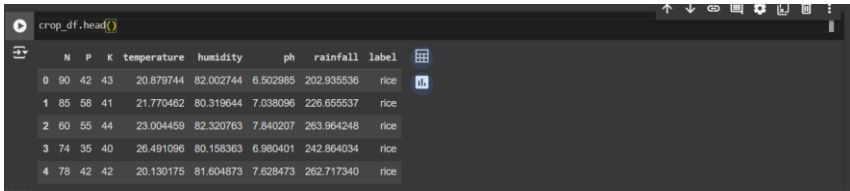
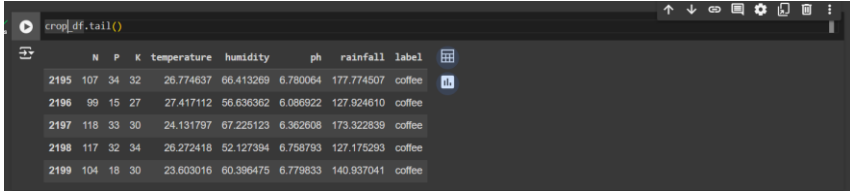
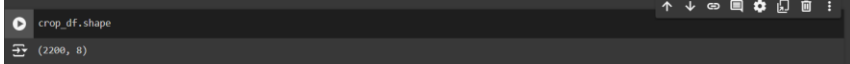


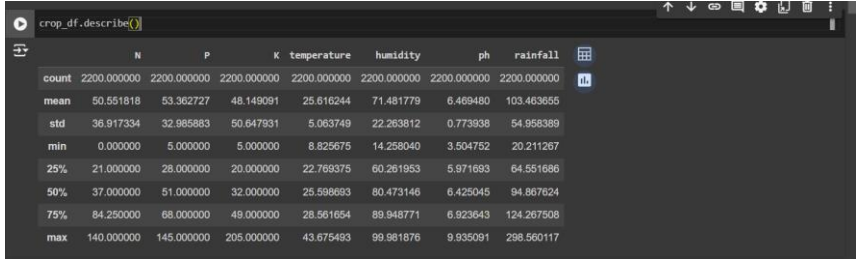
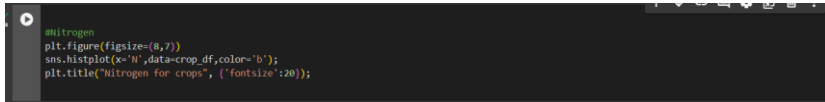
## Data Collection and Preprocessing Phase

Date	15 July 2024
Team ID	739737
Project Title	Crop Prediction using machine learning
Maximum Marks	6 Marks

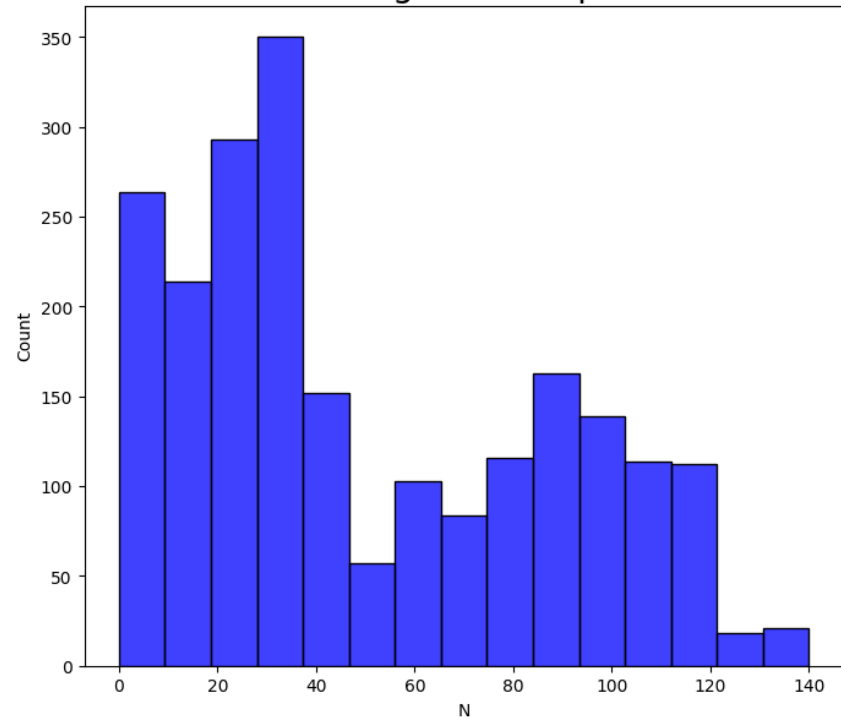
### Data Exploration and Preprocessing Template

Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

Section	Description
Data Overview	<p><b>#Structure of the data:</b></p>    <p><b>#Descriptive Statistical:</b></p> <p>Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.</p>

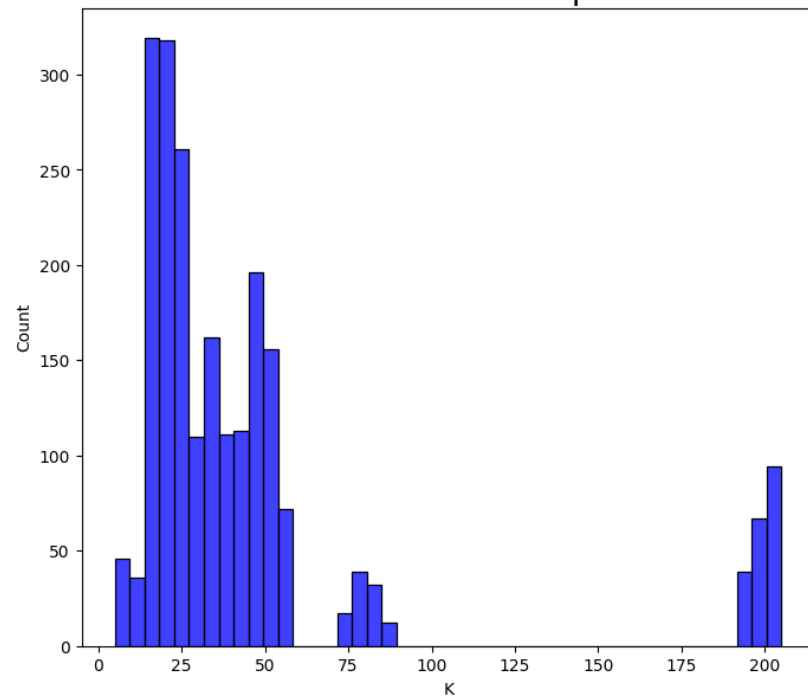
	
Univariate Analysis	<p>Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.</p> <p><b>Univariate Analysis:</b></p> <p>In simple words, univariate analysis is understanding the data with a single feature. We have displayed three different types of graphs and plots.</p> <p>For simple visualizations we can use the matplotlib. pyplot library. Here the plt. figure() command is used to determine the size of the plot.</p> <p>We have histogram for all features of the dataset which include phosphorus, humidity, temperature as well . The histogram shows the distribution of nitrogen fertilizers for crop.</p> 

Nitrogen for crops

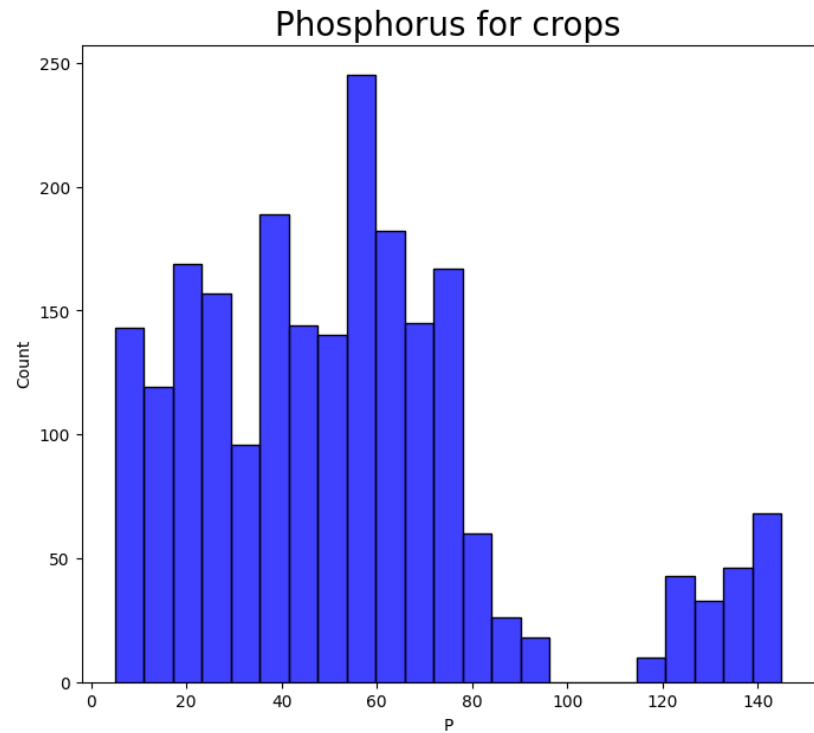


```
#Potassium
plt.figure(figsize=(8,7))
sns.histplot(x='K',data=crop_df,color='b');
plt.title('Potassium for crops',{'fontSize':20});
```

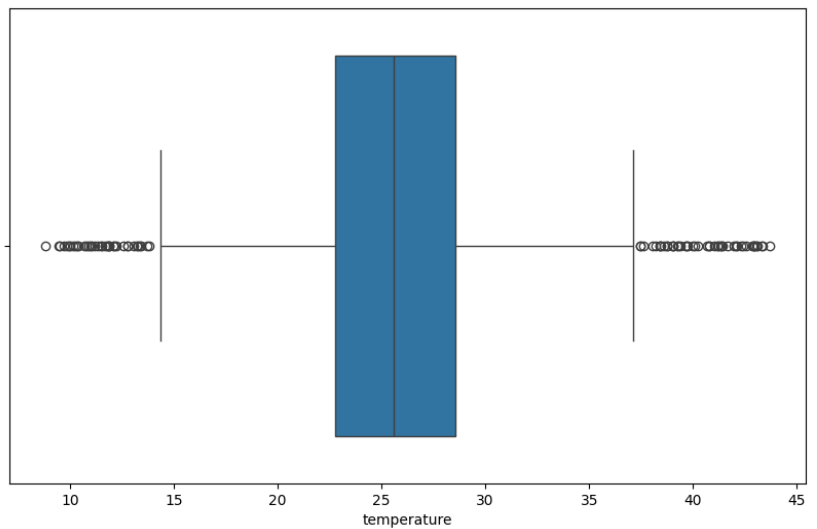
Potassium for crops



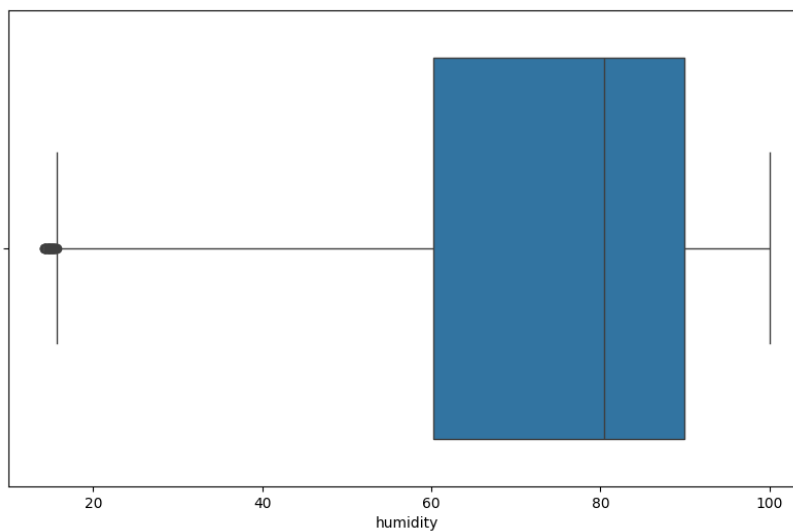
```
#phosphorus
plt.figure(figsize=(8,7))
sns.histplot(x='P',data=crop_df,color='b');
plt.title("Phosphorus for crops",('fontsize:20));
```



```
#temperature
plt.figure(figsize=(10,6))
sns.boxplot(x=crop_df.temperature);
```



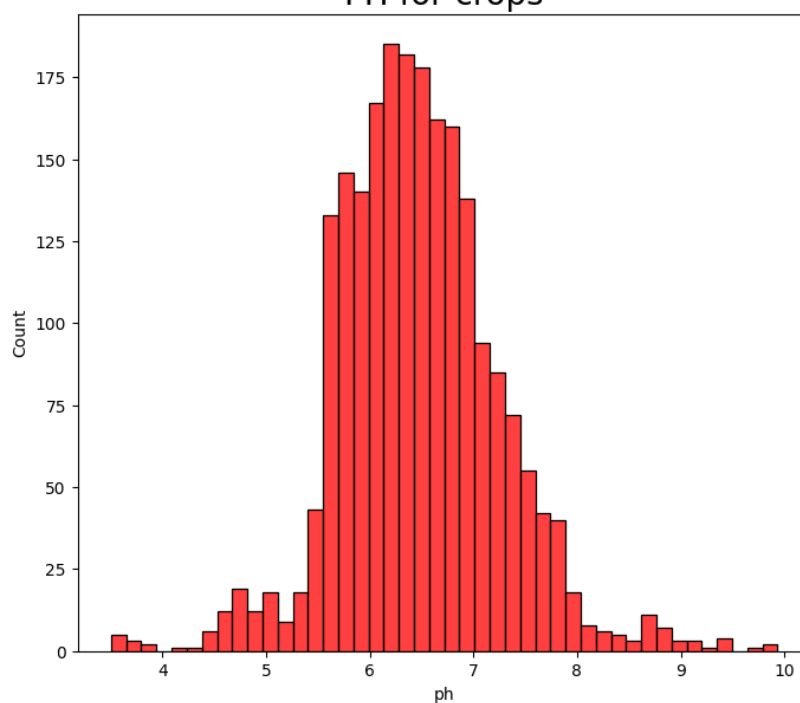
```
#humidity
plt.figure(figsize=(10,6))
sns.boxplot(x=crop_df.humidity);
```



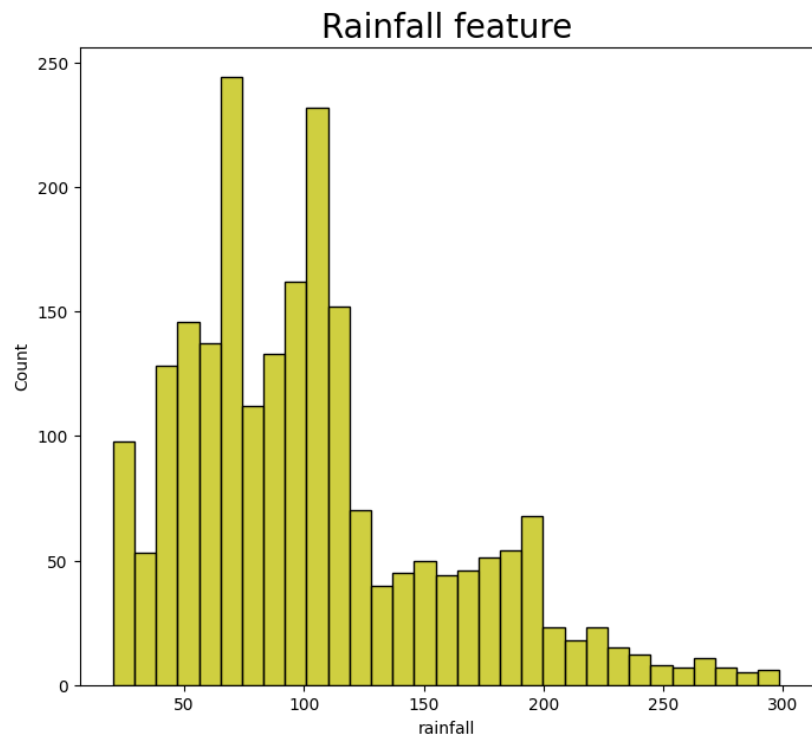
Here we have plotted Boxplot using seaborn library. These boxplots can be plotted without using any external library. We have plotted the boxplot using the inbuilt plot function in python.

```
#ph
plt.figure(figsize=(8,7))
sns.histplot(x='ph', data=crop_df,color='r')
plt.title("ph for crops",("fontSize:120));
```

PH for crops



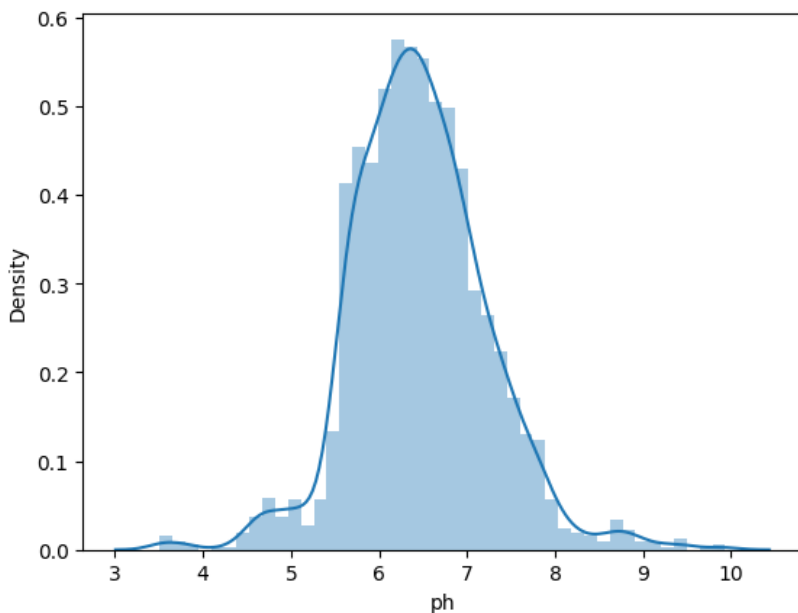
```
#rainfall
plt.figure(figsize=(8,7))
sns.histplot(x='rainfall', data=crop_df,color='y');
plt.title("rainfall feature",("fontSize:120));
```



Here we have plotted distribution plot using seaborn library. These plots can be plotted without using any external library. A Distplot or distribution plot, depicts the variation in the data distribution. Seaborn Distplot represents the overall distribution of continuous data variables.

```
sns.distplot(crop_df['ph'])
```

<ipython-input-18-2301da062035>:1: UserWarning:  
'distplot' is a deprecated function and will be removed in seaborn v0.14.0.  
Please adapt your code to use either 'displot' (a figure-level function with similar flexibility) or 'histplot' (an axes-level function for histograms).  
For a guide to updating your code to use the new functions, please see  
<https://gist.github.com/mwaskom/d64147ed2974457adb3727508be5231>  
sns.distplot(crop\_df['ph'])  
<Axes: xlabel='ph', ylabel='Density'>

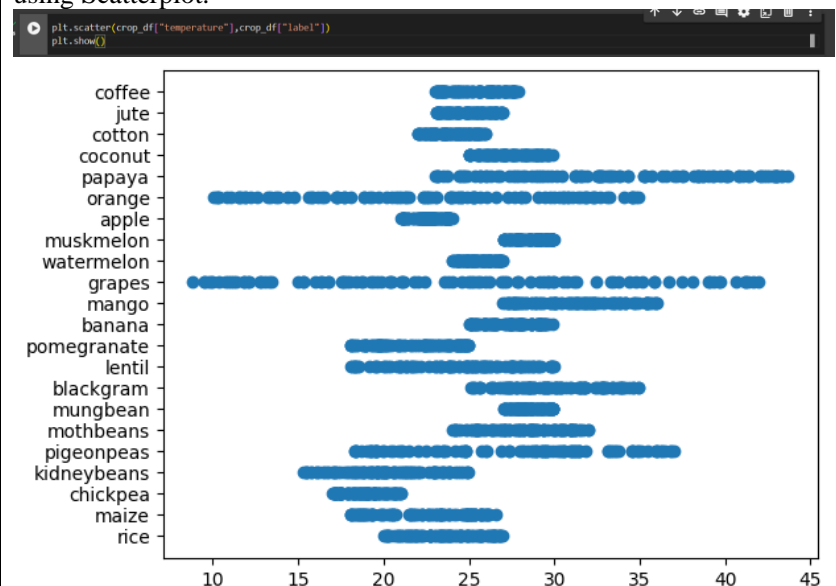


### Bivariate Analysis:

To find the relation between two features we use bivariate analysis. Here we are visualising the relationship between predicted crop and temperature.

#### #Scatter plot:

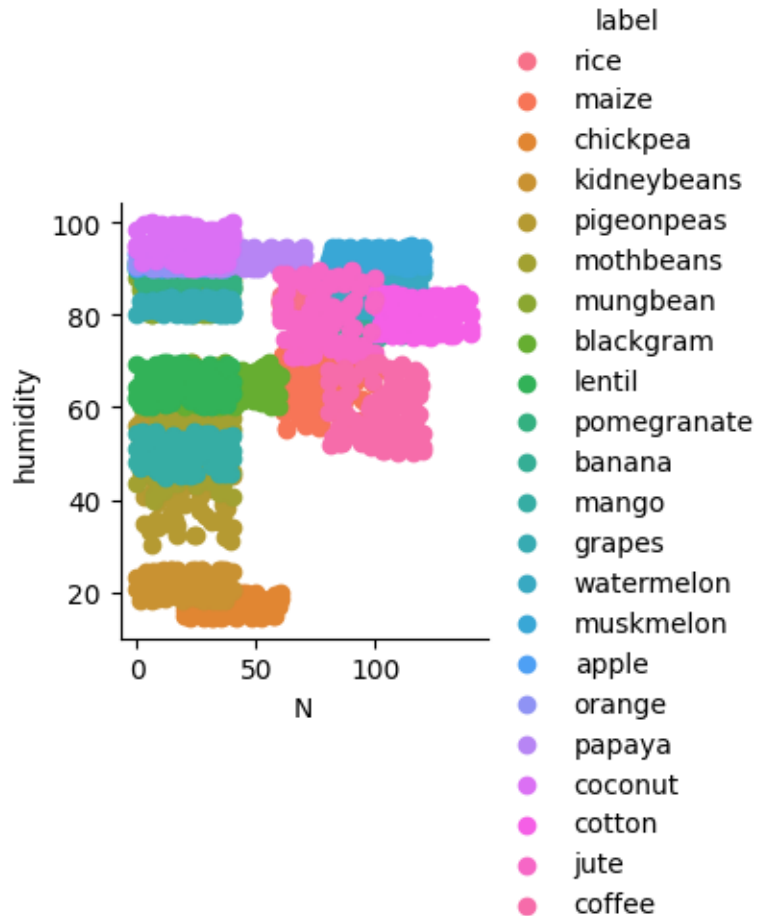
Scatterplot can be used with several semantic groupings which can help to understand well in a graph. They can plot two-dimensional graphics that can be enhanced by mapping up to three additional variables while using the semantics of hue, size, and style parameters. All the parameter control visual semantic which are used to identify the different subsets. Using redundant semantics can be helpful for making graphics more accessible. We have depicted the relationship between temperature and predicted crop using Scatterplot.



### Bivariate Analysis

### #FacetGrid:

FacetGrid class helps in visualizing distribution of one variable as well as the relationship between multiple variables separately within subsets of your dataset using multiple panels.



### Multivariate Analysis

#### Multivariate Analysis:

Multivariate analysis is a statistical technique used to analyse data that involves more than two variables. It aims to understand the relationships between multiple variables in a dataset by examining how they are related to each other and how they contribute to a particular outcome or phenomenon.

In multivariate analysis we try to find the relation between multiple features. This can be done primarily with the help of Correlation matrix.

```

# get correlations of each features in dataset
import pandas as pd
# if crop_df is not already a DataFrame, convert it
if isinstance(crop_df, np.ndarray):
    crop_df = pd.DataFrame(crop_df)
# calculate the correlation matrix
corrmat = crop_df.select_dtypes(include=['number']).corr()
# apply the background gradient styling
corrmat.style.background_gradient('coolwarm')
  
```

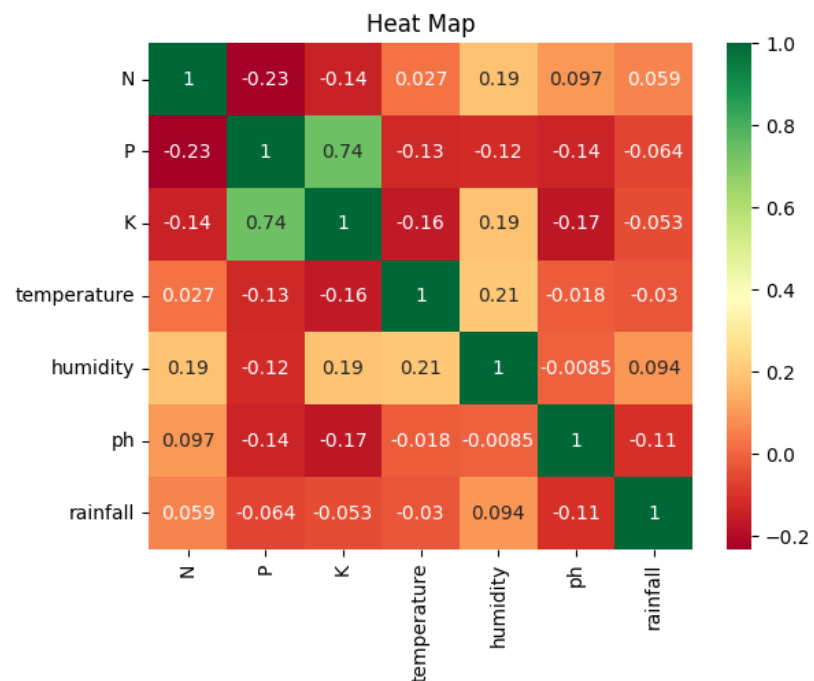


	N	P	K	temperature	humidity	ph	rainfall
N	1.000000	-0.231460	-0.140512	0.026504	0.190688	0.096683	0.059020
P	-0.231460	1.000000	0.736232	-0.127541	-0.118734	-0.138019	-0.063839
K	-0.140512	0.736232	1.000000	-0.160387	0.190859	-0.169503	-0.053461
temperature	0.026504	-0.127541	-0.160387	1.000000	0.205320	-0.017795	-0.030084
humidity	0.190688	-0.118734	0.190859	0.205320	1.000000	-0.008483	0.094423
ph	0.096683	-0.138019	-0.169503	-0.017795	-0.008483	1.000000	-0.109069
rainfall	0.059020	-0.063839	-0.053461	-0.030084	0.094423	-0.109069	1.000000

For multivariate analysis we will also plot a Heatmap

This code creates a heatmap that shows how much each column in a given data frame is related to each other column. It does this by first creating a new data frame that has all the columns of the original data frame except for 'Label' column. It then calculates the correlation between all the remaining columns and creates a matrix that shows these correlations. Finally, it generates a heatmap of this matrix using a library called Seaborn.

```
sns.heatmap(cormat, annot=True, cmap="magma")
plt.title("Heat Map")
plt.show()
```

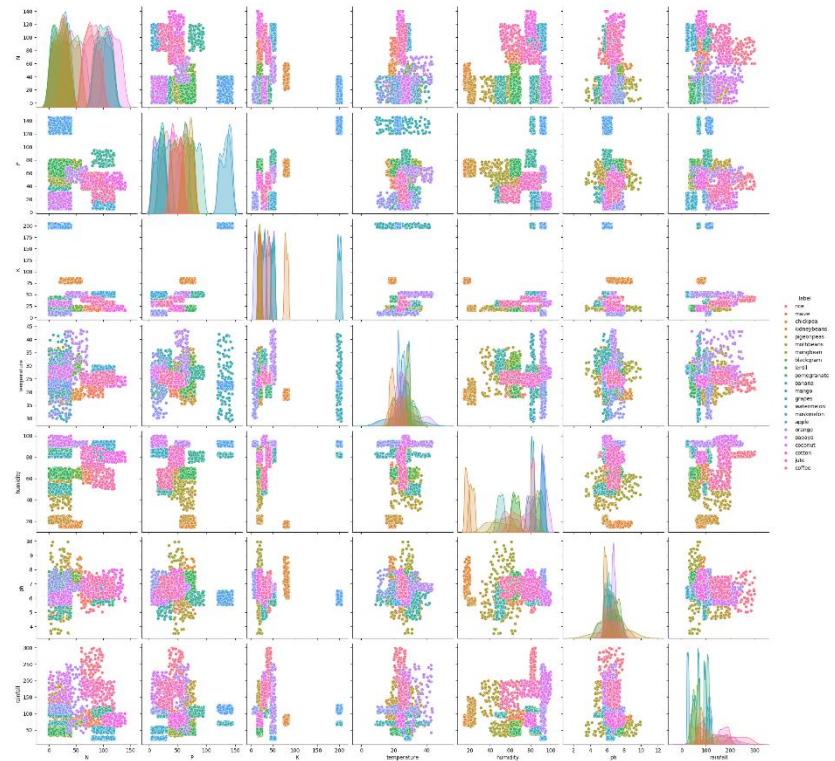


We also have plot `seaborn.pairplot()` :

To plot multiple pairwise bivariate distributions in a dataset, you can use the `.pairplot()` function.

```
sns.pairplot(crop_df, hue="label", size=5)
```

/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:2180: UserWarning: The "size" parameter has been renamed to "height"; please update your code.  
warnings.warn(msg, UserWarning)  
<seaborn.axisgrid.PairGrid at 0x7c094814c610>



Outliers and Anomalies

There is no Outliers in our project.

## Data Preprocessing Code Screenshots

Loading Data

#Loading the data

```
crop_df=pd.read_csv("/content/Crop_recommendation.csv")
crop_df
```

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice
...	...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	coffee

2200 rows x 8 columns

## Handling Missing Data

```
[ ] crop_df.shape
```

```
⇒ (2200, 8)
```

```
[ ] crop_df.info()
```

```
⇒ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  -
0   N                2200 non-null   int64
1   P                2200 non-null   int64
2   K                2200 non-null   int64
3   temperature      2200 non-null   float64
4   humidity         2200 non-null   float64
5   ph               2200 non-null   float64
6   rainfall         2200 non-null   float64
7   label            2200 non-null   object
dtypes: float64(4), int64(3), object(1)
memory usage: 137.6+ KB
```

```
▶ crop_df.isnull().sum()
```

```
⇒ N                0
   P                0
   K                0
   temperature      0
   humidity         0
   ph               0
   rainfall         0
   label            0
dtypes: int64
```

For checking the null values . isnull() function is used. To sum those null values we use. sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

## Data Transformation

```
-----
```

Feature Engineering	-----
Save Processed Data	-----