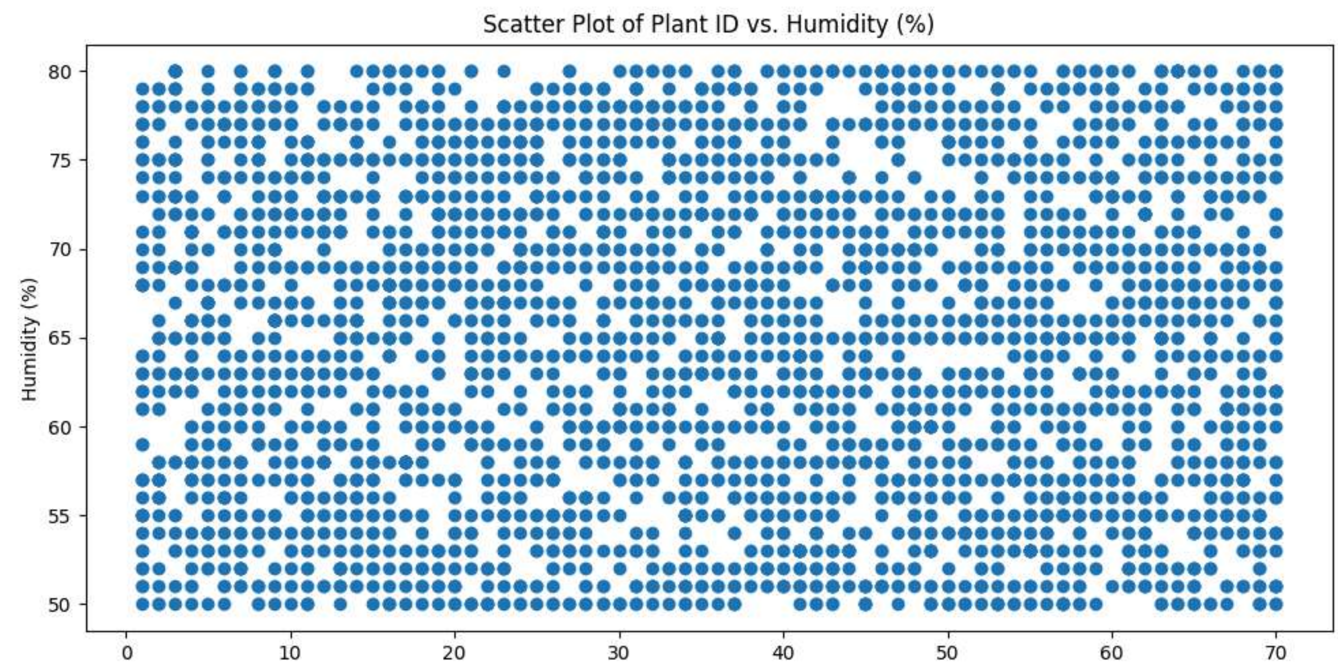
The exploratory data analysis, including descriptive statistics, feature histograms, and visual inspection via boxplots, indicates that the environmental factors (temperature, humidity, pH level, and total dissolved solids) exhibit distributions that are approximately normal. However, to validate the left-skewed distribution observed in temperature, we used scatter plots to examine its behavior across the entire sample. Additionally, to analyze other features, we created scatter plots for each feature across all samples. Here’s what we found.

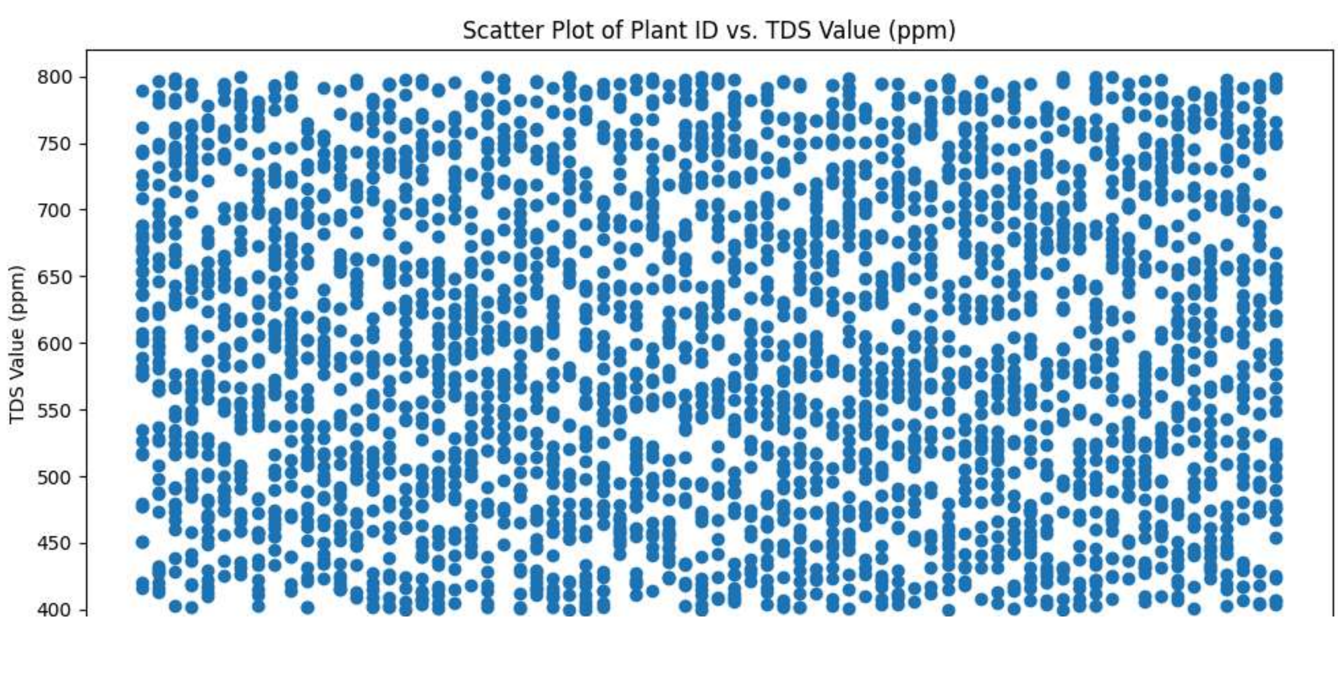
Features Scatter Plots

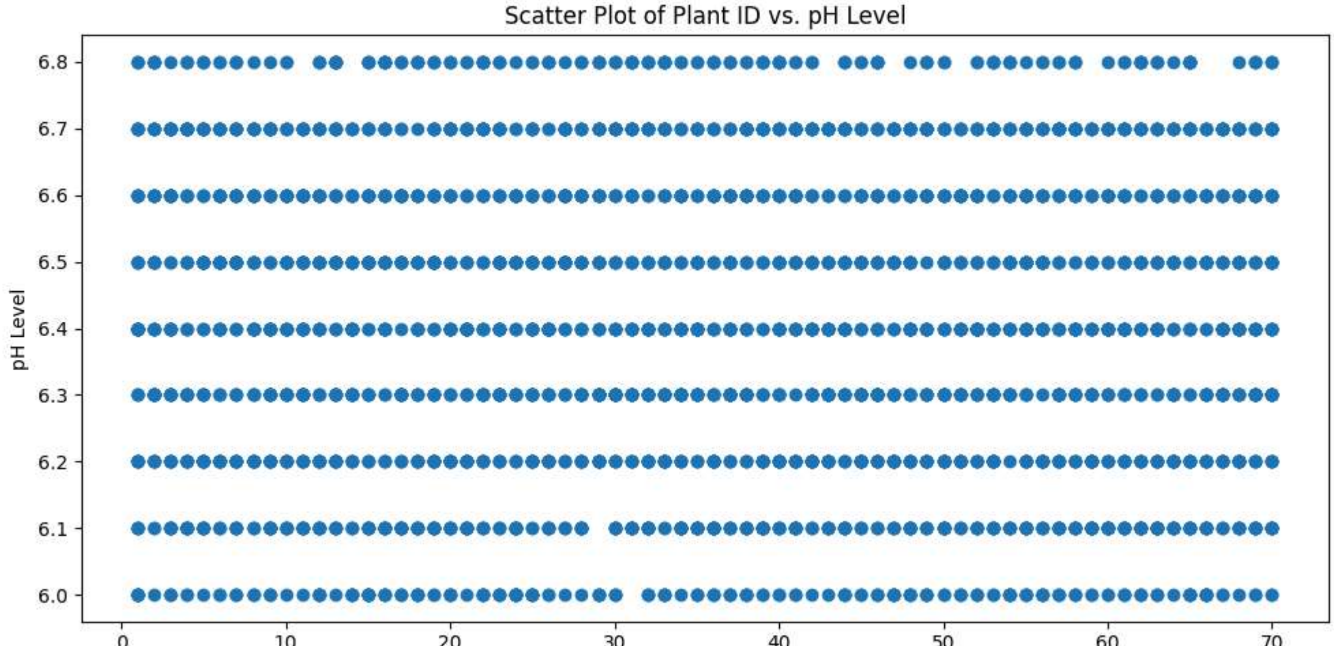
```
# Define the features for the y-axis
features = ['Temperature (°C)', 'Humidity (%)', 'TDS Value (ppm)', 'pH Level']

# Create a figure and axes for the plots
fig, axes = plt.subplots(len(features), 1, figsize=(10, 5 * len(features)), sharex=False

# Iterate through features and create scatter plots
for i, feature in enumerate(features):
    axes[i].scatter(df_lettuce_updated['Plant_ID'], df_lettuce_updated[feature])
    axes[i].set_ylabel(feature)
    axes[i].set_title(f'Scatter Plot of Plant ID vs. {feature}')

# Set x-axis label for the bottom subplot
axes[-1].set_xlabel('Plant ID')
```

```

# Adjust layout and display the plot
plt.tight_layout()
plt.show()
```


Scatter Plot of Plant ID vs. Temperature (°C)
Temperature (°C)
34
32
30
28
26
24
22
20
18
0
10
20
30
40
50
60
70


Scatter Plot of Plant ID vs. Humidity (%)
Humidity (%)
80
75
70
65
60
55
50
0
10
20
30
40
50
60
70


Scatter Plot of Plant ID vs. TDS Value (ppm)
TDS Value (ppm)
800
750
700
650
600
550
500
450
400



As observed in the scatter plots, temperature appears to act as a blocking variable, while other features exhibit an approximately normal distribution, suggesting natural conditions.

The distribution of temperature values is asymmetric, forming two distinct clusters:

For Plant_IDs below 51, temperatures are concentrated around 30–34°C.

For Plant_IDs above 51, temperatures are lower and more dispersed.

This pattern suggests a shift in temperature distribution between the two Plant_ID groups, indicating that temperature may not be naturally distributed across all samples. As a result, its correlation with growth days could be distorted.

If temperature is a controlled variable, its effect on growth days may not reflect real-world behavior but rather be an artifact of experimental conditions.

To address this, we conducted a stratified analysis, separating the dataset into two groups:

Plant_IDs 1–51

Plant_IDs 52–70

This approach allows us to compare trends within each group separately, ensuring a more accurate interpretation of temperature's influence.

```
df_lettuce_updated.groupby("Plant_ID")["Temperature (°C)"].describe()
```

Plant_ID	count	mean	std	min	25%	50%	75%	max
1	45.0	30.628889	2.377415	20.1	30.100	30.90	31.70	33.5
2	45.0	30.628889	2.377415	20.1	30.100	30.90	31.70	33.5
3	47.0	30.606383	2.329154	20.1	30.100	30.90	31.70	33.5
4	48.0	30.645833	2.302446	20.1	30.100	30.90	31.70	33.5
5	45.0	30.628889	2.377415	20.1	30.100	30.90	31.70	33.5
...	...	...	...	...	...	...	...	...
66	45.0	21.333333	1.819715	18.3	19.900	21.10	23.20	24.7
67	45.0	21.311111	1.978661	18.2	19.600	21.40	23.40	24.6
68	45.0	21.600000	2.054485	18.1	20.000	21.30	23.50	24.9
69	45.0	21.375556	1.855968	18.2	19.800	21.20	22.50	24.9
70	46.0	21.773913	1.879886	18.4	20.025	21.65	23.45	24.9

70 rows × 8 columns

The table above shows significant variation in temperature statistics between samples from Plant_IDs 1 to 65 and those above 66.

In the first group, the minimum temperature is 20.1°C, the mean is approximately 30.6°C, and the maximum is 33.5°C. In contrast, in the second group, the minimum temperature is 18.1°C, the mean ranges from 21.3°C to 21.7°C, and the maximum is 24.9°C. These results confirm our earlier findings.

Next, we will conduct a similar analysis for other features.

```
# Creating a new variable for groups
df_lettuce_updated['Temperature_Group'] = df_lettuce_updated['Plant_ID'].apply(lambda x:

# Compare means of environmental factors in each group
df_lettuce_updated.groupby('Temperature_Group')[['Humidity (%)', 'TDS Value (ppm)', 'pH

```

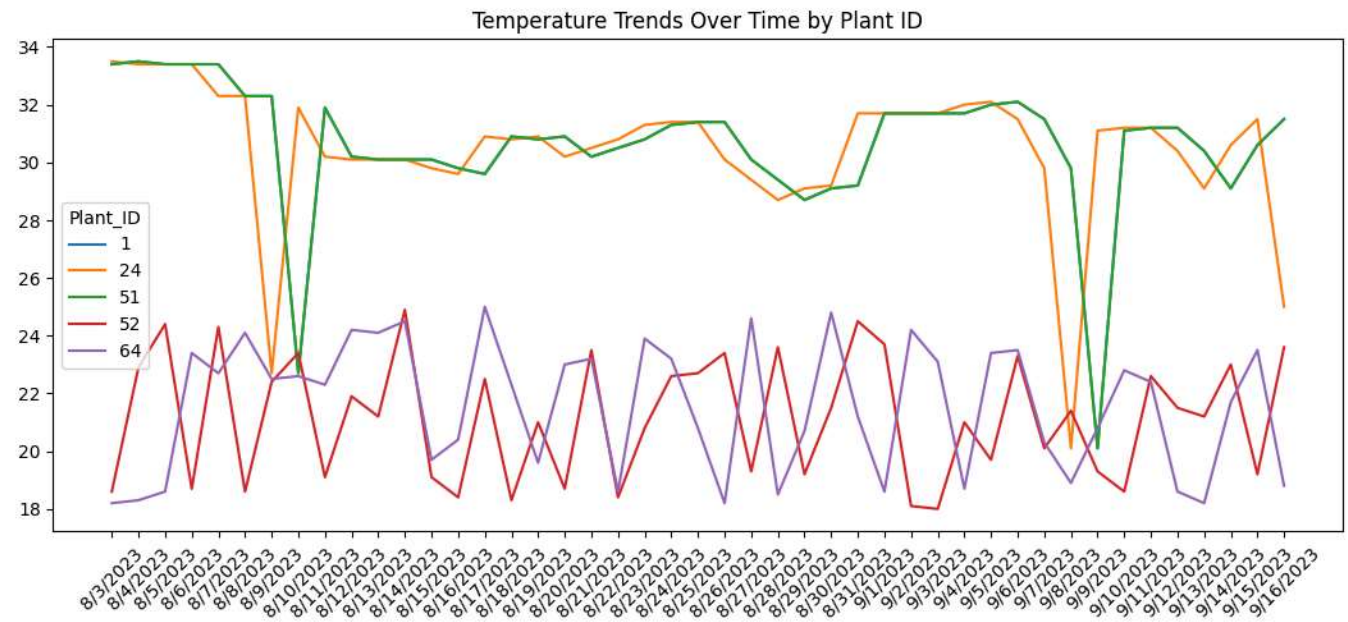
	Humidity (%)								TDS Value (pp	
	count	mean	std	min	25%	50%	75%	max	count	mean
Temperature_Group										
Group 1	2309.0	65.017324	8.969120	50.0	57.0	65.0	73.00	80.0	2309.0	597.25
Group 2	860.0	64.487209	9.036007	50.0	56.0	65.0	72.25	80.0	860.0	600.16

2 rows × 24 columns

However, in this case, the table does not show significant variation in the statistical values of the other features.

To better visualize temperature trends over time for different samples in each group, we will now plot a line chart.

```
from matplotlib.legend_handler import HandlerLine2D

plt.figure(figsize=(13, 5))

filtered_data = df_lettuce_updated[df_lettuce_updated['Plant_ID'].isin([1, 24, 51, 52, 6

handles = []
labels = []
for plant_id in [1, 24, 51, 52, 64]:
    plant_data = filtered_data[filtered_data['Plant_ID'] == plant_id]
    line, = plt.plot(plant_data["Date"], plant_data["Temperature (°C)"], label=plant_id)
    handles.append(line)
    labels.append(plant_id)


plt.xticks(rotation=45)
plt.title("Temperature Trends Over Time by Plant ID")

plt.legend(handles, labels, title="Plant_ID", handler_map={type(handles[0]): HandlerLine
display(Markdown("<br><br>"))
plt.show()
```



This graph exhibits a reduced temperature range for Plant_IDs between 1 and 51. In this group, most temperature records show little variation, though a few outliers fall within the range of the other group.

In contrast, the group with Plant_IDs between 52 and 70 shows a broader distribution without significant variation, suggesting more natural temperature conditions.

Accordingly, it is necessary to perform Spearman's correlation to determine whether the relationships between growth days and other features vary depending on the temperature group.

```
import scipy.stats as stats

group1 == 'Group 1'
group2 == 'Group 2'

# Compute correlation for each group
correlations_group1 = group1[['Humidity (%)', 'TDS Value (ppm)', 'pH Level', 'Growth Day
correlations_group2 = group2[['Humidity (%)', 'TDS Value (ppm)', 'pH Level', 'Growth Day

```