crop-recommendation-system

July 9, 2024

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report
from sklearn import metrics
from sklearn import tree
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
df = pd.read_csv('/content/Crop_recommendation.csv')
df.head()
```

[2]:

	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

[3]:
```
df.tail()
```

[3]:

	N	P	K	temperature	humidity	ph	rainfall	label
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

[4]:
```
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
```

[5]:
```
df.isnull().sum()
```

[5]:
```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

[6]:
```
df.size
```

[6]:
```
17600
```

[7]:
```
df.shape
```

[7]:
```
(2200, 8)
```

[8]:
```
df.columns
```

[8]:
```
Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'],
dtype='object')
```

[9]:
```
df['label'].unique()
```

[9]:
```
array(['rice', 'maize', 'chickpea', 'kidneybeans', 'pigeonpeas',
       'mothbeans', 'mungbean', 'blackgram', 'lentil', 'pomegranate',
       'banana', 'mango', 'grapes', 'watermelon', 'muskmelon', 'apple',
       'orange', 'papaya', 'coconut', 'cotton', 'jute', 'coffee'],
      dtype=object)
```

[10]:
```
df.dtypes
```

[10]:
```
N                int64
P                int64
```

```
K                  int64
temperature      float64
humidity         float64
ph               float64
rainfall         float64
label             object
dtype: object
```

[11]:
```python
df['label'].value_counts()
```

[11]:
```
label
rice           100
maize          100
jute           100
cotton         100
coconut        100
papaya         100
orange         100
apple          100
muskmelon      100
watermelon     100
grapes         100
mango          100
banana         100
pomegranate    100
lentil         100
blackgram      100
mungbean       100
mothbeans      100
pigeonpeas     100
kidneybeans    100
chickpea       100
coffee         100
Name: count, dtype: int64
```

[33]:
```python
crop_summary = pd.pivot_table(df,index=['label'],aggfunc='mean')
crop_summary
```

[33]:

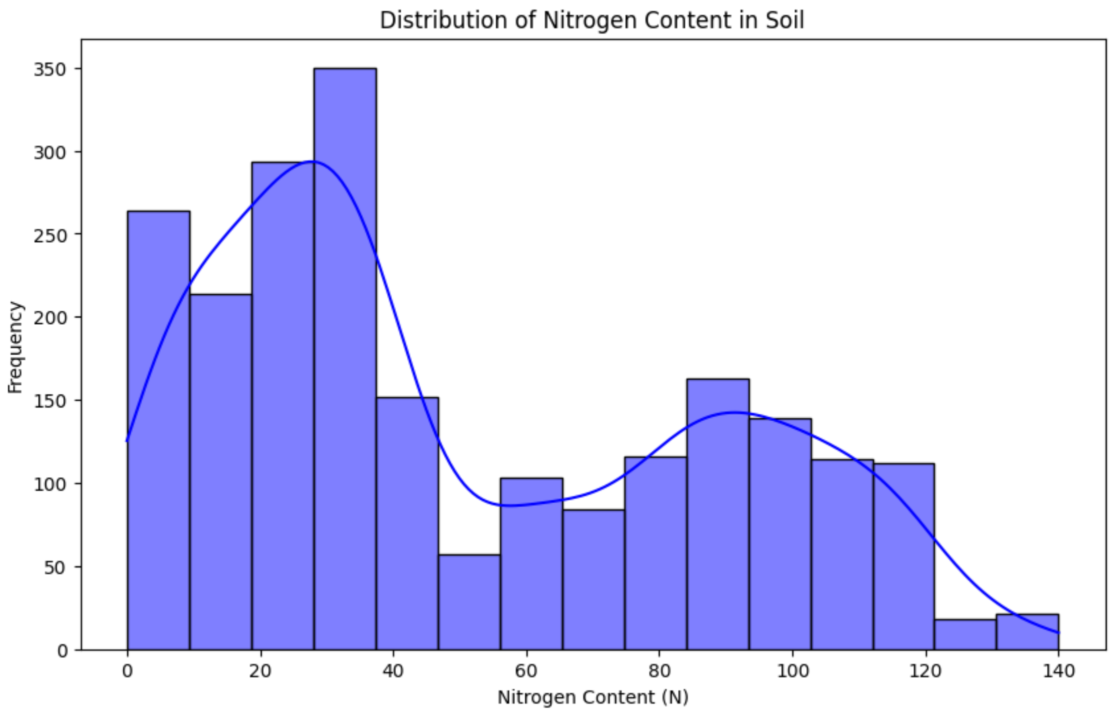
label	K	N	P	humidity	ph	rainfall	\
apple	199.89	20.80	134.22	92.333383	5.929663	112.654779	
banana	50.05	100.23	82.01	80.358123	5.983893	104.626980	
blackgram	19.24	40.02	67.47	65.118426	7.133952	67.884151	
chickpea	79.92	40.09	67.79	16.860439	7.336957	80.058977	
coconut	30.59	21.98	16.93	94.844272	5.976562	175.686646	
coffee	29.94	101.20	28.74	58.869846	6.790308	158.066295	
cotton	19.56	117.77	46.24	79.843474	6.912675	80.398043	

```
grapes        200.11   23.18  132.53  81.875228  6.025937   69.611829
jute           39.99   78.40   46.86  79.639864  6.732778  174.792798
kidneybeans    20.05   20.75   67.54  21.605357  5.749411  105.919778
lentil         19.41   18.77   68.36  64.804785  6.927932   45.680454
maize          19.79   77.76   48.44  65.092249  6.245190   84.766988
mango          29.92   20.07   27.18  50.156573  5.766373   94.704515
mothbeans      20.23   21.44   48.01  53.160418  6.831174   51.198487
mungbean       19.87   20.99   47.28  85.499975  6.723957   48.403601
muskmelon      50.08  100.32   17.72  92.342802  6.358805   24.689952
orange         10.01   19.58   16.55  92.170209  7.016957  110.474969
papaya         50.04   49.88   59.05  92.403388  6.741442  142.627839
pigeonpeas     20.29   20.73   67.73  48.061633  5.794175  149.457564
pomegranate    40.21   18.87   18.75  90.125504  6.429172  107.528442
rice           39.87   79.89   47.58  82.272822  6.425471  236.181114
watermelon     50.22   99.42   17.00  85.160375  6.495778   50.786219

             temperature
label
apple          22.630942
banana         27.376798
blackgram      29.973340
chickpea       18.872847
coconut        27.409892
coffee         25.540477
cotton         23.988958
grapes         23.849575
jute           24.958376
kidneybeans    20.115085
lentil         24.509052
maize          22.389204
mango          31.208770
mothbeans      28.194920
mungbean       28.525775
muskmelon      28.663066
orange         22.765725
papaya         33.723859
pigeonpeas     27.741762
pomegranate    21.837842
rice           23.689332
watermelon     25.591767
```

[12]:

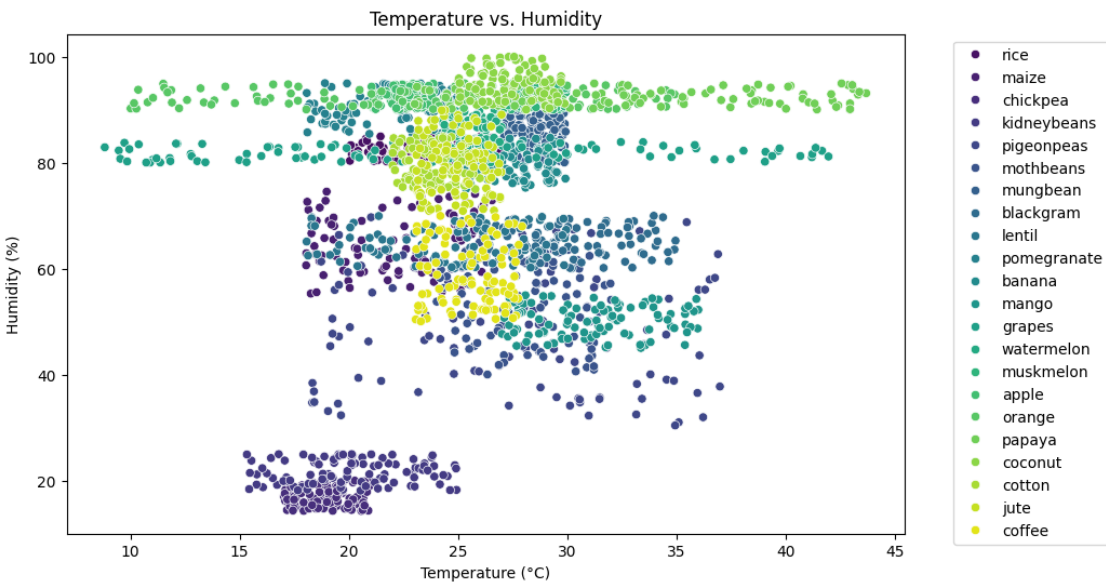
```python
# Visualization 1: Histogram of Nitrogen (N) Content
plt.figure(figsize=(10, 6))
sns.histplot(df['N'], kde=True, color='blue')

plt.title('Distribution of Nitrogen Content in Soil')
plt.xlabel('Nitrogen Content (N)')
```

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

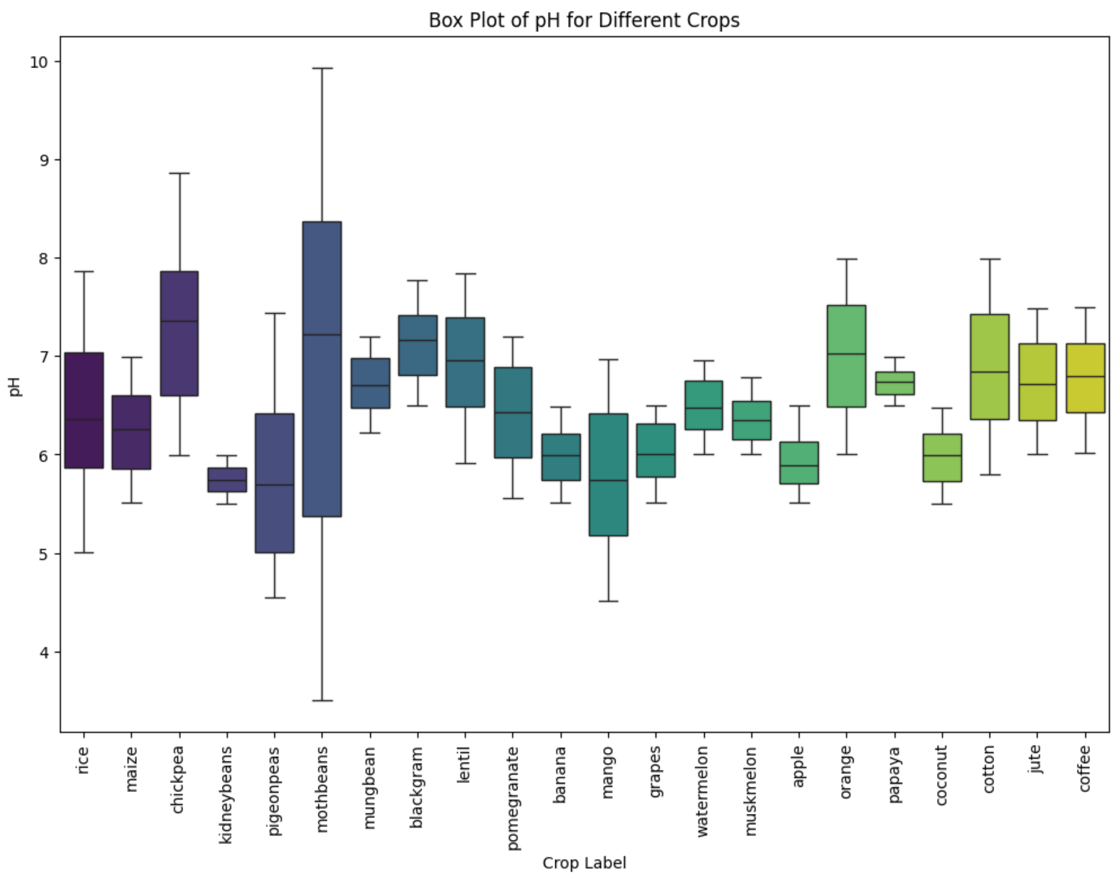


[13]:

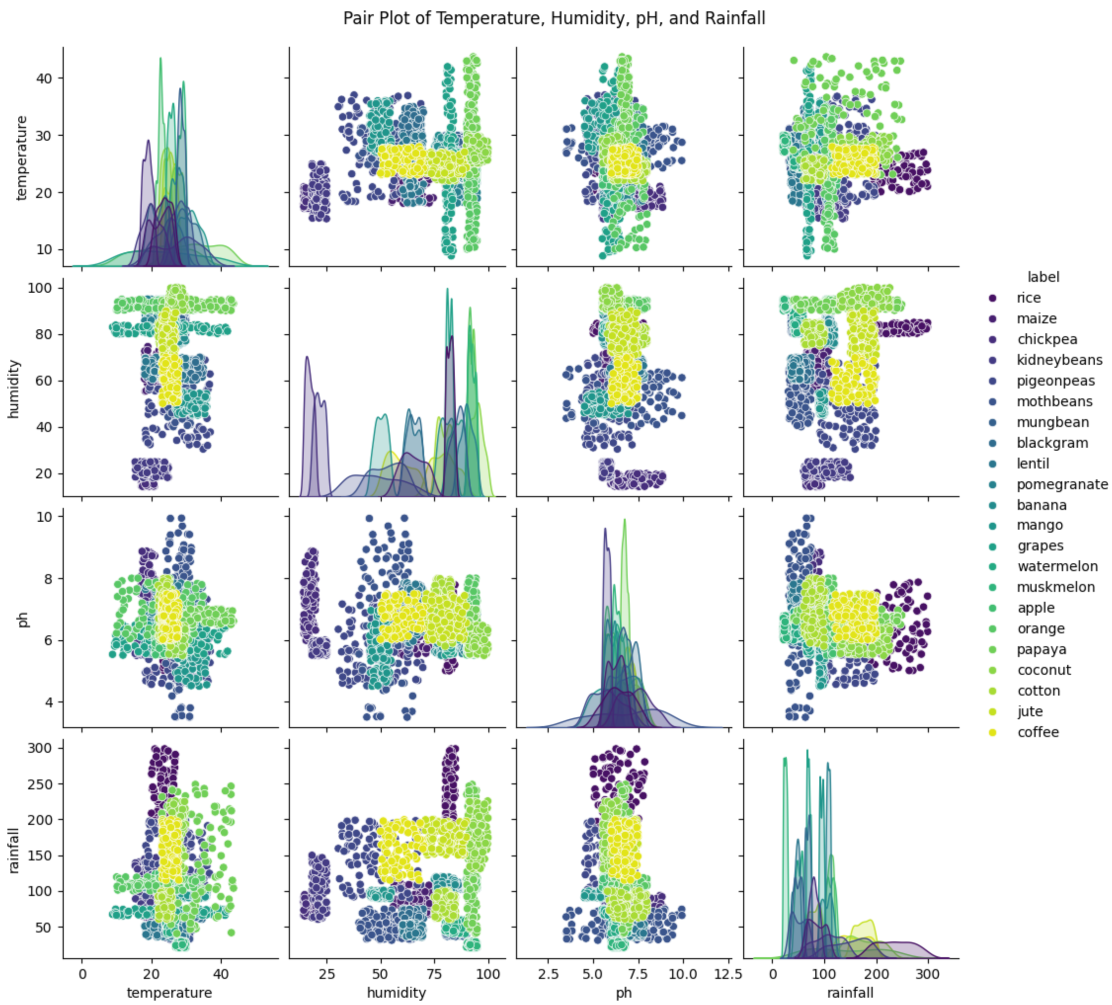
```
# Visualization 2: Scatter Plot of Temperature vs. Humidity
plt.figure(figsize=(10, 6))
sns.scatterplot(x=df['temperature'], y=df['humidity'], hue=df['label'],␣
 ↪palette='viridis')
plt.title('Temperature vs. Humidity')
plt.xlabel('Temperature (°C)')
plt.ylabel('Humidity (%)')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



[14]:
```python
# Visualization 3: Box Plot of pH for Different Crops
plt.figure(figsize=(12, 8))
sns.boxplot(x='label', y='ph', data=df, palette='viridis')
plt.title('Box Plot of pH for Different Crops')
plt.xlabel('Crop Label')
plt.ylabel('pH')
plt.xticks(rotation=90)
plt.show()
```

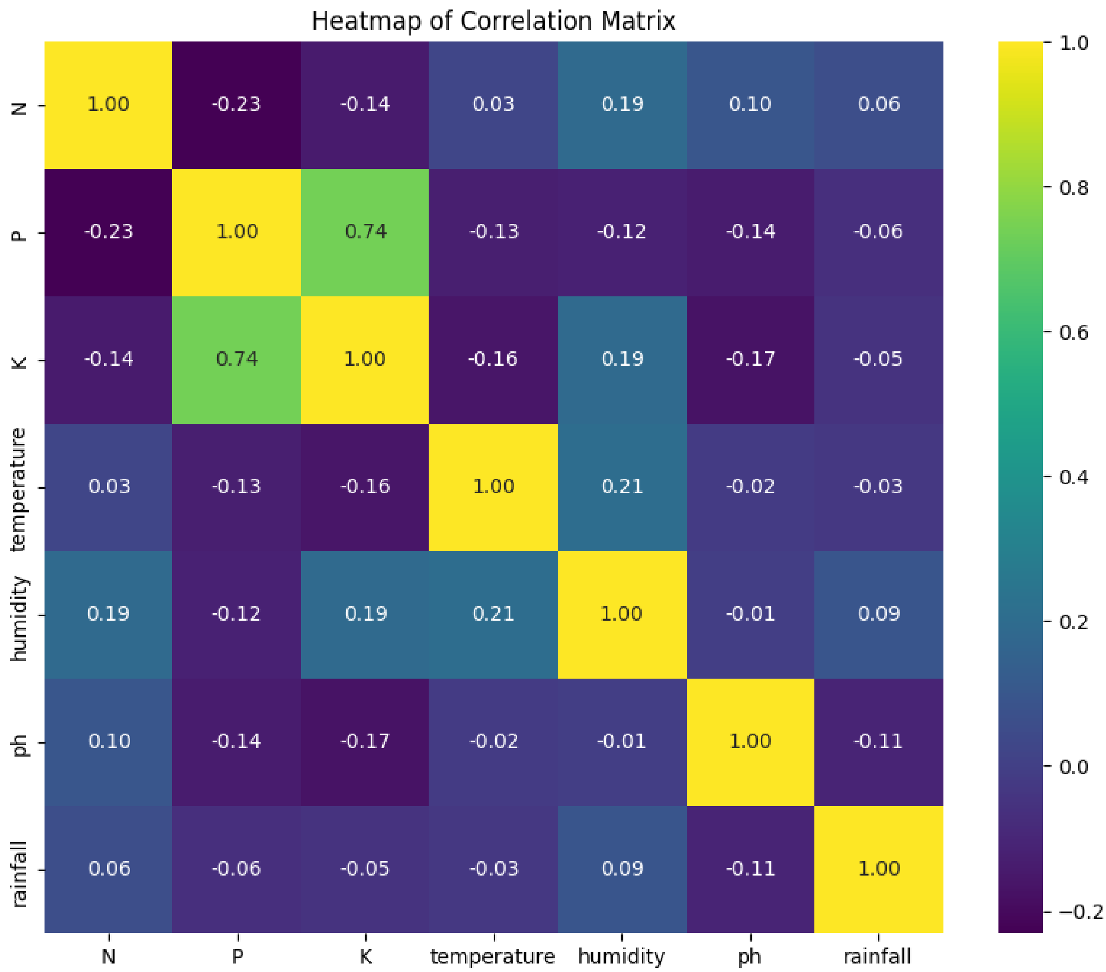


[15]:
```
# Visualization 4: Pair Plot of Temperature, Humidity, pH, and Rainfall
pairplot_features = ['temperature', 'humidity', 'ph', 'rainfall', 'label']
sns.pairplot(df[pairplot_features], hue='label', palette='viridis',␣
  ↪diag_kind='kde')
plt.suptitle('Pair Plot of Temperature, Humidity, pH, and Rainfall', y=1.02)
plt.show()
```

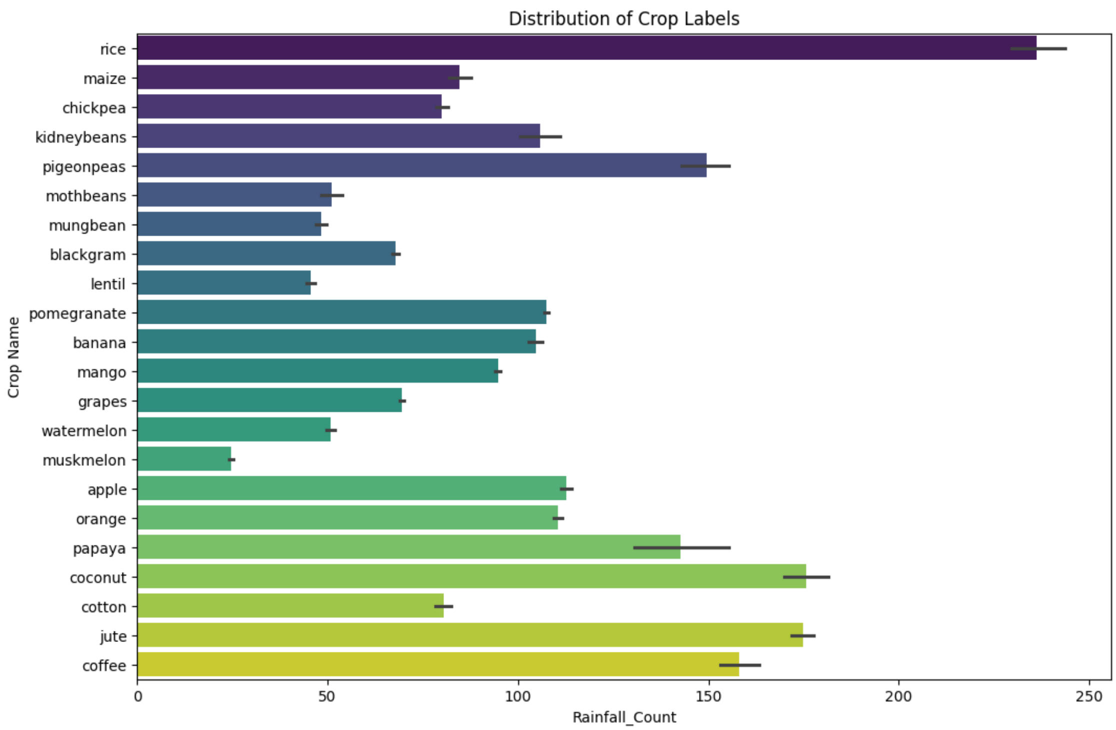


[16]:
```
# Ensure only numerical columns are included in the correlation matrix
numerical_data = df.select_dtypes(include=['float64', 'int64'])
```

[17]:
```
# Visualization 5: Heatmap of the Correlation Matrix
plt.figure(figsize=(10, 8))
sns.heatmap(numerical_data.corr(),annot=True,cmap='viridis', fmt='.2f')
plt.title('Heatmap of Correlation Matrix')
plt.show()
```



[18]:
```python
# Visualization: Bar Plot of Crop Label Counts
plt.figure(figsize=(12, 8))
sns.barplot(y='label', x='rainfall', data=df, palette='viridis')
plt.title('Distribution of Crop Labels')
plt.xlabel('Rainfall_Count')
plt.ylabel('Crop Name')
plt.show()
```



[19]:
```
x = df.drop('label', axis = 1)
y = df['label']
```

[21]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.
↪2,random_state =2)
```

#1.Decision Tree

[22]:
```
from sklearn.tree import DecisionTreeClassifier
model_1 = DecisionTreeClassifier(criterion='entropy',max_depth = 6,␣
↪random_state = 2)
model_1.fit(x_train, y_train)
y_pred_1 = model_1.predict(x_test)
decision_acc = accuracy_score(y_test, y_pred_1)
print("Accuracy of decision  tree is : = " + str(decision_acc))
print(classification_report(y_test,y_pred_1))
```

```
Accuracy of decision  tree is : = 0.9590909090909091
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       0.80      1.00      0.89        16
```

```
    chickpea       1.00      1.00      1.00        21
     coconut       1.00      1.00      1.00        21
      coffee       1.00      1.00      1.00        22
      cotton       1.00      1.00      1.00        20
      grapes       1.00      1.00      1.00        18
        jute       0.84      0.96      0.90        28
 kidneybeans       1.00      0.79      0.88        14
      lentil       0.95      0.83      0.88        23
       maize       1.00      1.00      1.00        21
       mango       1.00      1.00      1.00        26
   mothbeans       0.67      0.74      0.70        19
    mungbean       1.00      1.00      1.00        24
   muskmelon       1.00      1.00      1.00        23
      orange       1.00      1.00      1.00        29
      papaya       1.00      1.00      1.00        19
  pigeonpeas       1.00      1.00      1.00        18
 pomegranate       1.00      1.00      1.00        17
        rice       0.92      0.69      0.79        16
  watermelon       1.00      1.00      1.00        15

    accuracy                           0.96       440
   macro avg       0.96      0.95      0.96       440
weighted avg       0.96      0.96      0.96       440
```

##2.LogisticRegression

[23]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
logistic_acc = accuracy_score(y_test, y_pred)
print("Accuracy of logistic regression is : =  " + str(logistic_acc))
print(classification_report(y_test,y_pred))
```

```
Accuracy of logistic regression is : =  0.9522727272727273
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       0.86      0.75      0.80        16
    chickpea       1.00      1.00      1.00        21
     coconut       1.00      1.00      1.00        21
      coffee       1.00      1.00      1.00        22
      cotton       0.86      0.90      0.88        20
      grapes       1.00      1.00      1.00        18
        jute       0.84      0.93      0.88        28
 kidneybeans       1.00      1.00      1.00        14
```

```
     lentil       0.88      1.00      0.94        23
      maize       0.90      0.86      0.88        21
      mango       0.96      1.00      0.98        26
   mothbeans       0.84      0.84      0.84        19
    mungbean       1.00      0.96      0.98        24
   muskmelon       1.00      1.00      1.00        23
     orange       1.00      1.00      1.00        29
     papaya       1.00      0.95      0.97        19
  pigeonpeas       1.00      1.00      1.00        18
 pomegranate       1.00      1.00      1.00        17
       rice       0.85      0.69      0.76        16
  watermelon       1.00      1.00      1.00        15

    accuracy                           0.95       440
   macro avg       0.95      0.95      0.95       440
weighted avg       0.95      0.95      0.95       440
```

##3.GaussianNB

[24]:
```python
from sklearn.naive_bayes import GaussianNB
model_3 = GaussianNB()
model_3.fit(x_train, y_train)
y_pred_3 = model_3.predict(x_test)
naive_bayes_acc = accuracy_score(y_test, y_pred_3)
print("Accuracy of naive_bayes is  : = " + str(naive_bayes_acc))
print(classification_report(y_test,y_pred_3))
```

```
Accuracy of naive_bayes is  : = 0.990909090909091
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       1.00      1.00      1.00        16
    chickpea       1.00      1.00      1.00        21
     coconut       1.00      1.00      1.00        21
      coffee       1.00      1.00      1.00        22
      cotton       1.00      1.00      1.00        20
      grapes       1.00      1.00      1.00        18
        jute       0.88      1.00      0.93        28
 kidneybeans       1.00      1.00      1.00        14
      lentil       1.00      1.00      1.00        23
       maize       1.00      1.00      1.00        21
       mango       1.00      1.00      1.00        26
   mothbeans       1.00      1.00      1.00        19
    mungbean       1.00      1.00      1.00        24
   muskmelon       1.00      1.00      1.00        23
      orange       1.00      1.00      1.00        29
```

```
      papaya       1.00      1.00      1.00        19
  pigeonpeas       1.00      1.00      1.00        18
 pomegranate       1.00      1.00      1.00        17
        rice       1.00      0.75      0.86        16
  watermelon       1.00      1.00      1.00        15

    accuracy                           0.99       440
   macro avg       0.99      0.99      0.99       440
weighted avg       0.99      0.99      0.99       440
```

##4.Support Vector Machine

[25]:
```
from sklearn.svm import SVC
model_svm = SVC()
model_svm.fit(x_train, y_train)
y_pred_svm = model_svm.predict(x_test)
svm_acc = accuracy_score(y_test, y_pred_svm)
print("Accuracy of Support Vector Machine is: " + str(svm_acc))
print(classification_report(y_test,y_pred_svm))
```

```
Accuracy of Support Vector Machine is: 0.9772727272727273
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       0.94      1.00      0.97        16
    chickpea       1.00      1.00      1.00        21
     coconut       1.00      1.00      1.00        21
      coffee       1.00      1.00      1.00        22
      cotton       0.95      1.00      0.98        20
      grapes       1.00      1.00      1.00        18
        jute       0.85      1.00      0.92        28
 kidneybeans       0.93      1.00      0.97        14
      lentil       0.92      1.00      0.96        23
       maize       1.00      0.95      0.98        21
       mango       1.00      1.00      1.00        26
   mothbeans       1.00      0.84      0.91        19
    mungbean       1.00      1.00      1.00        24
   muskmelon       1.00      1.00      1.00        23
      orange       1.00      1.00      1.00        29
      papaya       1.00      1.00      1.00        19
  pigeonpeas       1.00      0.94      0.97        18
 pomegranate       1.00      1.00      1.00        17
        rice       1.00      0.69      0.81        16
  watermelon       1.00      1.00      1.00        15

    accuracy                           0.98       440
```

```
   macro avg       0.98      0.97      0.98       440
weighted avg       0.98      0.98      0.98       440
```

##5.Random Forest

[26]:
```
from sklearn.ensemble import RandomForestClassifier
model_rf = RandomForestClassifier()
model_rf.fit(x_train, y_train)
y_pred_rf = model_rf.predict(x_test)
rf_acc = accuracy_score(y_test, y_pred_svm)
print("Accuracy of Support Vector Machine is: " + str(rf_acc))
print(classification_report(y_test,y_pred_rf))
```

```
Accuracy of Support Vector Machine is: 0.9772727272727273
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       1.00      1.00      1.00        16
    chickpea       1.00      1.00      1.00        21
     coconut       1.00      1.00      1.00        21
      coffee       1.00      1.00      1.00        22
      cotton       1.00      1.00      1.00        20
      grapes       1.00      1.00      1.00        18
        jute       0.93      1.00      0.97        28
 kidneybeans       1.00      1.00      1.00        14
      lentil       1.00      1.00      1.00        23
       maize       1.00      1.00      1.00        21
       mango       1.00      1.00      1.00        26
   mothbeans       1.00      1.00      1.00        19
    mungbean       1.00      1.00      1.00        24
   muskmelon       1.00      1.00      1.00        23
      orange       1.00      1.00      1.00        29
      papaya       1.00      1.00      1.00        19
  pigeonpeas       1.00      1.00      1.00        18
 pomegranate       1.00      1.00      1.00        17
        rice       1.00      0.88      0.93        16
  watermelon       1.00      1.00      1.00        15

    accuracy                           1.00       440
   macro avg       1.00      0.99      1.00       440
weighted avg       1.00      1.00      1.00       440
```

[ ]:
```
```

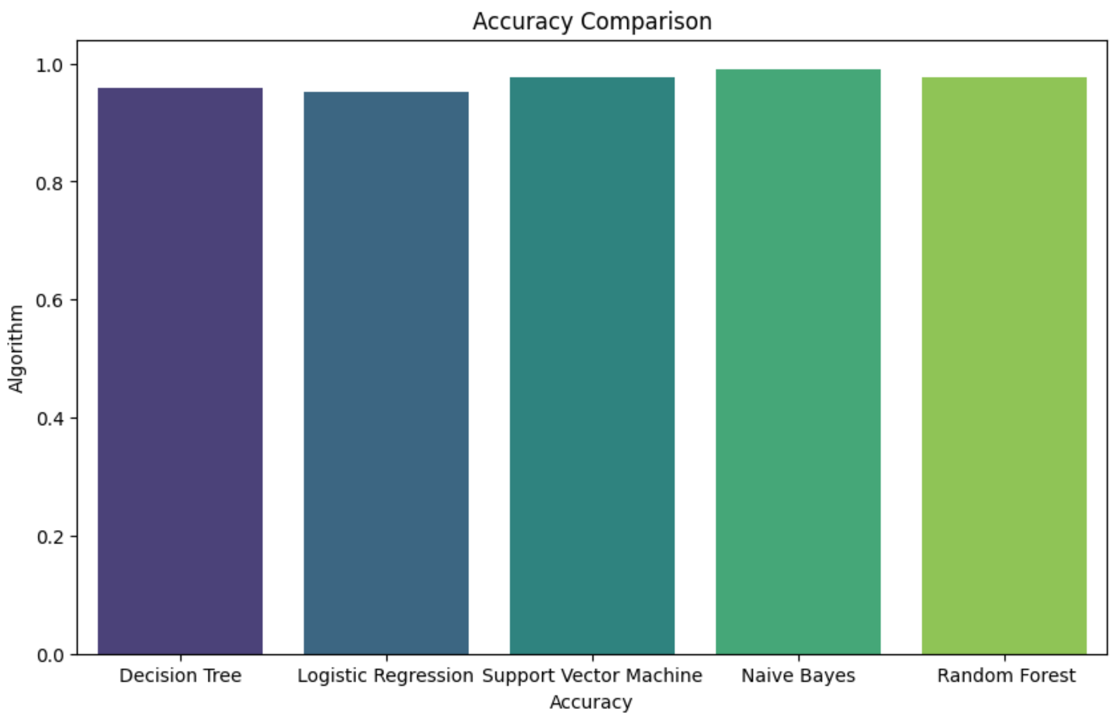
[27]:
```
print("Accuracy of decision  tree is : = " + str(decision_acc))
print("Accuracy of logistic regression is : =  " + str(logistic_acc))
print("Accuracy of Support Vector Machine is: " + str(svm_acc))
print("Accuracy of naive_bayes is  : = " + str(naive_bayes_acc))
```

```
Accuracy of decision  tree is : = 0.9590909090909091
Accuracy of logistic regression is : =  0.9522727272727273
Accuracy of Support Vector Machine is: 0.9772727272727273
Accuracy of naive_bayes is  : = 0.990909090909091
```

Accuracy Comparison

[28]:
```
accuracies = [decision_acc, logistic_acc, svm_acc, naive_bayes_acc,rf_acc]
models = ['Decision Tree', 'Logistic Regression', 'Support Vector Machine',␣
 ↪'Naive Bayes', " Random Forest"]
```

[29]:
```
plt.figure(figsize=(10,6))
sns.barplot(x=models, y=accuracies, palette='viridis')
plt.title('Accuracy Comparison')
plt.xlabel('Accuracy')
plt.ylabel('Algorithm')
plt.show()
```



0.1 Making a prediction

[30]:
```
data = np.array([[104,18, 30, 23.603016, 60.3, 6.7, 140.91]])
prediction = model_3.predict(data)
print(prediction)
```

```
['coffee']
```

[31]:
```
data = np.array([[60,        55,        44        ,23.004459,82.320763,7.
↪840207,        263.964248]])
prediction = model_rf.predict(data)
print(prediction)
```

```
['rice']
```

Conclusion

The SVM model achieved the highest accuracy for the crop recommendation system, of approximately 0.18. This suggests that the SVM model is highly effective in predicting the appropriate crops based on soil and environmental features. For further improvements, ensemble techniques like Bagging and Boosting could be employed to enhance the model's performance. This system can significantly assist farmers in optimizing crop selection, leading to improved agricultural productivity and sustainability.

[ ]:
```
```