Course:- PGD in AI and ML Department:- SECE

Subject:- Machine Learning Algorithms MLA PBL Project

By Shantanu Sarwade - 1032233099

# Crop Recommendation

* Our aim is to train the model on the dataset and make it accurate to generalize well on

unseen new data provided.

* This will help in better crop recommendation to farmers and crop producers around the globe.

Data Description

This dataset contains information on the levels of nitrogen, phosphorus, and potassium in soil, as well as temperature, humidity, pH, and rainfall, and their impact on the growth of crops. The data can be used to make data-driven recommendations for achieving optimal nutrient and environmental conditions to improve crop yield.

1. Importing Libraries

import pandas as pd import numpy as np

import matplotlib.pyplot as plt import matplotlib.cm as cm import seaborn as sns

from sklearn.model\_selection import train\_test\_split

1. Reading Dataset

df = pd.read\_csv("Crop\_recommendation.csv") df.head()

Nitrogen phosphorus potassium temperature humidity ph \

0 90 42 43 20.879744 82.002744 6.502985

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
| 1 | 85 | 58 | 41 | 21.770462 | 80.319644 | 7.038096 |
|  |  |  |  |  |  |  |
| 2 | 60 | 55 | 44 | 23.004459 | 82.320763 | 7.840207 |
|  |  |  |  |  |  |  |
| 3 | 74 | 35 | 40 | 26.491096 | 80.158363 | 6.980401 |
|  |  |  |  |  |  |  |  |
|  | 4 | 78 | 42 | 42 | 20.130175 | 81.604873 | 7.628473 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | rainfall 202.935536 | label Unnamed: 8 Unnamed: 9 rice NaN NaN | | |
| 1 | 226.655537 | rice | NaN | NaN |
| 2 | 263.964248 | rice | NaN | NaN |
| 3 | 242.864034 | rice | NaN | NaN |
| 4 | 262.717340 | rice | NaN | NaN |

df.tail()

Nitrogen phosphorus potassium temperature humidity

ph \

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2195 | 107 | 34 | 32 | 26.774637 | 66.413269 |
| 6.780064 |  | | | | |
| 2196 | 99 | 15 | 27 | 27.417112 | 56.636362 |
| 6.086922 |  | | | | |
| 2197 | 118 | 33 | 30 | 24.131797 | 67.225123 |
| 6.362608 |  | | | | |
| 2198 | 117 | 32 | 34 | 26.272418 | 52.127394 |
| 6.758793 |  | | | | |
| 2199 | 104 | 18 | 30 | 23.603016 | 60.396475 |
| 6.779833 |  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | rainfall | label | Unnamed: 8 | Unnamed: 9 |
| 2195 | 177.774507 | coffee | NaN | NaN |
| 2196 | 127.924610 | coffee | NaN | NaN |
| 2197 | 173.322839 | coffee | NaN | NaN |
| 2198 | 127.175293 | coffee | NaN | NaN |
| 2199 | 140.937041 | coffee | NaN | NaN |

*# Dataset Summary or Information*

def data\_summary(data):

print("

")

print("The data has {} records and {} columns.".format(data.shape[0], data.shape[1]))

print("

")

print("The Information of data:", data.info())

print("

")

print("The data description:\n", data.describe())

print("

")

print("The Null Count check on data:\n", data.isnull().sum())

print("

") print(data\_summary(df))

The data has 2200 records and 10 columns.

<class 'pandas.core.frame.DataFrame'> RangeIndex: 2200 entries, 0 to 2199 Data columns (total 10 columns):

# Column Non-Null Count Dtype

* 1. Nitrogen 2200 non-null int64
  2. phosphorus 2200 non-null int64
  3. potassium 2200 non-null int64
  4. temperature 2200 non-null float64
  5. humidity 2200 non-null float64
  6. ph 2200 non-null float64
  7. rainfall 2200 non-null float64

7 label 2200 non-null object

1. Unnamed: 8 0 non-null float64
2. Unnamed: 9 0 non-null float64 dtypes: float64(6), int64(3), object(1) memory usage: 172.0+ KB

The Information of data: None

The data description:

Nitrogen phosphorus potassium temperature humidity \

count 2200.000000 2200.000000 2200.000000 2200.000000 2200.000000

mean 50.551818 53.362727 48.149091 25.616244 71.481779

std 36.917334 32.985883 50.647931 5.063749 22.263812

min 0.000000 5.000000 5.000000 8.825675 14.258040

25% 21.000000 28.000000 20.000000 22.769375 60.261953

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| 50% | 37.000000 | 51.000000 | 32.000000 | 25.598693 | 80.473146 |
|  |  |  |  |  |  |
| 75% | 84.250000 | 68.000000 | 49.000000 | 28.561654 | 89.948771 |
|  |  |  |  |  |  |  |
|  | max | 140.000000 | 145.000000 | 205.000000 | 43.675493 | 99.981876 |

|  |  |
| --- | --- |
| ph rainfall Unnamed: 8 Unnamed: 9  count 2200.000000 2200.000000 0.0 0.0  mean 6.469480 103.463655 NaN NaN  std 0.773938 54.958389 NaN NaN  min 3.504752 20.211267 NaN NaN  25% 5.971693 64.551686 NaN NaN  50% 6.425045 94.867624 NaN NaN  75% 6.923643 124.267508 NaN NaN  max 9.935091 298.560117 NaN NaN | |
|  |  |
| The Null Count check on data: Nitrogen 0  phosphorus 0  potassium 0  temperature 0  humidity 0  ph 0  rainfall 0  label 0  Unnamed: 8 2200  Unnamed: 9 2200  dtype: int64 | |
|  |  |
| None |

Observation

* From above observation we have last two columns which are having null values. So we can remove those columns as they are not needed.

df.columns

Index(['Nitrogen', 'phosphorus', 'potassium', 'temperature', 'humidity', 'ph',

'rainfall', 'label', 'Unnamed: 8', 'Unnamed: 9'],

dtype='object')

df.drop(columns=['Unnamed: 8', 'Unnamed: 9'], axis=1, inplace=True) df

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | |  | |  |  |  |
| Ni | trogen phosphorus po | | | tassium | | temperature | humidity |
| ph \ |  |  | |  | |  |  |
| 0 | 90 | 42 | | 43 | | 20.879744 | 82.002744 |
| 6.502985 |  |  | |  | |  |  |
| 1 | 85 | 58 | | 41 | | 21.770462 | 80.319644 |
| 7.038096 |  |  | |  | |  |  |
| 2 | 60 | 55 | | 44 | | 23.004459 | 82.320763 |
| 7.840207 |  |  | |  | |  |  |
| 3 | 74 | 35 | | 40 | | 26.491096 | 80.158363 |
| 6.980401 |  |  | |  | |  |  |
| 4 | 78 | 42 | | 42 | | 20.130175 | 81.604873 |
| 7.628473 |  |  | |  | |  |  |
| ... | ... | ... | | ... | | ... | ... | .. |
|  | . |  |  | |  | |  |  |  |
| 2195 | 107 | | 34 | | 32 | 26.774637 | 66.413269 |
| 6.780064 |  | | | | | | |
| 2196 | 99 | | 15 | | 27 | 27.417112 | 56.636362 |
| 6.086922 |  | | | | | | |
| 2197 | 118 | | 33 | | 30 | 24.131797 | 67.225123 |
| 6.362608 |  | | | | | | |
| 2198 | 117 | | 32 | | 34 | 26.272418 | 52.127394 |
| 6.758793 |  | | | | | | |
| 2199 | 104 | | 18 | | 30 | 23.603016 | 60.396475 |
| 6.779833 |  | | | | | | |

|  |  |  |
| --- | --- | --- |
|  | rainfall | label |
| 0 | 202.935536 | rice |
| 1 | 226.655537 | rice |
| 2 | 263.964248 | rice |
| 3 | 242.864034 | rice |
| 4 | 262.717340 | rice |
| ... | ... | ... |
| 2195 | 177.774507 | coffee |
| 2196 | 127.924610 | coffee |
| 2197 | 173.322839 | coffee |
| 2198 | 127.175293 | coffee |
| 2199 | 140.937041 | coffee |

[2200 rows x 8 columns] df.columns

Index(['Nitrogen', 'phosphorus', 'potassium', 'temperature', 'humidity', 'ph',

'rainfall', 'label'],

dtype='object')

Observation:

* We have our target column: 'label'
* We have ourr feature columns: 'Nitrogen', 'phosphorus', 'potassium', 'temperature', 'humidity', 'ph', 'rainfall'

Data Visualization

df.columns

Index(['Nitrogen', 'phosphorus', 'potassium', 'temperature', 'humidity', 'ph',

'rainfall', 'label'],

dtype='object')

labels = list(df['label'].value\_counts().index)

# Univariate Analysis

*# Extracting Nitrogen content from the dataset*

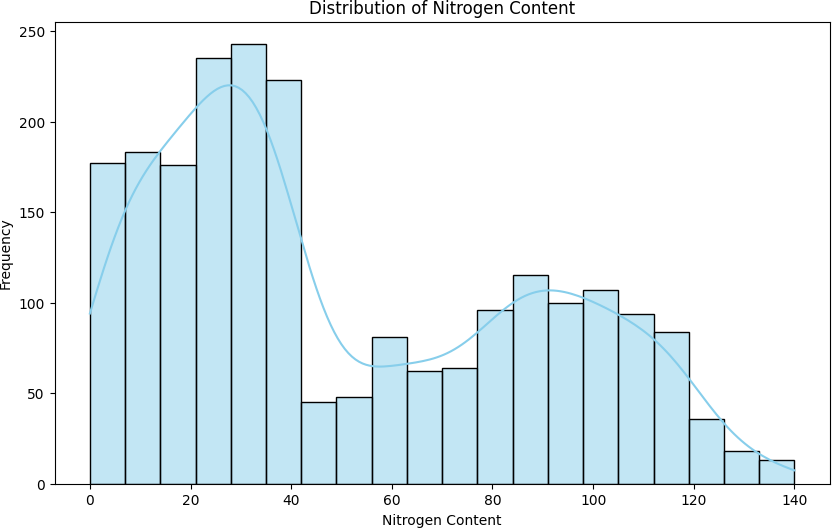
nitrogen\_data = df['Nitrogen']

*# Plotting the histogram*

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

sns.histplot(nitrogen\_data, bins=20, kde=True, color='skyblue') plt.title('Distribution of Nitrogen Content') plt.xlabel('Nitrogen Content')

plt.ylabel('Frequency') plt.show()



Observations

* The highest crop yields are associated with nitrogen content levels between 20 and 60. Beyond a nitrogen content of 60, crop yields do not increase significantly.
* There is a significant amount of variation in crop yields for each given level of nitrogen content.
* This variation is likely due to a number of factors, such as phosphorus and potassium levels, temperature, humidity, pH, and rainfall.

*# Extracting phosphorus content from the dataset*

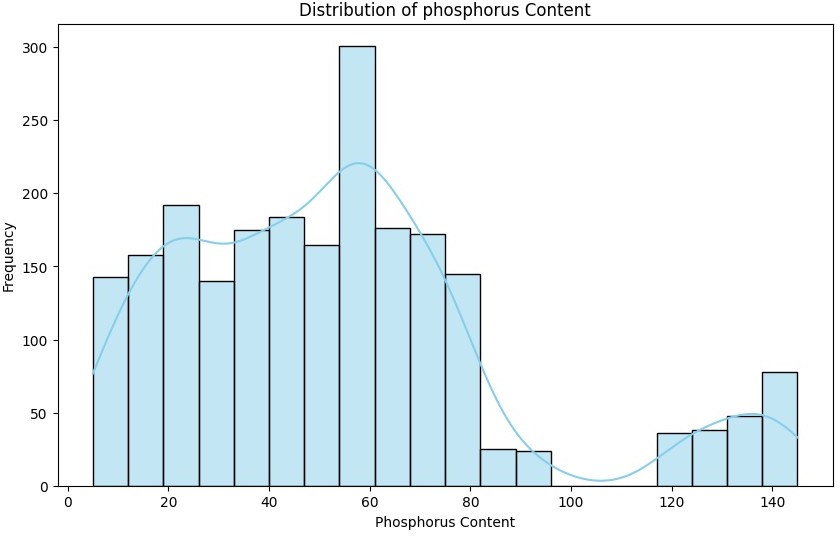
phosphorus\_data = df['phosphorus']

*# Plotting the histogram*

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

sns.histplot(phosphorus\_data, bins=20, kde=True, color='skyblue') plt.title('Distribution of phosphorus Content') plt.xlabel('Phosphorus Content')

plt.ylabel('Frequency') plt.show()



Observations

* The highest crop yields are associated with phosphorus content levels between 10 and 30.
* Beyond a phosphorus content of 30, crop yields do not increase significantly.
* There is a significant amount of variation in crop yields for each given level of phosphorus content.
* This variation is likely due to a number of factors, such as nitrogen and potassium levels, temperature, humidity, pH, and rainfall.

*# Extracting Potassium content from the dataset*

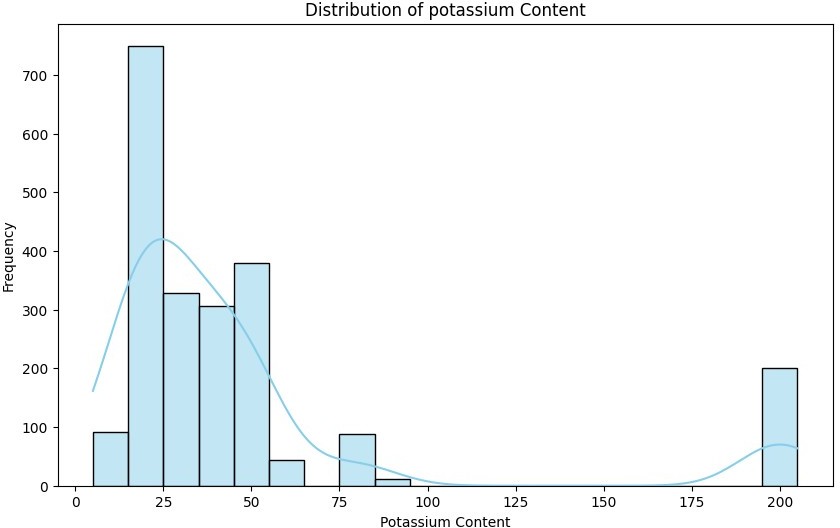
potassium\_data = df['potassium']

*# Plotting the histogram*

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

sns.histplot(potassium\_data, bins=20, kde=True, color='skyblue') plt.title('Distribution of potassium Content') plt.xlabel('Potassium Content')

plt.ylabel('Frequency') plt.show()



Observation

* The highest crop yields are associated with potassium content levels between 20 and 60.
* Beyond a potassium content of 60, crop yields do not increase significantly.
* There is a significant amount of variation in crop yields for each given level of potassium content.

*# Extracting Temperature data from the dataset*

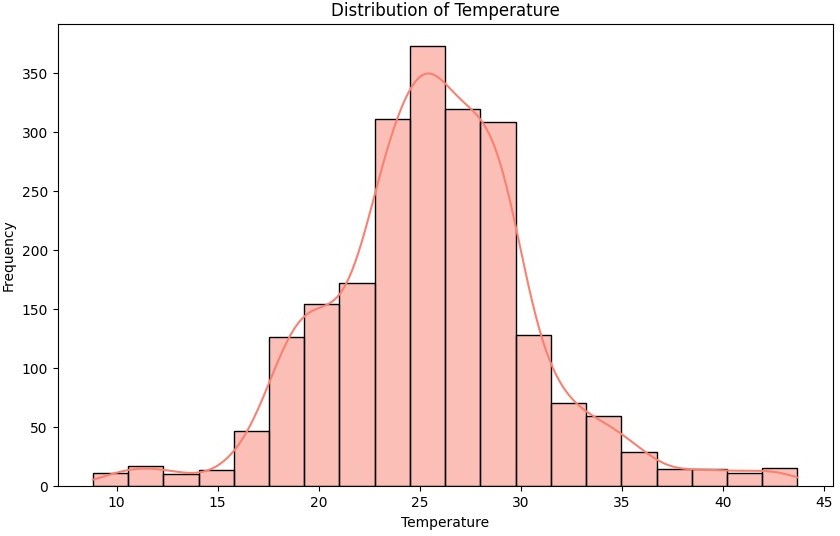
temperature\_data = df['temperature']

*# Plotting the histogram*

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

sns.histplot(temperature\_data, bins=20, kde=True, color='salmon') plt.title('Distribution of Temperature') plt.xlabel('Temperature')

plt.ylabel('Frequency') plt.show()



Observations

* The highest crop yields are associated with temperatures within the optimal temperature range for each crop.
* Outside of the optimal temperature range, crop yields decline.
* The optimal temperature range varies depending on the crop.

*# Extracting Humidity data from the dataset*

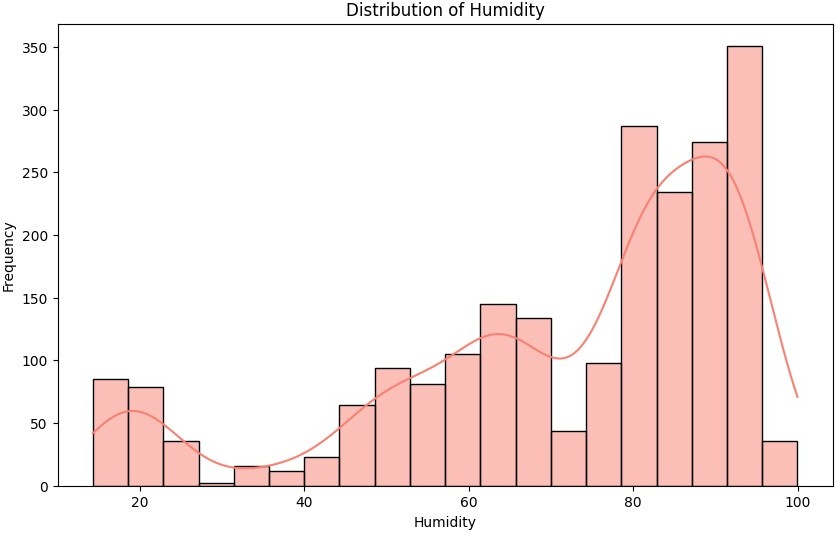
humidity\_data = df['humidity']

*# Plotting the histogram*

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

sns.histplot(humidity\_data, bins=20, kde=True, color='salmon') plt.title('Distribution of Humidity')

plt.xlabel('Humidity') plt.ylabel('Frequency') plt.show()



*# Extracting ph data from the dataset*

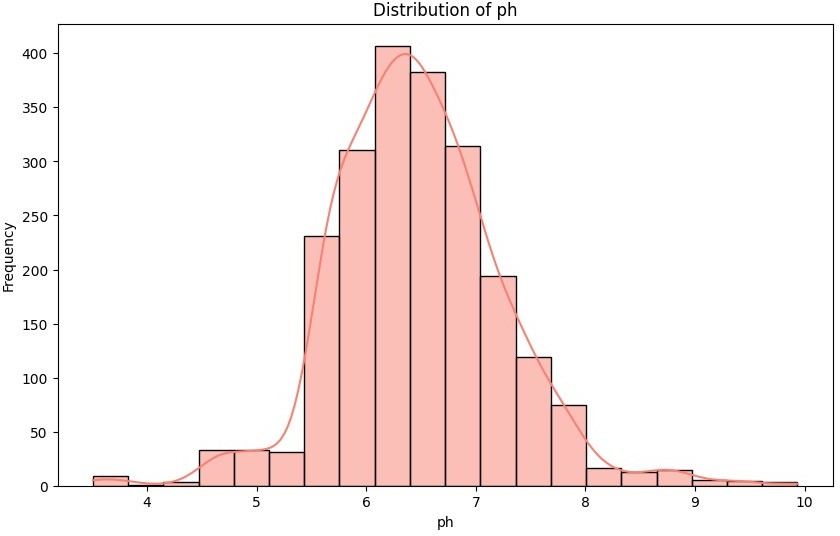
ph\_data = df['ph']

*# Plotting the histogram*

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

sns.histplot(ph\_data, bins=20, kde=True, color='salmon') plt.title('Distribution of ph')

plt.xlabel('ph') plt.ylabel('Frequency') plt.show()



*# Extracting ph data from the dataset*

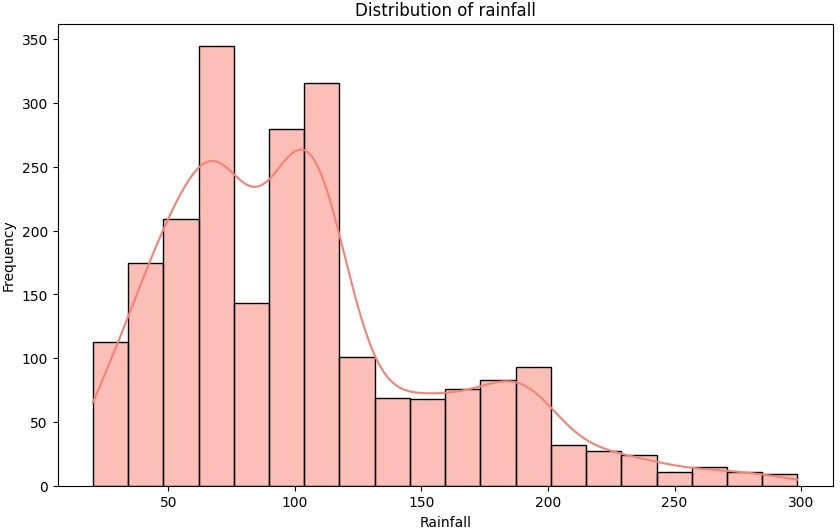
rainfall\_data = df['rainfall']

*# Plotting the histogram*

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

sns.histplot(rainfall\_data, bins=20, kde=True, color='salmon') plt.title('Distribution of rainfall')

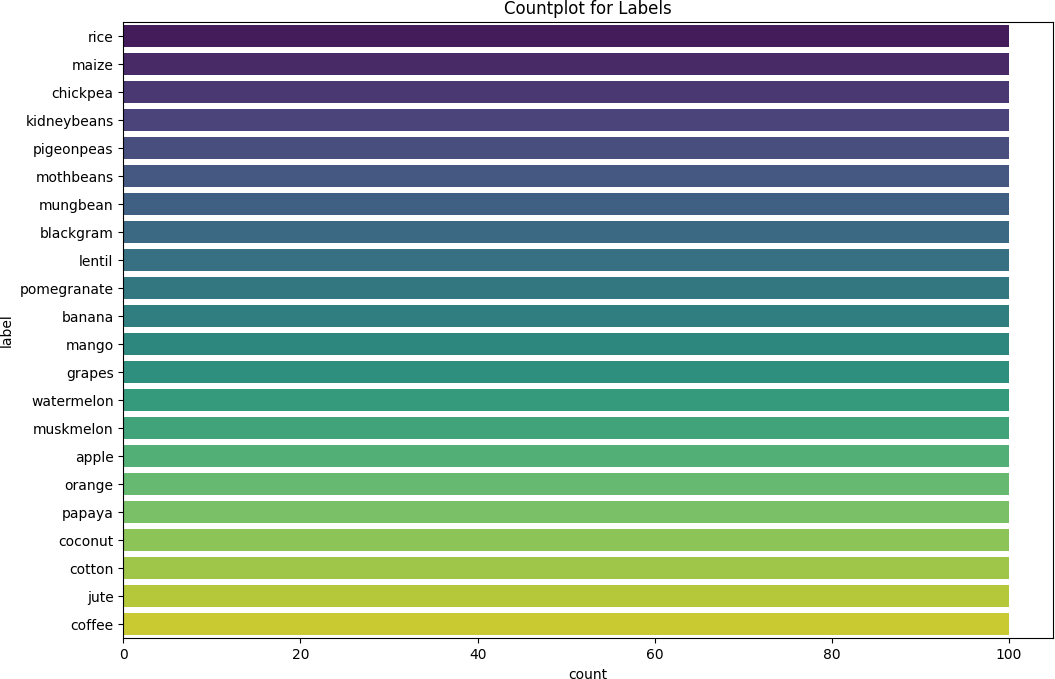
plt.xlabel('Rainfall') plt.ylabel('Frequency') plt.show()



plt.figure(figsize=[12,8])

sns.countplot(data=df, y='label', palette = "viridis") plt.title("Countplot for Labels")

plt.show()



df.describe()

Nitrogen phosphorus potassium temperature humidity

\

count 2200.000000 2200.000000 2200.000000 2200.000000 2200.000000

mean 50.551818 53.362727 48.149091 25.616244 71.481779

std 36.917334 32.985883 50.647931 5.063749 22.263812

min 0.000000 5.000000 5.000000 8.825675 14.258040

25% 21.000000 28.000000 20.000000 22.769375 60.261953

50% 37.000000 51.000000 32.000000 25.598693 80.473146

75% 84.250000 68.000000 49.000000 28.561654 89.948771

max 140.000000 145.000000 205.000000 43.675493 99.981876

|  |  |  |
| --- | --- | --- |
|  | ph | rainfall |
| count | 2200.000000 | 2200.000000 |
| mean | 6.469480 | 103.463655 |
| std | 0.773938 | 54.958389 |
| min | 3.504752 | 20.211267 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 25% | 5.971693 | 64.551686 |
| 50% | 6.425045 | 94.867624 |
| 75% | 6.923643 | 124.267508 |
|  | max | 9.935091 | 298.560117 |

# Bi-variate Analysis

fig, ax = plt.subplots(figsize=(16, 8))

sns.scatterplot(x = "humidity", y = "temperature", hue = "label", data

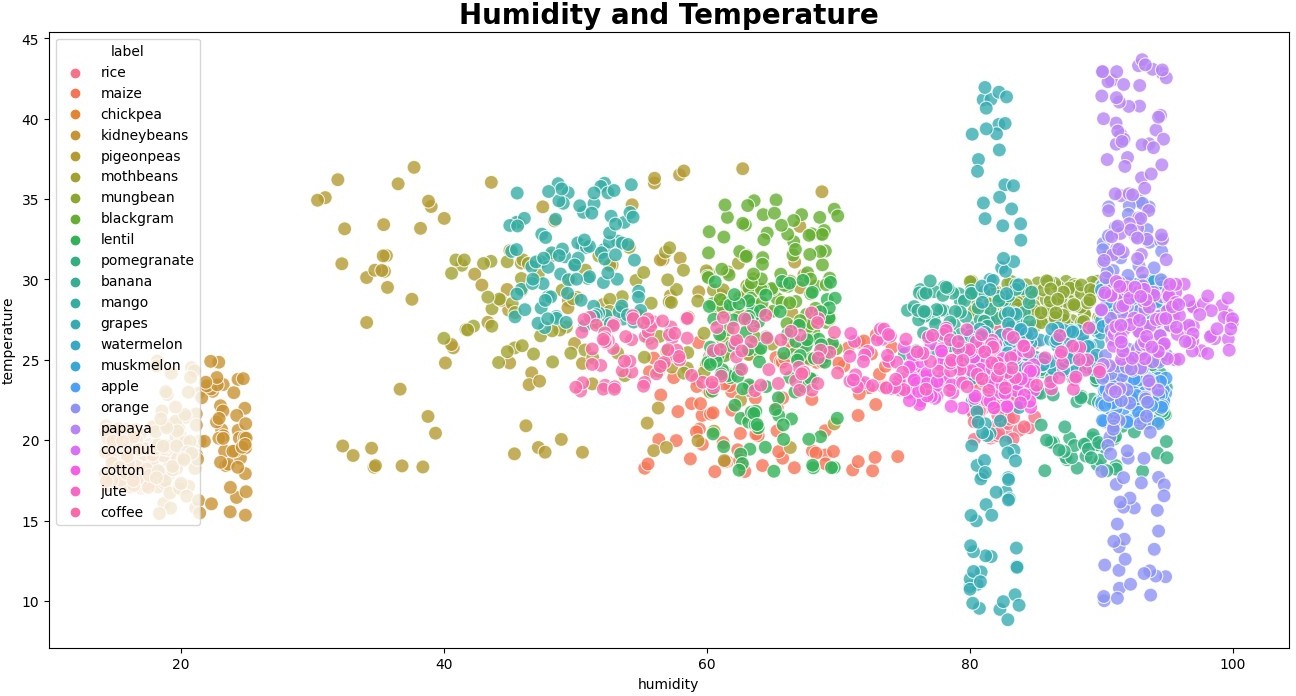
= df, s = 100, alpha = 0.8)

ax.set\_title("Humidity and Temperature", fontsize=20, fontweight = 'bold')

*# ax.legend(title = "Crop", fontsize = 14, loc = 'upper left', frameon*

*= True, edgecolor = 'black', shadow = True)*

plt.show()



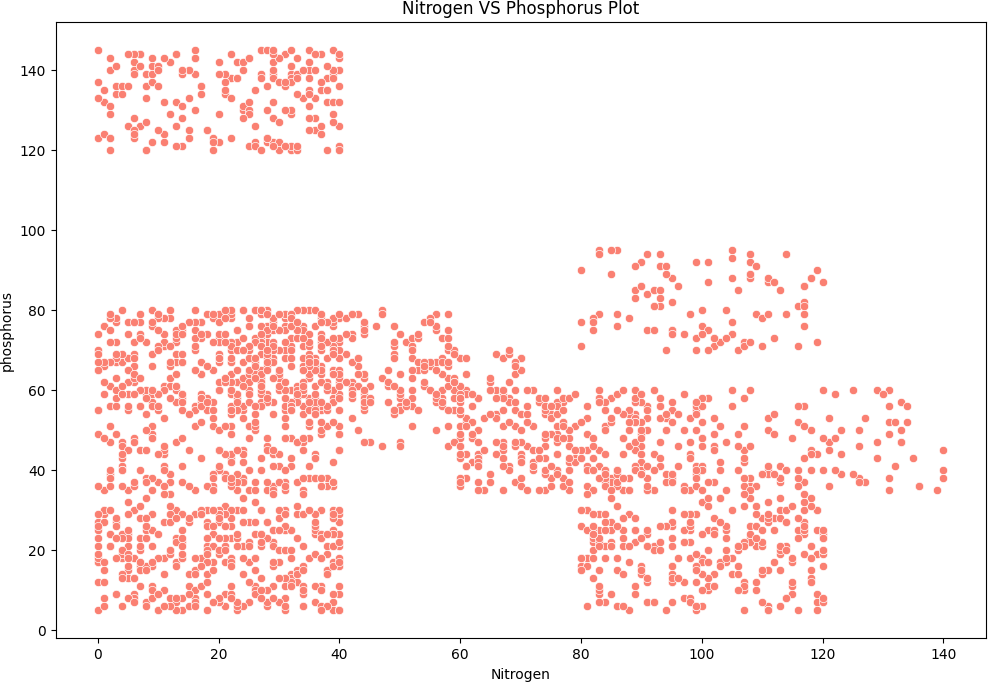
Observations

* Nitrogen is an important factor influencing maize yield.
* There is a point of diminishing returns for nitrogen fertilization. Applying too much nitrogen can actually be harmful to maize plants.
* Other factors, such as phosphorus and potassium levels, temperature, humidity, pH, and rainfall, also play a role in maize yield.

plt.figure(figsize=[12,8])

sns.scatterplot(data=df, x="Nitrogen", y="phosphorus", color='salmon') plt.title("Nitrogen VS Phosphorus Plot")

plt.show()



Observations

* Some crops, such as potatoes and rye, are more tolerant of a wide range of pH levels than other crops, such as maize and wheat. This means that potato and rye crops may be able to produce higher yields in a wider range of soil conditions.
* The impact of pH stress on crop yields can be mitigated by using liming and other practices to adjust the pH of the soil.
* Farmers should monitor their crops closely and adjust their management practices as needed to minimize the impact of pH stress.

*# Extracting Rainfall and Crop Yield data from the dataset*

rainfall\_data = df['rainfall'] crop\_yield\_data = df['label']

*# Plotting the scatter plot*

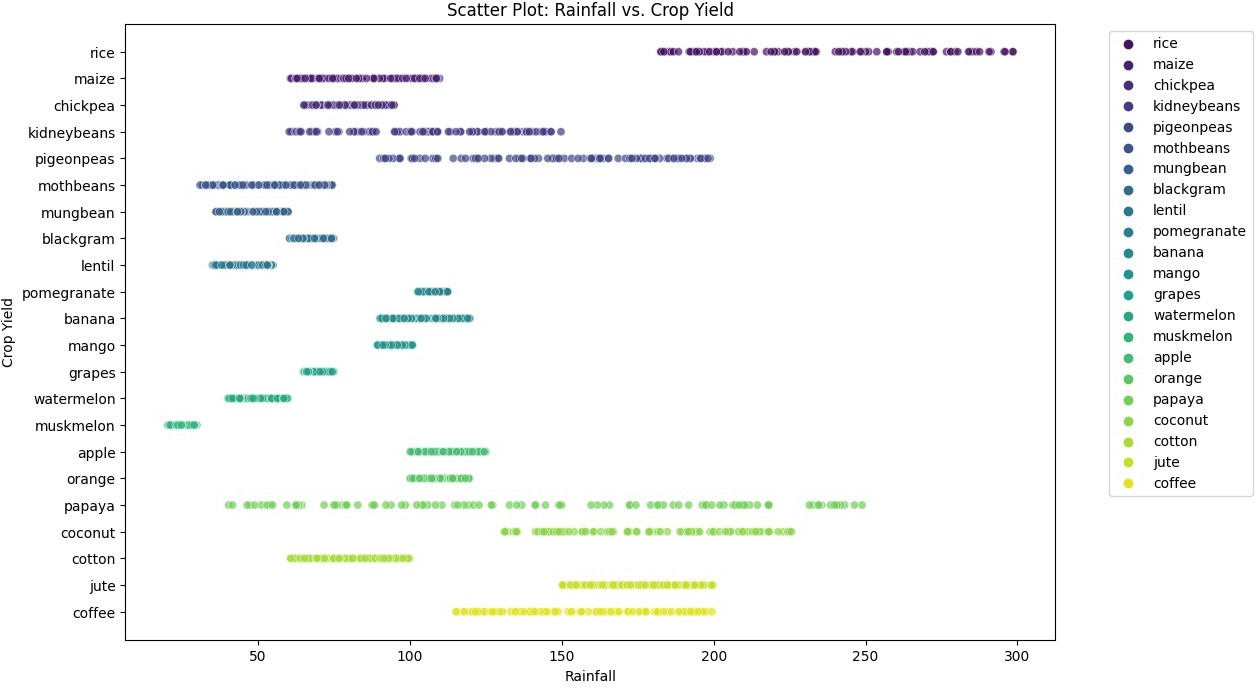
plt.figure(figsize=(12, 8)) sns.scatterplot(x=rainfall\_data, y=crop\_yield\_data, hue=crop\_yield\_data, palette='viridis', alpha=0.7)

plt.title('Scatter Plot: Rainfall vs. Crop Yield') plt.xlabel('Rainfall')

plt.ylabel('Crop Yield')

plt.legend(bbox\_to\_anchor=(1.05, 1), loc='upper left') *# To display legend outside the plot*

plt.show()



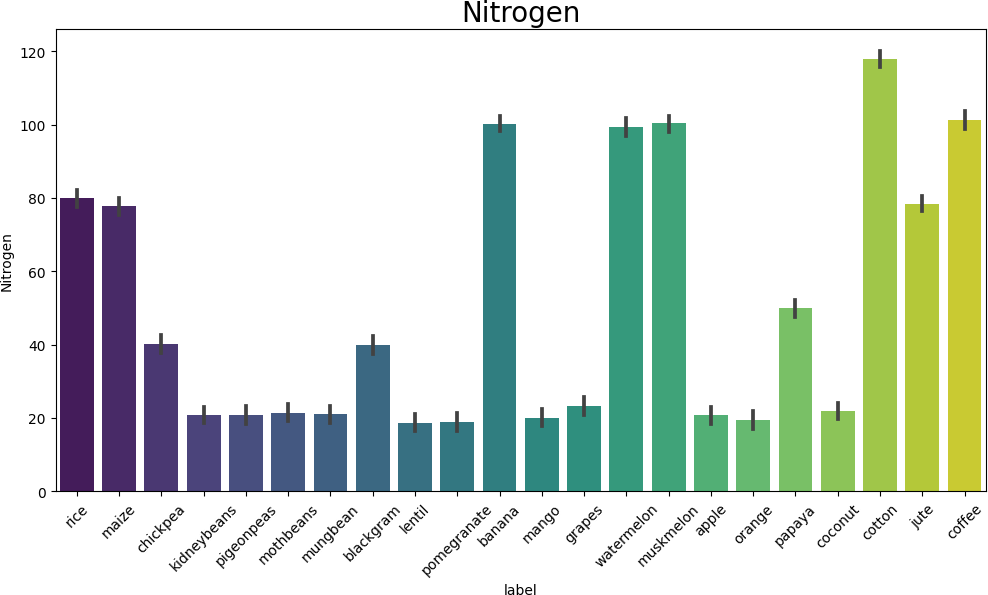
Obsevation

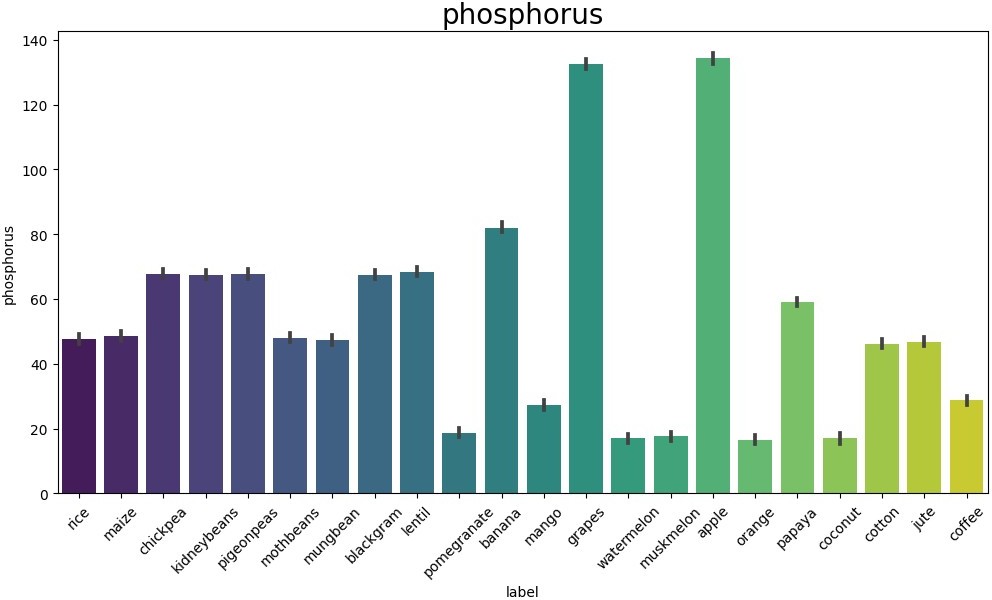
* Some crops, such as potatoes and maize, are more responsive to potassium than other crops, such as wheat and rice. This means that potato and maize crops may require higher potassium content levels to achieve optimal yields.
* The optimal potassium content level may vary depending on the climate and soil conditions. For example, crops grown in sandy soils may require higher potassium content levels than crops grown in clay soils.
* Farmers should monitor their crops closely and adjust their potassium fertilizer applications as needed.
* Too much potassium can actually be harmful to crops, so it is important to avoid over- fertilization.

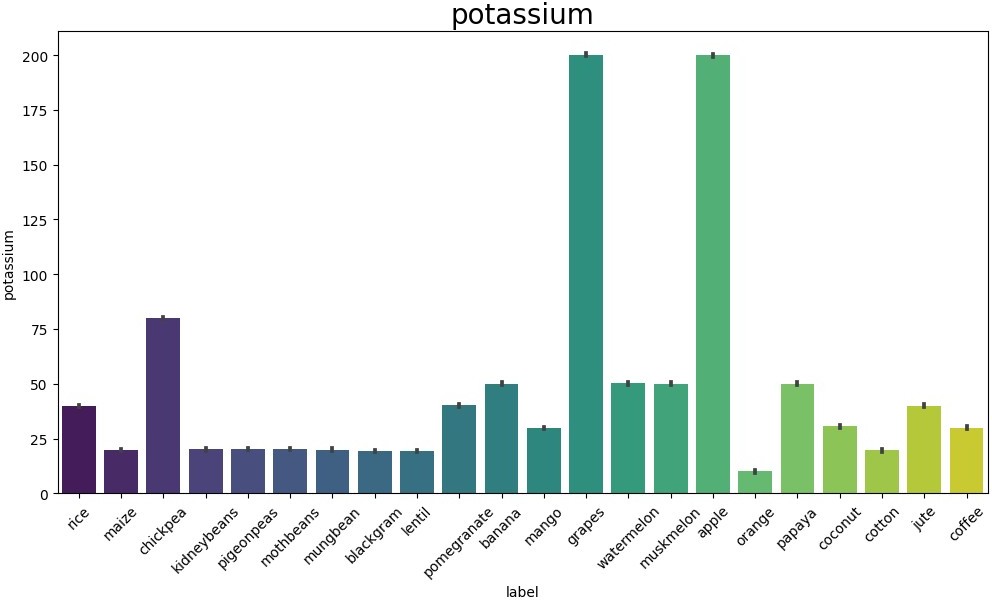
columns = df.select\_dtypes(include = ['float64', 'int64']).columns

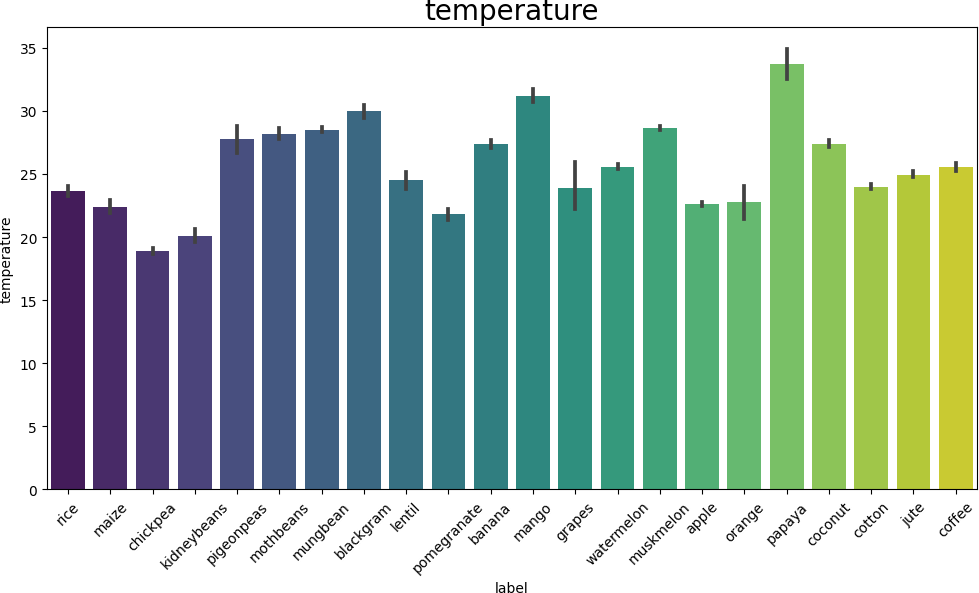
for col in columns: plt.figure(figsize = (12,6)) plt.title(col, fontsize = 20)

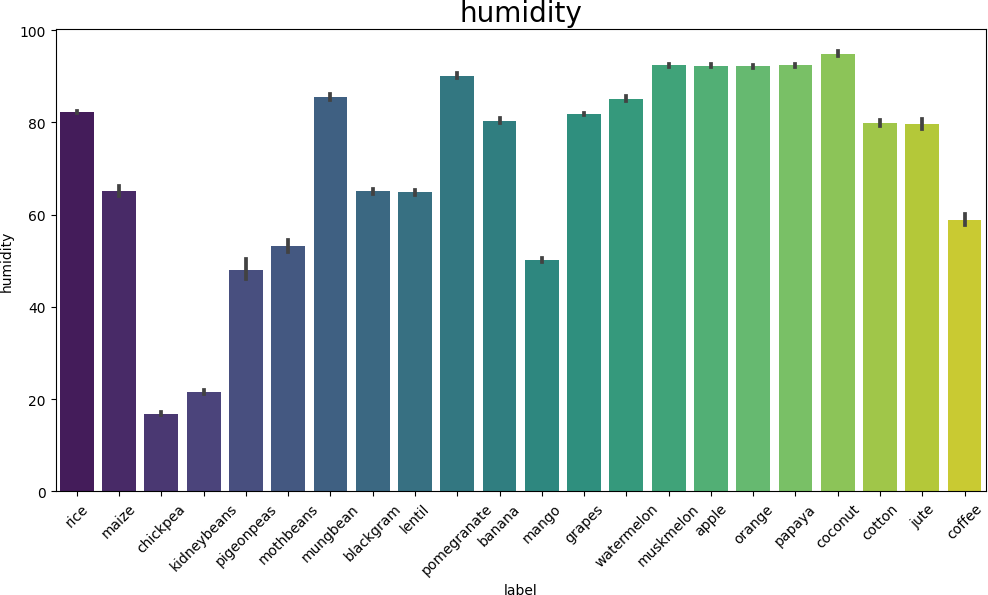
sns.barplot(x = 'label', y = col, palette = 'viridis', data = df) plt.xticks(rotation = 45)

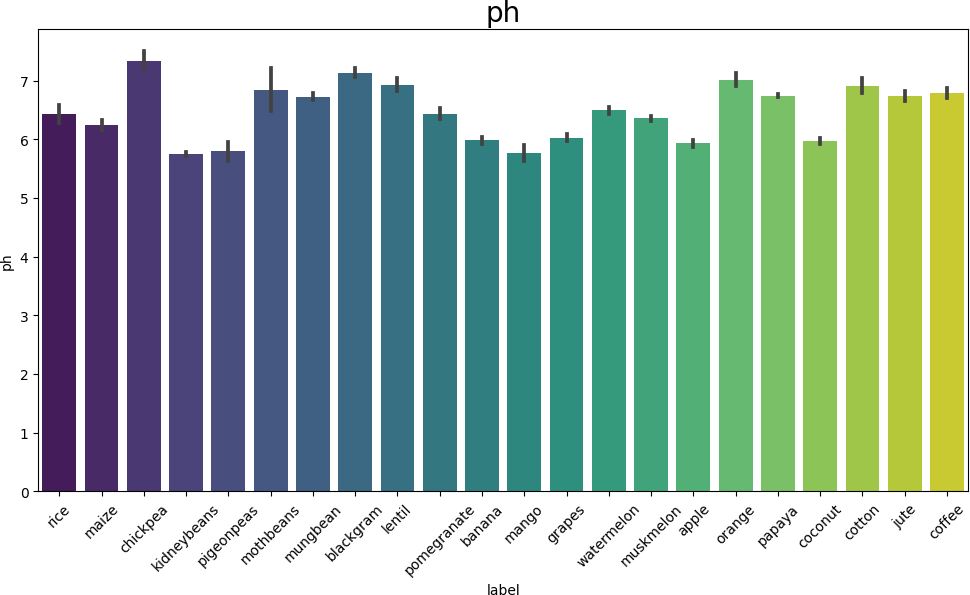


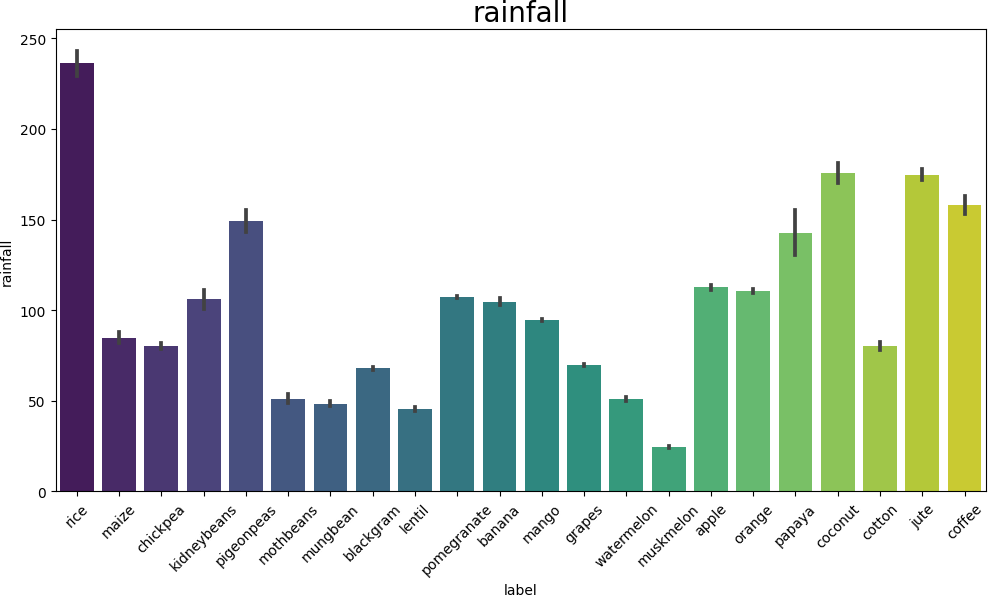












labels = list(df['label'].unique())

for label in labels:

new\_df = df[df['label']==label]

colors = cm.viridis\_r([0.3, 0.5, 0.8])

fig, ax = plt.subplots(figsize = (8, 8))

nutrients = ['Nitrogen', 'phosphorus', 'potassium']

sizes = [new\_df['Nitrogen'].mean(), new\_df['phosphorus'].mean(), new\_df['potassium'].mean()]

explode = [0.05, 0.05, 0.05]

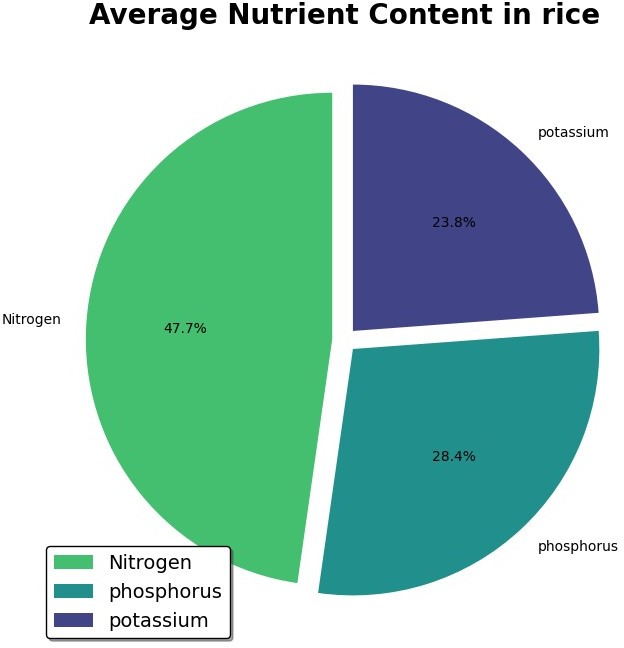
ax.pie(sizes, labels = nutrients, colors = colors, autopct = '%1.1f%

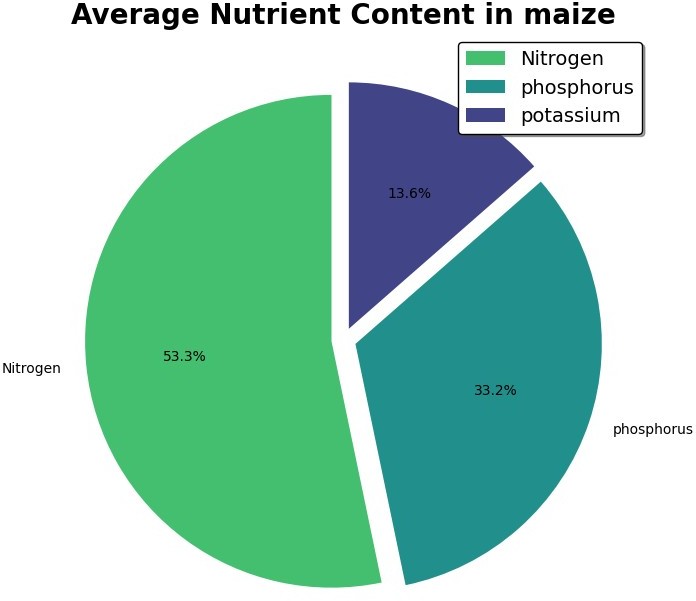
%', startangle = 90, explode = explode)

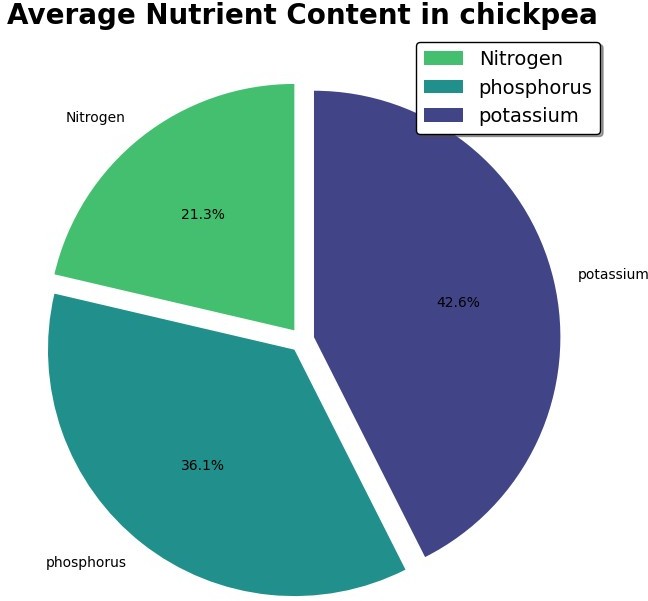
ax.set\_title(f'Average Nutrient Content in {label}', fontsize = 20, fontweight = 'bold')

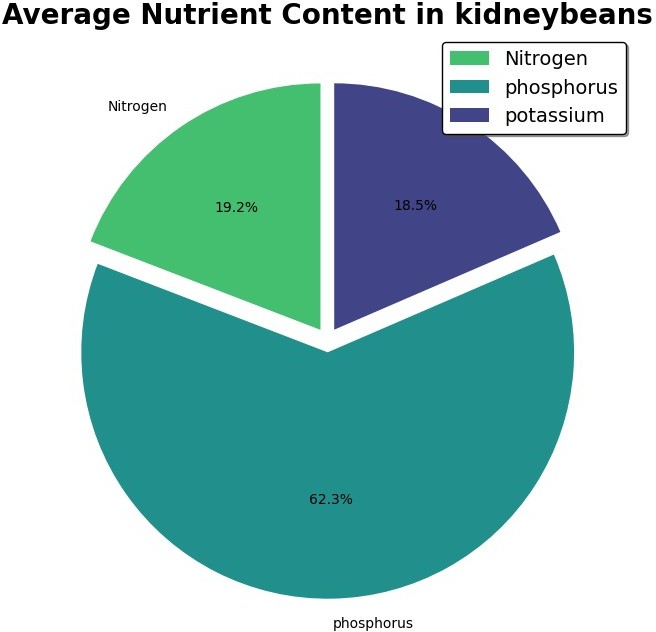
ax.legend(fontsize = 14, loc = 'best', frameon = True, edgecolor = 'black', shadow = True)

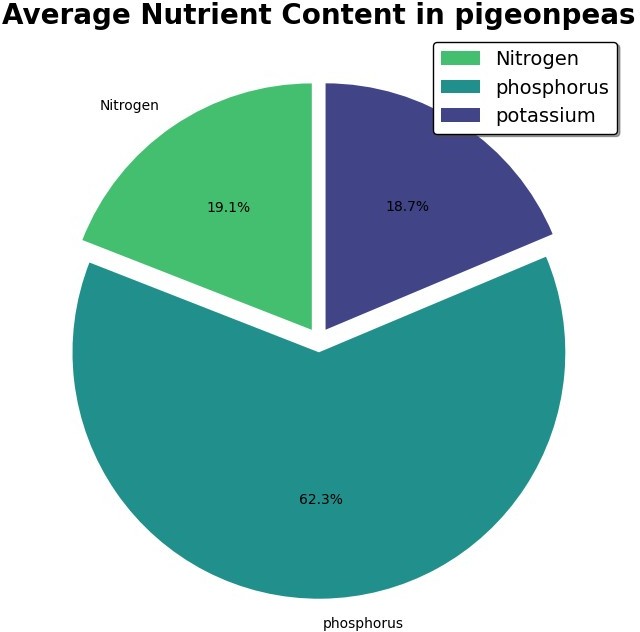
plt.show()

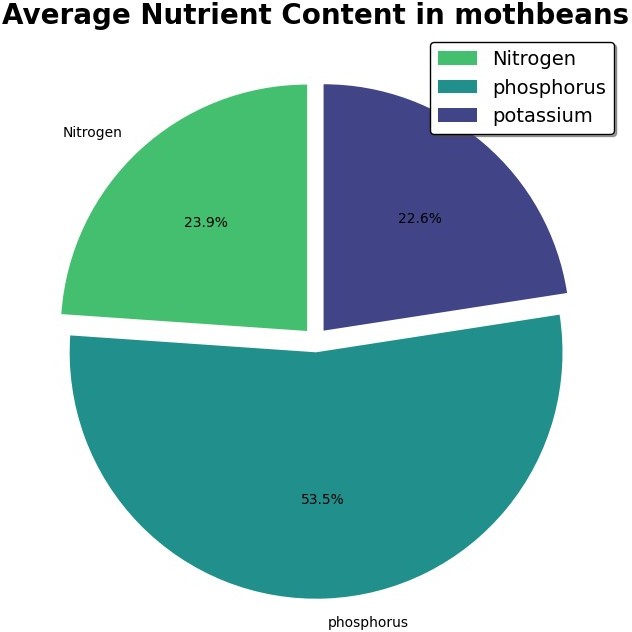


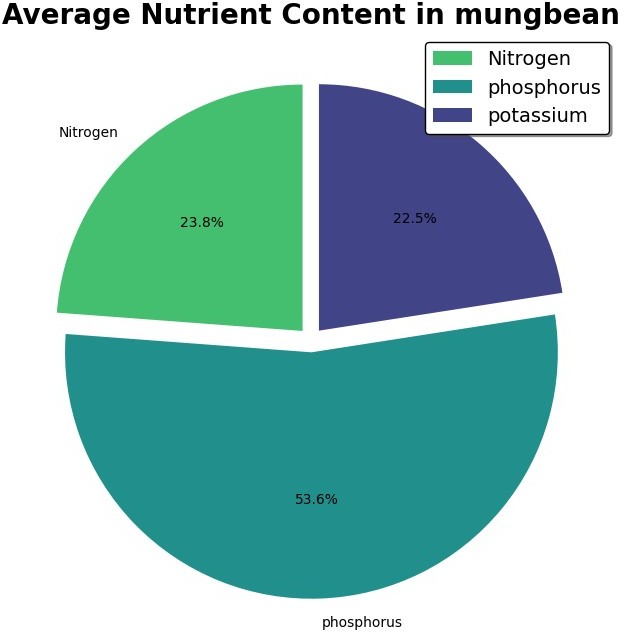


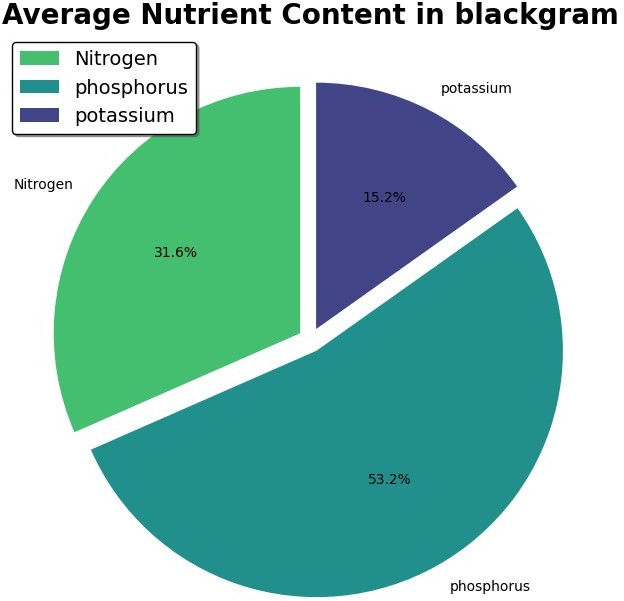


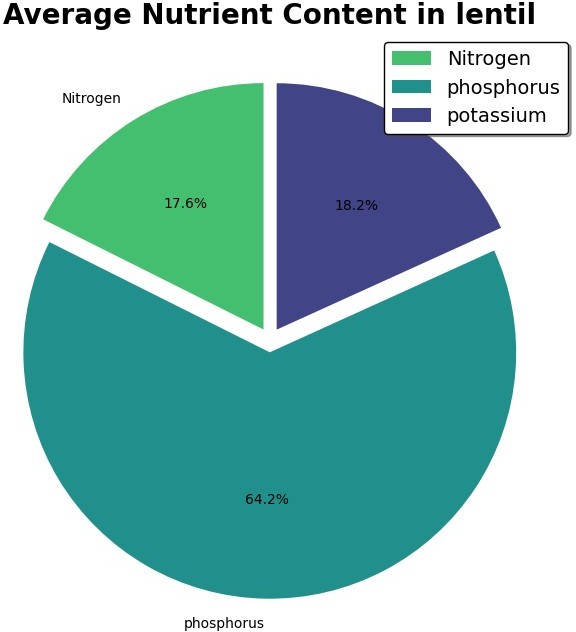


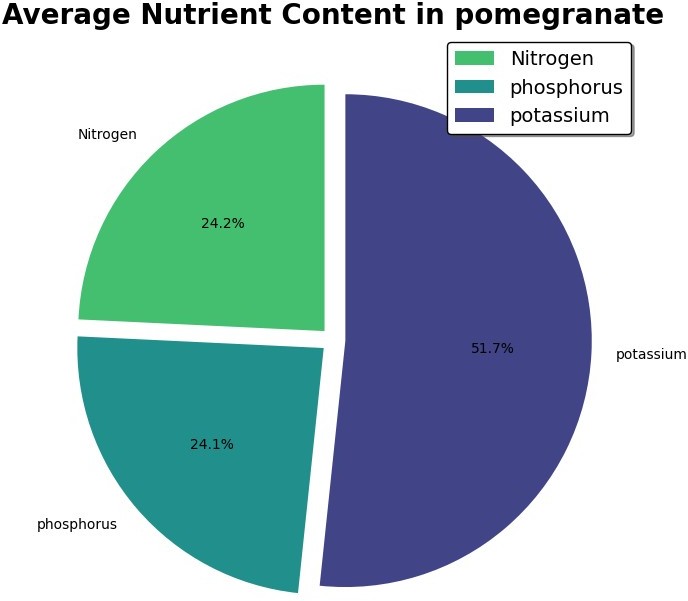


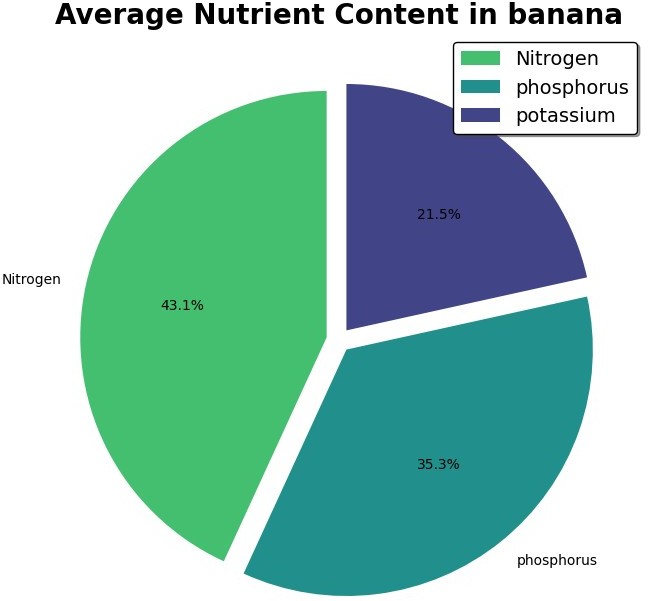


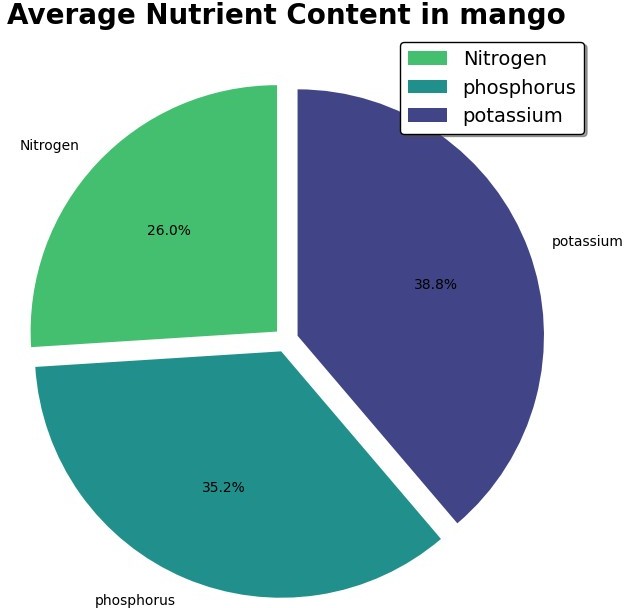


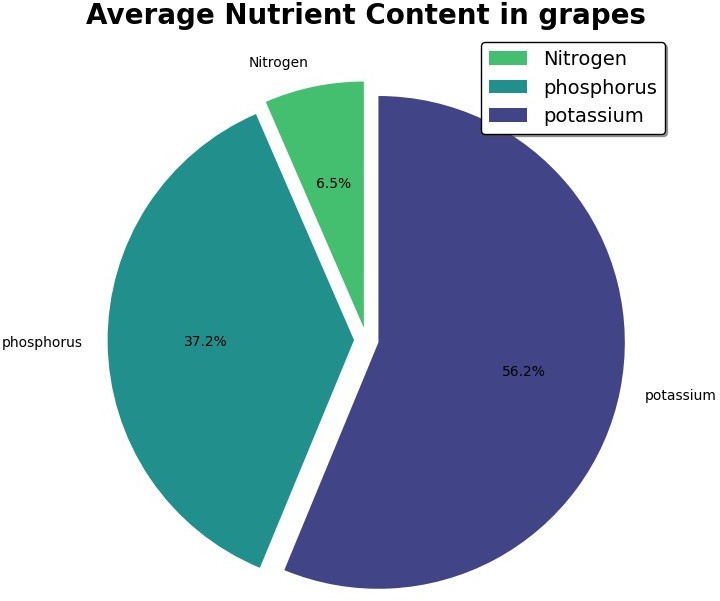


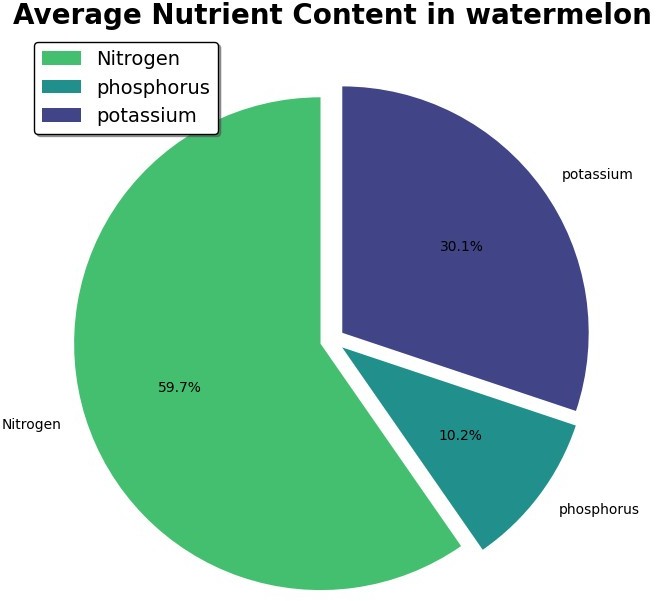


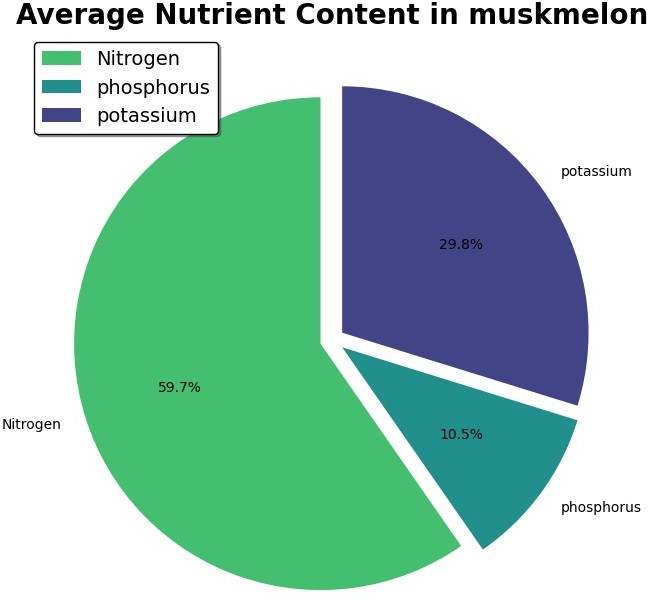


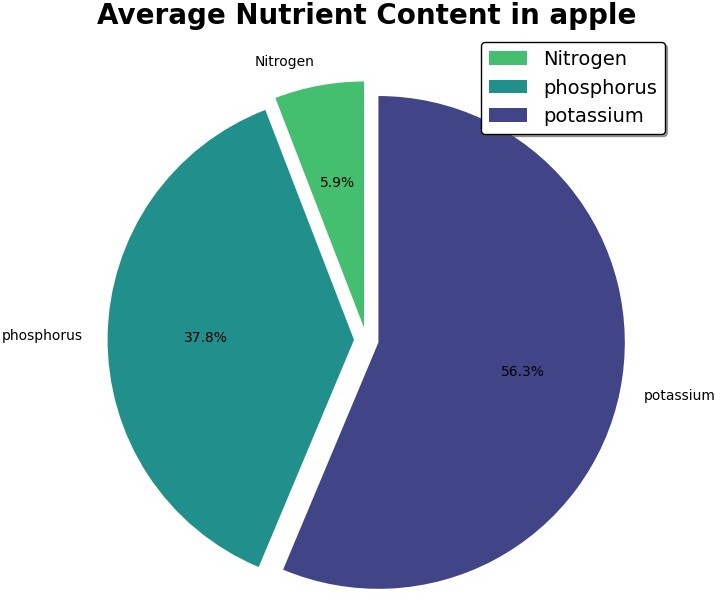


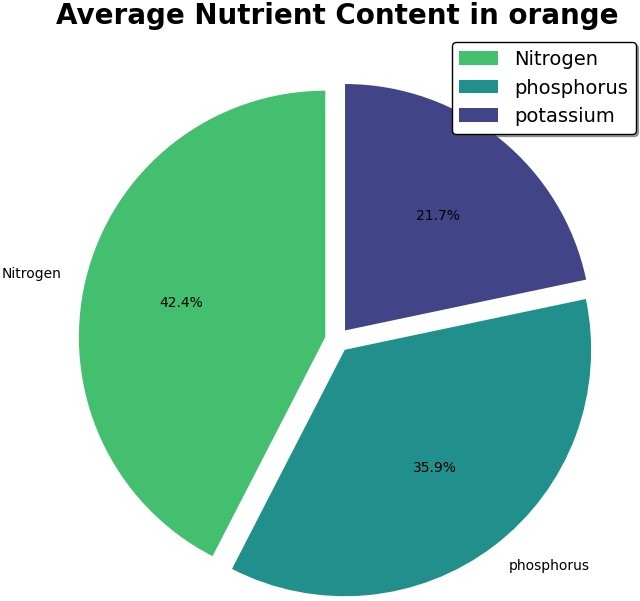


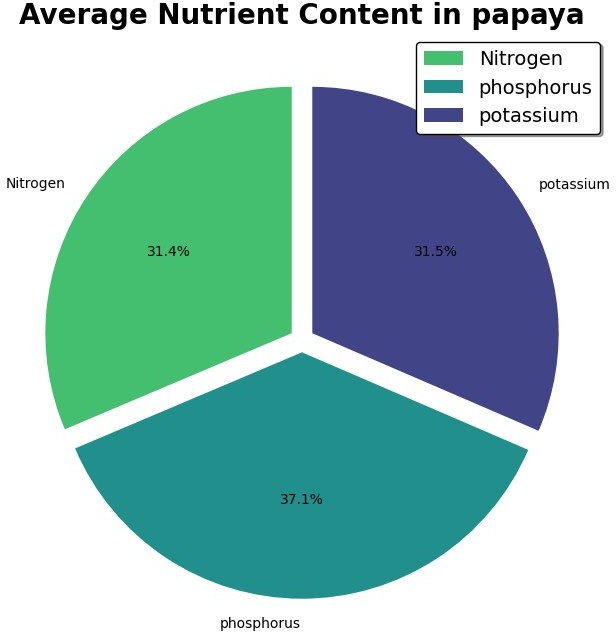


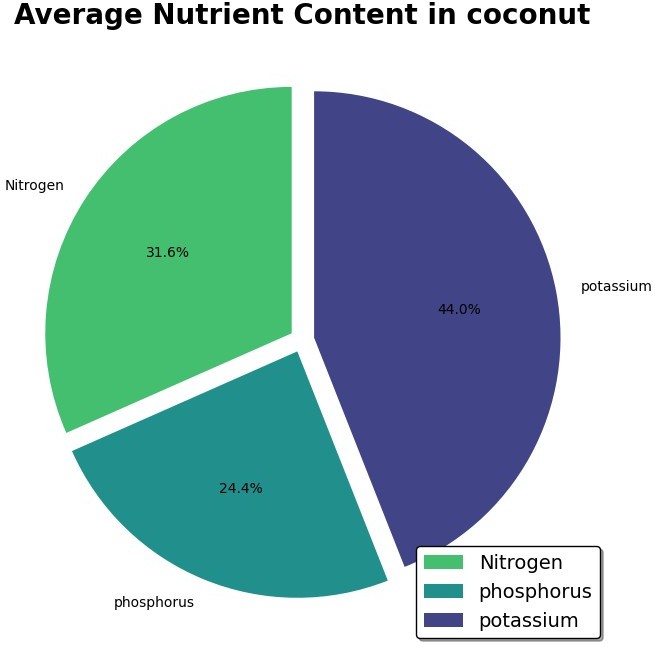


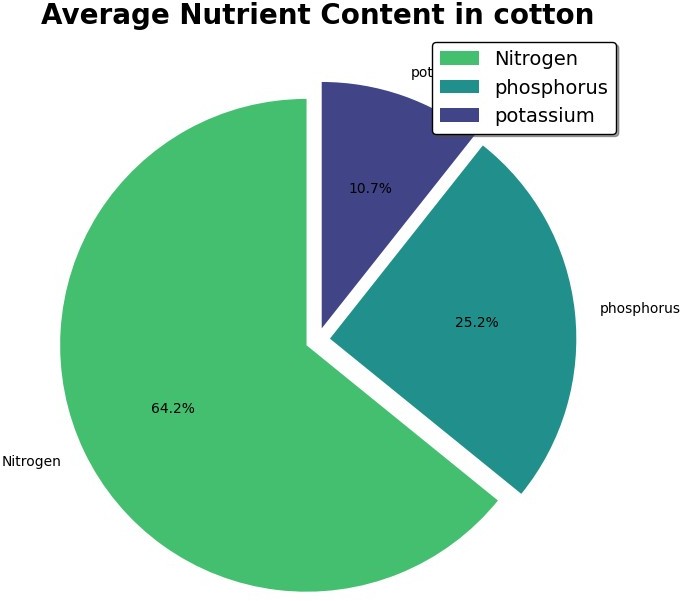


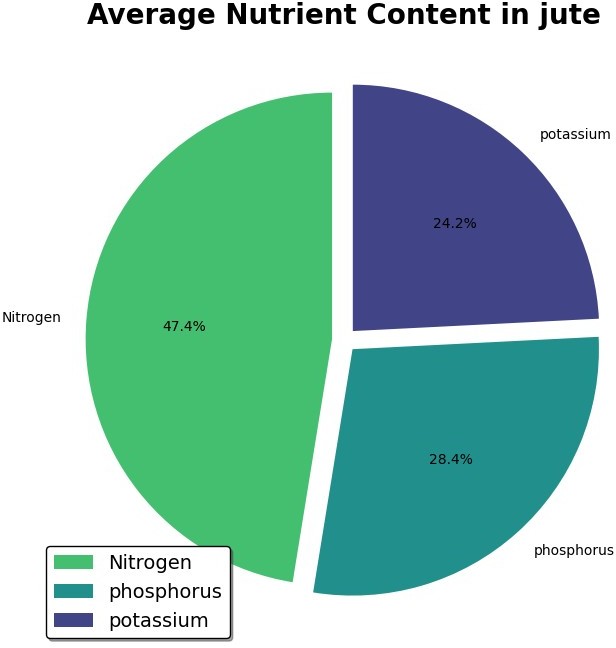


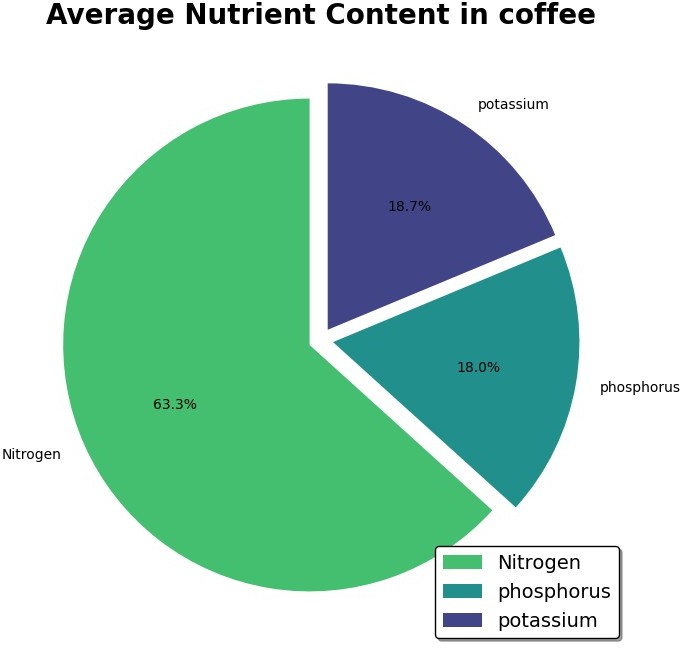












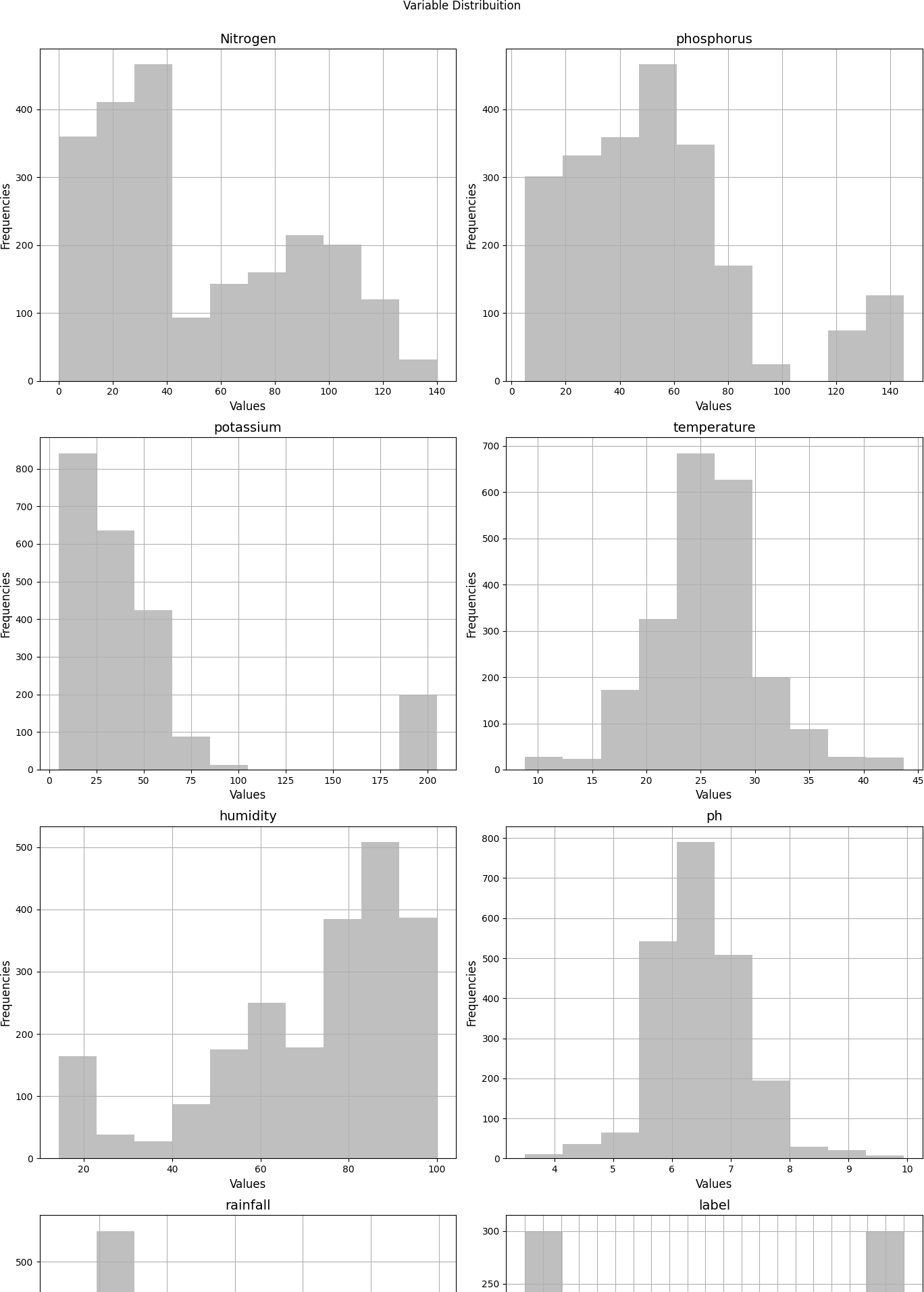
list\_columns = list(df.columns)

fig, axs = plt.subplots(4, 2, figsize = (14, 24)) fig.suptitle('Variable Distribuition')

for i, column in enumerate(list\_columns): ax = axs.flat[i]

ax.hist(df[column], color = 'grey', alpha = 0.5) ax.set\_title(column, fontsize = 14) ax.set\_xlabel('Values', fontsize = 12) ax.set\_ylabel('Frequencies', fontsize = 12) ax.grid(True)

plt.tight\_layout() plt.subplots\_adjust(top = 0.95)



corr\_matrix = df.corr()

fig, ax = plt.subplots(figsize = (12, 10))

heatmap = sns.heatmap(corr\_matrix, cmap = "viridis", annot = True,

fmt = ".2f", square = True, linewidths = .5, cbar\_kws = {"shrink": .5}, ax = ax)

heatmap.set\_title("Correlation Matrix", fontsize = 20, fontweight = 'bold')

plt.subplots\_adjust(left = 0.15, bottom = 0.15)

ax.tick\_params(labelsize = 12)

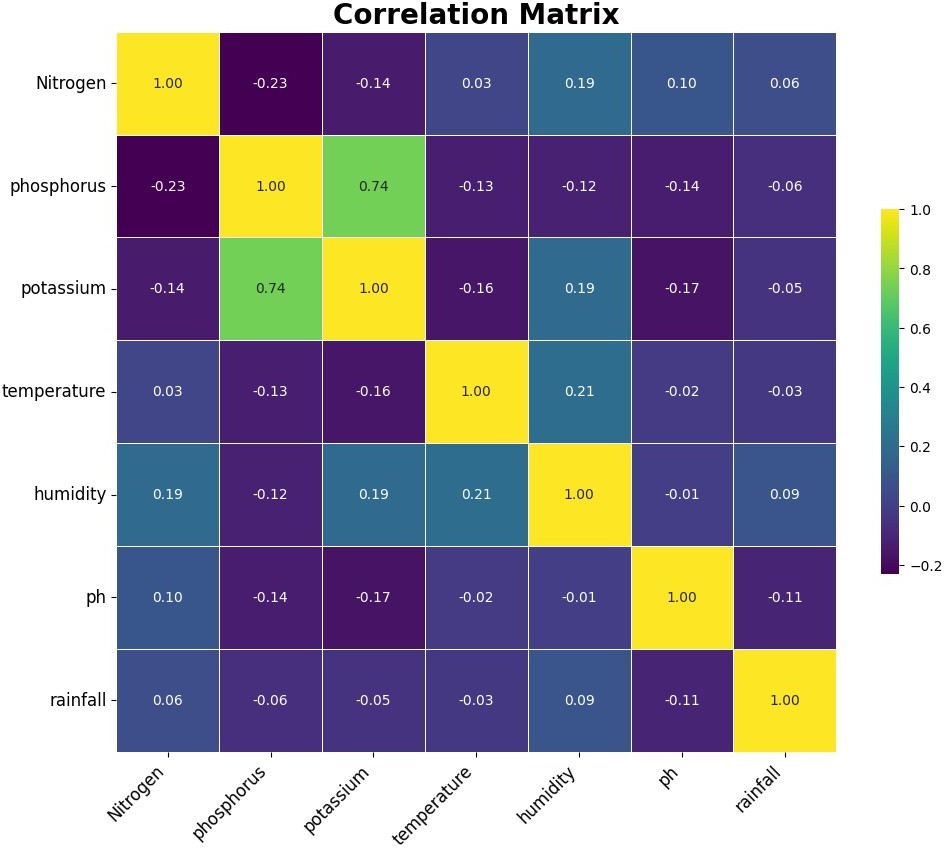
heatmap.set\_xticklabels(heatmap.get\_xticklabels(), rotation = 45, ha = 'right')

heatmap.set\_yticklabels(heatmap.get\_yticklabels(), rotation = 0, ha = 'right')

plt.show()

<ipython-input-29-91aa9334a4ab>:1: FutureWarning: The default value of numeric\_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric\_only to silence this warning.

corr\_matrix = df.corr()



Observation

* Phosphorus and potassium are highly correlated
* After that humidity and tempareture are correlated and play important factor in crop yeild.

Train Test Split

X = df.drop('label', axis = 1) y = df['label']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20, random\_state = 0)

# Modelling

Decision Tree

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *# Decision Tree Model Training*  from sklearn.tree import DecisionTreeClassifier dtree\_model = DecisionTreeClassifier() dtree\_model.fit(X\_train, y\_train)  DecisionTreeClassifier()  *# Model Evaluation*  from sklearn.metrics import accuracy\_score, classification\_report predictions = dtree\_model.predict(X\_test)  dtree\_accuracy = accuracy\_score(y\_test, predictions) print("Accuracy:", dtree\_accuracy\*100) print(classification\_report(y\_test, predictions))  Accuracy: 98.86363636363636  precision recall f1-score support | | | | | |  |
|  | apple | 1.00 | 1.00 | 1.00 | 18 |  |
|  | banana | 1.00 | 1.00 | 1.00 | 18 |  |
|  | blackgram | 1.00 | 0.95 | 0.98 | 22 |  |
|  | chickpea | 0.92 | 1.00 | 0.96 | 23 |  |
|  | coconut | 1.00 | 1.00 | 1.00 | 15 |  |
|  | coffee | 1.00 | 1.00 | 1.00 | 17 |  |
|  | cotton | 1.00 | 1.00 | 1.00 | 16 |  |
|  | grapes | 1.00 | 1.00 | 1.00 | 18 |  |
|  | jute | 0.95 | 0.95 | 0.95 | 21 |  |
|  | kidneybeans | 1.00 | 0.90 | 0.95 | 20 |  |
|  | lentil | 1.00 | 1.00 | 1.00 | 17 |  |
|  | maize | 1.00 | 1.00 | 1.00 | 18 |  |
|  | mango | 1.00 | 1.00 | 1.00 | 21 |  |
|  | mothbeans | 0.96 | 1.00 | 0.98 | 25 |  |
|  | mungbean | 1.00 | 1.00 | 1.00 | 17 |  |
|  | muskmelon | 1.00 | 1.00 | 1.00 | 23 |  |
|  | orange | 1.00 | 1.00 | 1.00 | 23 |  |
|  | papaya | 1.00 | 1.00 | 1.00 | 21 |  |
|  | pigeonpeas | 1.00 | 1.00 | 1.00 | 22 |  |
|  | pomegranate | 1.00 | 1.00 | 1.00 | 23 |  |
|  | rice | 0.96 | 0.96 | 0.96 | 25 |  |
|  | watermelon | 1.00 | 1.00 | 1.00 | 17 |  |
|  |  |  |  |  |  |  |
|  | accuracy |  |  | 0.99 | 440 |  |
|  | macro avg | 0.99 | 0.99 | 0.99 | 440 |  |
|  | weighted avg | 0.99 | 0.99 | 0.99 | 440 |  |
|  |  |  |  |  |  |  |
|  | Random Forest |  |  |  |  |  |

*# Random Forest*

from sklearn.ensemble import RandomForestClassifier rfc\_model = RandomForestClassifier() rfc\_model.fit(X\_train, y\_train)

RandomForestClassifier()

from sklearn.metrics import accuracy\_score, classification\_report predictions = rfc\_model.predict(X\_test)

rfc\_accuracy = accuracy\_score(y\_test, predictions) print("Accuracy:", rfc\_accuracy\*100) print(classification\_report(y\_test, predictions))

Accuracy: 99.77272727272727

precision recall f1-score support

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| apple | 1.00 | 1.00 | 1.00 | 18 |
| banana | 1.00 | 1.00 | 1.00 | 18 |
| blackgram | 1.00 | 1.00 | 1.00 | 22 |
| chickpea | 1.00 | 1.00 | 1.00 | 23 |
| coconut | 1.00 | 1.00 | 1.00 | 15 |
| coffee | 1.00 | 1.00 | 1.00 | 17 |
| cotton | 1.00 | 1.00 | 1.00 | 16 |
| grapes | 1.00 | 1.00 | 1.00 | 18 |
| jute | 0.95 | 1.00 | 0.98 | 21 |
| kidneybeans | 1.00 | 1.00 | 1.00 | 20 |
| lentil | 1.00 | 1.00 | 1.00 | 17 |
| maize | 1.00 | 1.00 | 1.00 | 18 |
| mango | 1.00 | 1.00 | 1.00 | 21 |
| mothbeans | 1.00 | 1.00 | 1.00 | 25 |
| mungbean | 1.00 | 1.00 | 1.00 | 17 |
| muskmelon | 1.00 | 1.00 | 1.00 | 23 |
| orange | 1.00 | 1.00 | 1.00 | 23 |
| papaya | 1.00 | 1.00 | 1.00 | 21 |
| pigeonpeas | 1.00 | 1.00 | 1.00 | 22 |
| pomegranate | 1.00 | 1.00 | 1.00 | 23 |
| rice | 1.00 | 0.96 | 0.98 | 25 |
| watermelon | 1.00 | 1.00 | 1.00 | 17 |
|  |  |  |  |  |
| accuracy |  |  | 1.00 | 440 |
| macro avg | 1.00 | 1.00 | 1.00 | 440 |
| weighted avg | 1.00 | 1.00 | 1.00 | 440 |

feature\_importance = rfc\_model.feature\_importances\_ feature\_names = X.columns

importance\_dict = dict(zip(feature\_names, feature\_importance)) sorted\_importance = sorted(importance\_dict.items(), key=lambda x: x[1], reverse=True)

print("Feature Importance:")

for feature, importance in sorted\_importance: print(f"{feature}: {importance}")

Feature Importance:

rainfall: 0.22348252779066283

humidity: 0.2160312589195526

potassium: 0.1746128215051649

phosphorus: 0.15153862344011582

Nitrogen: 0.10028259377271308

temperature: 0.07879270541118465

ph: 0.055259469160606256

KNN

from sklearn.preprocessing import StandardScaler scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train) X\_test\_scaled = scaler.transform(X\_test)

from sklearn.neighbors import KNeighborsClassifier

knn\_model = KNeighborsClassifier(n\_neighbors=5) *# You can choose the value of 'n\_neighbors'*

knn\_model.fit(X\_train\_scaled, y\_train) KNeighborsClassifier()

from sklearn.metrics import accuracy\_score, classification\_report predictions = knn\_model.predict(X\_test\_scaled)

knn\_accuracy = accuracy\_score(y\_test, predictions) print("Accuracy:", knn\_accuracy\*100) print(classification\_report(y\_test, predictions))

Accuracy: 97.72727272727273

precision recall f1-score support

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| apple | 1.00 | 1.00 | 1.00 | 18 |
| banana | 1.00 | 1.00 | 1.00 | 18 |
| blackgram | 0.91 | 0.95 | 0.93 | 22 |
| chickpea | 1.00 | 1.00 | 1.00 | 23 |
| coconut | 1.00 | 1.00 | 1.00 | 15 |
| coffee | 1.00 | 1.00 | 1.00 | 17 |
| cotton | 1.00 | 1.00 | 1.00 | 16 |
| grapes | 1.00 | 1.00 | 1.00 | 18 |
| jute | 0.84 | 1.00 | 0.91 | 21 |
| kidneybeans | 0.95 | 1.00 | 0.98 | 20 |
| lentil | 0.89 | 0.94 | 0.91 | 17 |
| maize | 1.00 | 1.00 | 1.00 | 18 |
| mango | 1.00 | 1.00 | 1.00 | 21 |
| mothbeans | 1.00 | 0.92 | 0.96 | 25 |
| mungbean | 1.00 | 1.00 | 1.00 | 17 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| muskmelon | 1.00 | 1.00 | 1.00 | 23 |
| orange | 1.00 | 0.96 | 0.98 | 23 |
| papaya | 1.00 | 1.00 | 1.00 | 21 |
| pigeonpeas | 1.00 | 0.95 | 0.98 | 22 |
| pomegranate | 0.96 | 1.00 | 0.98 | 23 |
| rice | 1.00 | 0.84 | 0.91 | 25 |
| watermelon | 1.00 | 1.00 | 1.00 | 17 |
|  |  |  |  |  |
| accuracy |  |  | 0.98 | 440 |
| macro avg | 0.98 | 0.98 | 0.98 | 440 |
|  | weighted avg | 0.98 | 0.98 | 0.98 | 440 |

models = ['Decision Tree', 'Random Forest Classifier', 'KNN'] Accuracy = [dtree\_accuracy\*100, rfc\_accuracy\*100, knn\_accuracy\*100]

acc\_table = {"Models": models, "Accuracy Scores": Accuracy} acc\_data = pd.DataFrame(acc\_table)

acc\_data

|  |  |
| --- | --- |
| Models | Accuracy Scores |
| 0 Decision Tree | 98.863636 |
| 1 Random Forest Classifier | 99.772727 |
| 2 KNN | 97.727273 |