Report:Agrisage

AUTHOR
Subhangi Pandey

Abstract

This project focuses on the development and implementation of a predictive model for crop selection based on a comprehensive agricultural dataset. The dataset comprises multiple features such as nutrients in soil, weather conditions, and other agronomic factors, with the target variable being the crop type.

The primary objective was to analyze the dataset to understand the key factors influencing crop selection, followed by the construction of a predictive model to recommend the most suitable crops for given conditions. Various data preprocessing techniques and exploratory data analysis (EDA) were employed to enhance the quality of the data.

Multiple machine learning algorithms were evaluated, including decision trees, random forests, and logistic regression, to determine the most accurate model. The final model was selected based on its accuracy.

Additionally, a Streamlit application was developed to provide an interactive user interface for stakeholders, enabling them to input specific agronomic conditions and receive crop recommendations with associated probabilities. This tool aims to support farmers and agricultural planners in making informed decisions to optimize crop yields and sustainability. Additionally, it ensures the validity of the model presented.

Introduction

We have a dataset that showcases what crop is best suited for a combination of different features(quantitatively) like Nitrogen, Phosphorus, Potassium and so on.

▶ Code

▶ Code

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Nitrogen     2200 non-null   int64  
 1   Phosphorus   2200 non-null   int64  
 2   Potassium    2200 non-null   int64  
 3   Temperature  2200 non-null   float64
 4   Humidity     2200 non-null   float64
 5   pH_Value     2200 non-null   float64
 6   Rainfall     2200 non-null   float64
 7   Crop         2200 non-null   object 
```

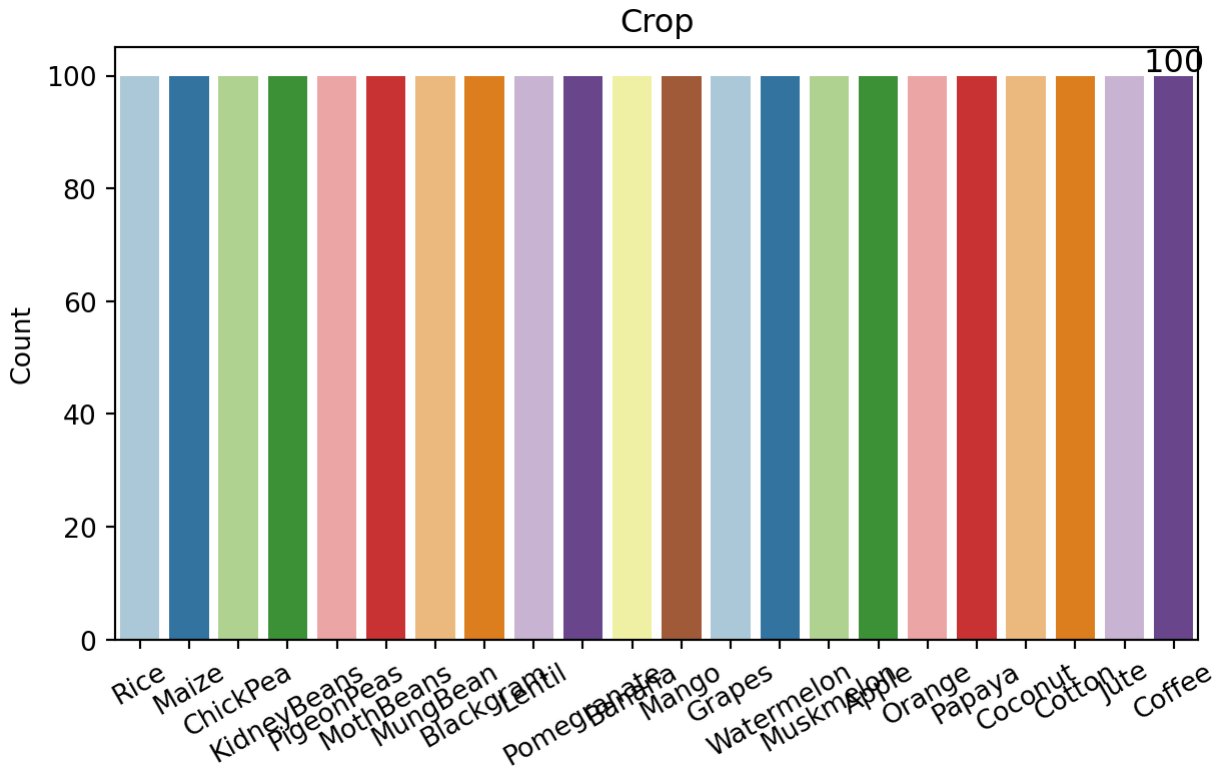
```
dtypes: float64(4), int64(3), object(1)
memory usage: 137.6+ KB
```

▶ Code

Exploratory Data Analysis (EDA)

▶ Code

```
<Figure size 672x480 with 0 Axes>
```



The above figure helps us in a clear visualization of our data.

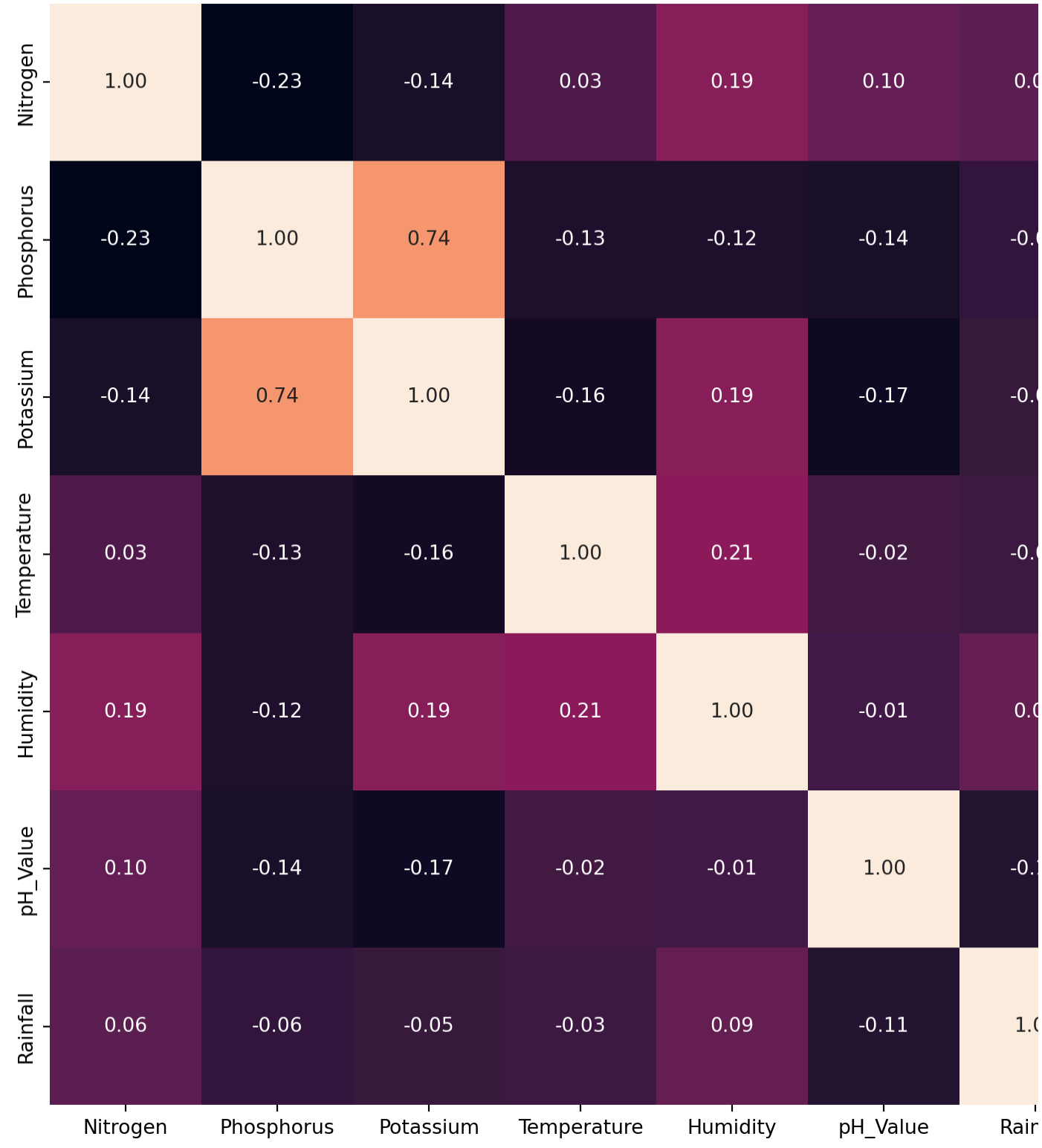
1. Quantification: We have 22 possible targets that is crops for our datset and since each crop has a 100 samples, we have 2200 samples.

2. Class Balance: The fact that each crop has an equal number of samples (100) suggests that the dataset is balanced. This is beneficial for training machine learning models as it can prevent bias towards any particular class.

3. Granularity: In a classification problem like the one presented here, having 100 samples per target allows for a detailed representation of each class.

However, the number of samples is a subject to change after further EDA.

▶ Code



The above figures tells us that Phosphorus and Potassium have a high positive correation. This can be due to several underlying reasons related to soil chemistry, plant physiology, and farming practices. Here are some common reasons why these two nutrients might show correlation:

1. Soil Composition: Phosphorus (P) and potassium (K) availability in soil can be influenced by similar factors such as soil type, pH levels, and organic matter content.

2. Fertilizer Application: Farmers often apply fertilizers that contain both phosphorus and potassium together. This simultaneous application can lead to their concentrations being correlated in the soil.

3. Sampling and Analysis: Sometimes, the correlation observed could be due to the way samples are collected or analyzed. If samples are taken from similar locations or depths within a field, they may show similar nutrient profiles.

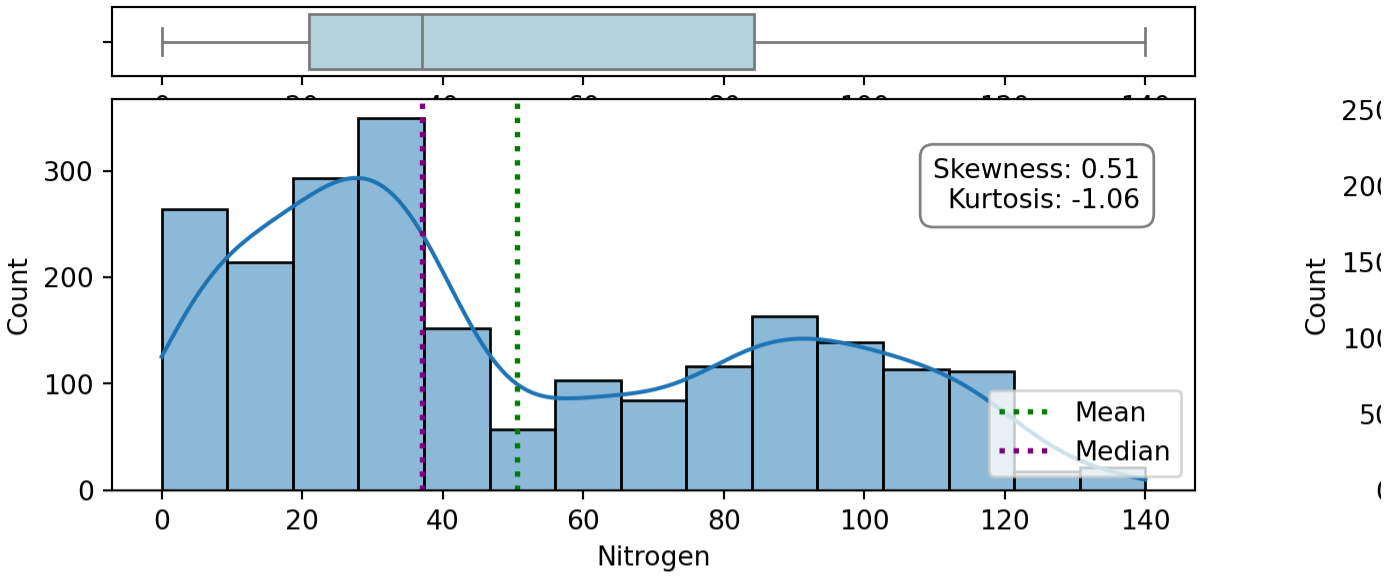
4. Environmental Factors: Environmental conditions such as rainfall, temperature, and humidity can affect the mobility and availability of both phosphorus and potassium in the soil. Similar environmental impacts can result in correlated values. In our datset since these factors would be same for a particular area, the correlation is understandable.

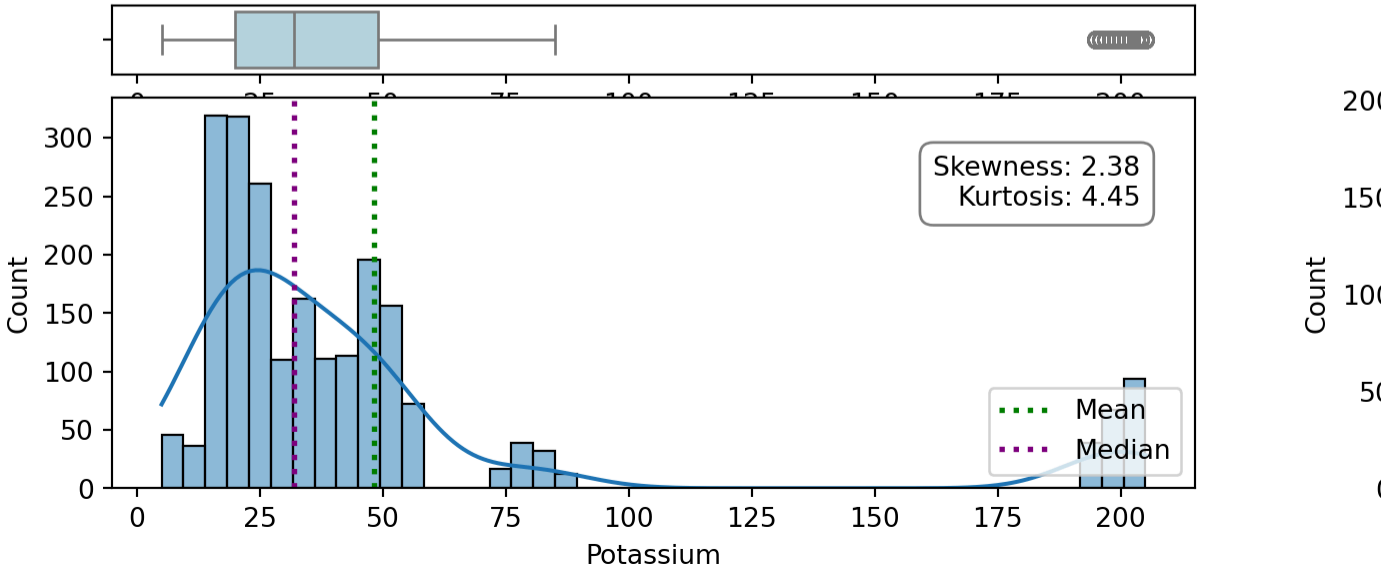
Highly correlated variables can adversely affect the performance of predictive models, particularly in regression and classification tasks:

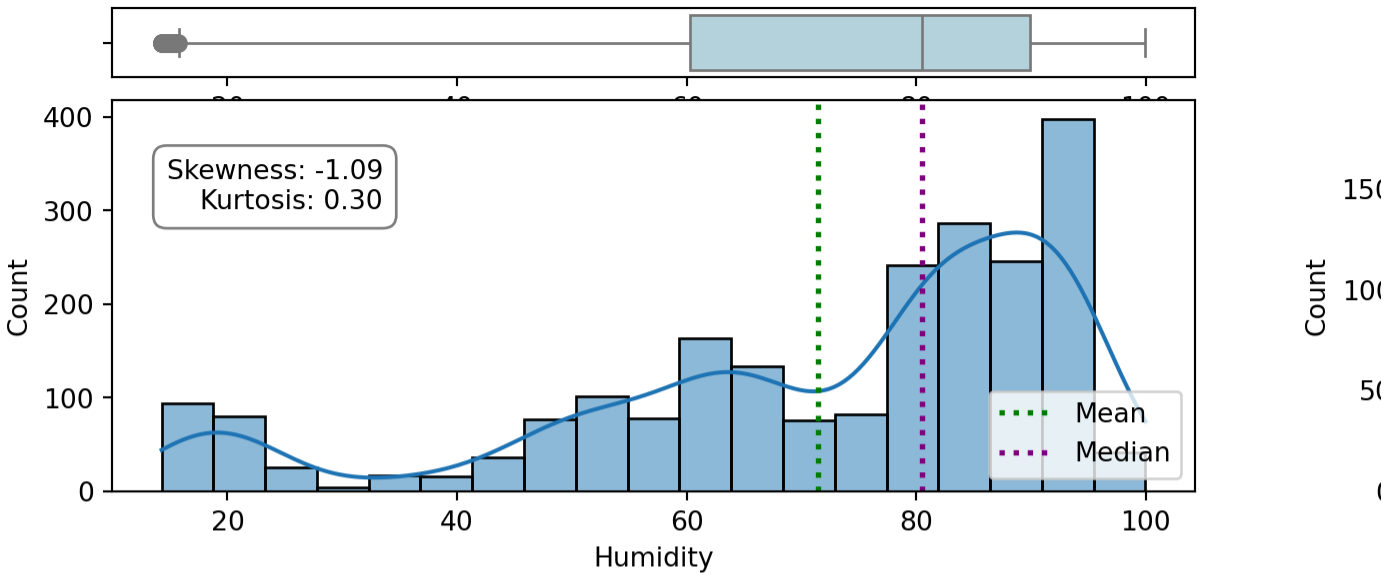
1. It can lead to multicollinearity, where the coefficients become unstable and difficult to interpret in regression models.

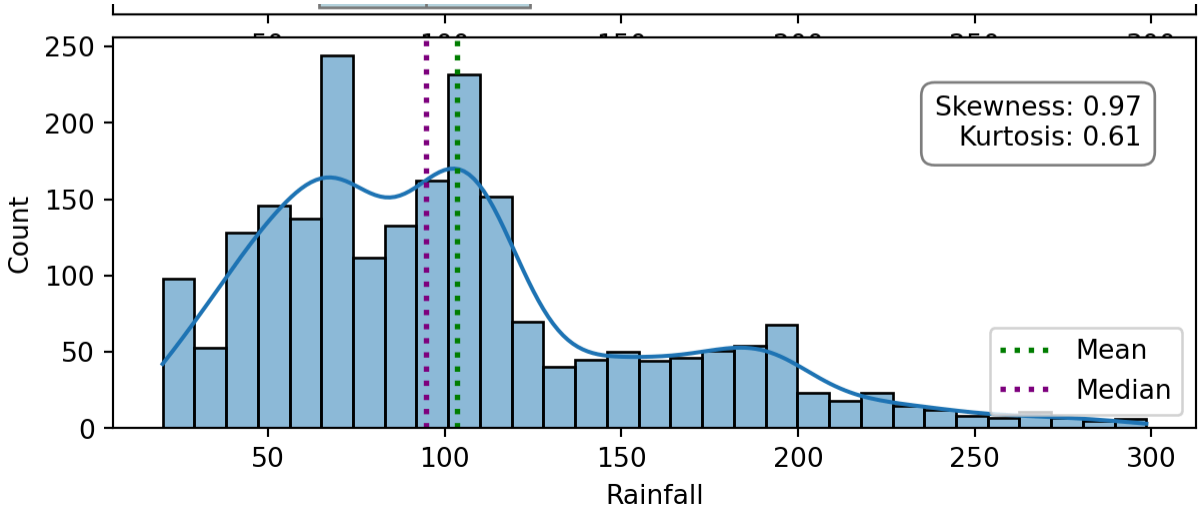
2. It can also lead to OVERFITTING occurs when the model captures noise or random fluctuations rather than underlying patterns.

▶ Code


Skewness: 0.51
Kurtosis: -1.06
Mean
Median
Count
Nitrogen


Skewness: 2.38
Kurtosis: 4.45
Mean
Median
Count
Potassium


Skewness: -1.09
Kurtosis: 0.30
Mean
Median
Count
Humidity



From the above graph, we are able to easily identify the outliers that need to be dealt with as they can negatively influence the performance of machine learning models. These are a few outliers:

1. 100 > Potassium> 200
2. 15> Temperature >40
3. 4> pH_Value >8
4. count > 250

By addressing outliers, we can improve the robustness and reliability of your models. It can also help in better visualization.

Handling Outliers

The Interquartile Range (IQR) method is a common statistical technique used to identify and handle outliers in a dataset. The IQR is the range between the first quartile (Q1) and the third quartile (Q3) of the dataset.

▶ Code

```
25th percentile of the given data is
 Nitrogen       21.000000
Phosphorus     28.000000
Potassium      20.000000
Temperature    22.769375
Humidity       60.261953
pH_Value        5.971693
Rainfall       64.551686
Name: 0.25, dtype: float64
75th percentile of the given data is
 Nitrogen        84.250000
Phosphorus      68.000000
Potassium       49.000000
```

```
Temperature     28.561654
Humidity        89.948771
pH_Value         6.923643
Rainfall       124.267508
Name: 0.75, dtype: float64
Interquartile range is
 Nitrogen       63.250000
Phosphorus     40.000000
Potassium      29.000000
Temperature     5.792279
Humidity       29.686818
pH_Value        0.951950
Rainfall       59.715822
dtype: float64
```

▶ Code

```
lower bound of the given data is
 Nitrogen      -73.875000
Phosphorus    -32.000000
Potassium     -23.500000
Temperature    14.080956
Humidity       15.731726
pH_Value        4.543768
Rainfall      -25.022047
dtype: float64
upper bound of the given data is
 Nitrogen       179.125000
Phosphorus     128.000000
Potassium       92.500000
Temperature     37.250073
Humidity       134.478998
pH_Value         8.351567
Rainfall       213.841241
dtype: float64
```

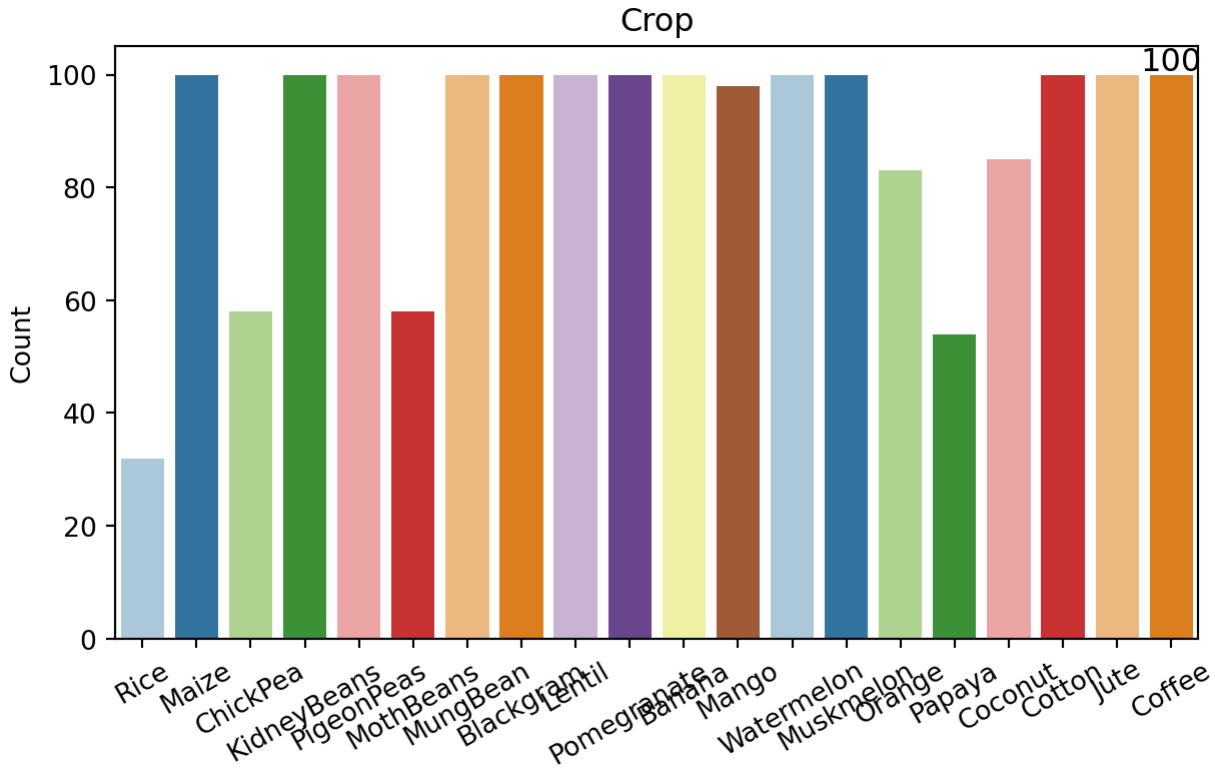
▶ Code

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1768 entries, 0 to 1767
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Nitrogen     1768 non-null   int64
 1   Phosphorus   1768 non-null   int64
 2   Potassium    1768 non-null   int64
 3   Temperature  1768 non-null   float64
 4   Humidity     1768 non-null   float64
 5   pH_Value     1768 non-null   float64
 6   Rainfall     1768 non-null   float64
 7   Crop         1768 non-null   object
```

```
dtypes: float64(4), int64(3), object(1)
memory usage: 110.6+ KB
```

These are the number of crop samples left after removing the outliers along with graph:

▶ Code



It is clear to us that we cannot remove outliers as it leads to serious reduction of samples for some crops like rice and papaya. Reducing the number of samples can decrease the statistical power of our analysis. With fewer samples, the ability to detect true patterns or relationships in the data diminishes.

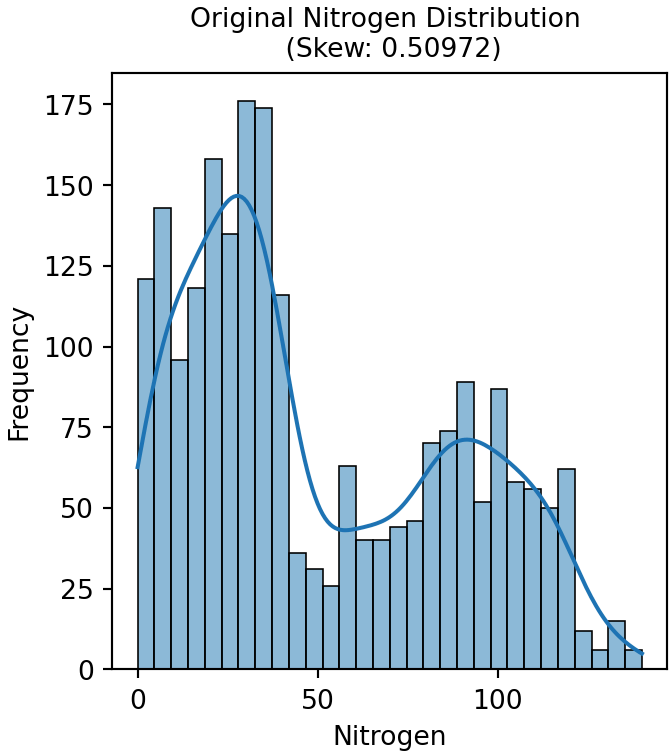
Retaining outliers might provide a more comprehensive understanding of the factors influencing crop yield, leading to more robust and reliable predictions.

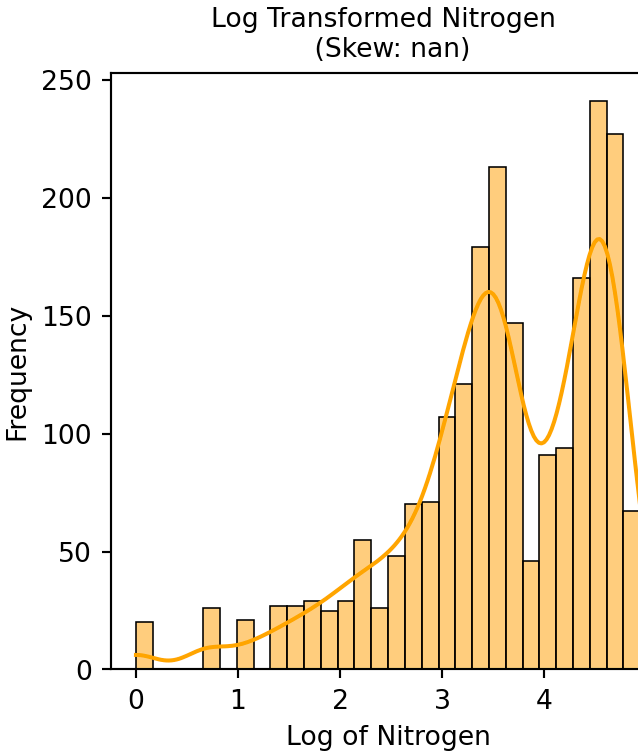
We transform the data:

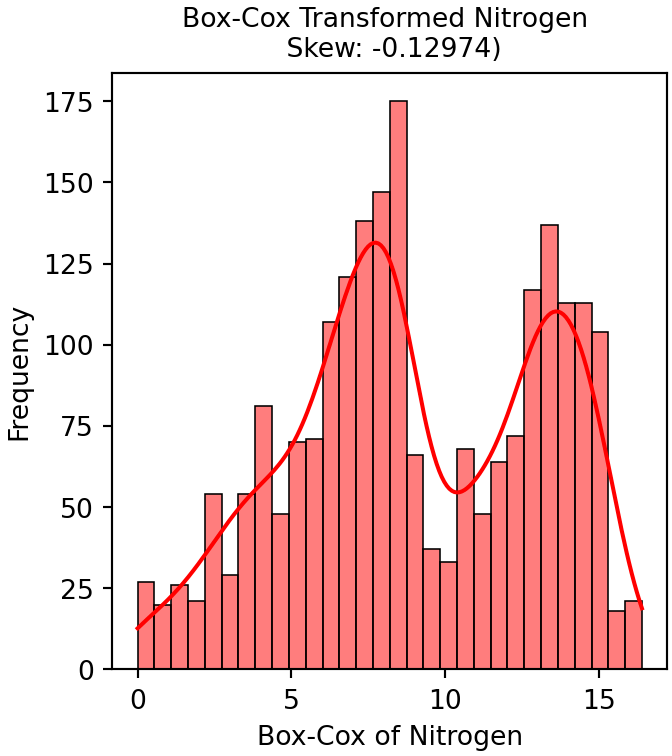
▶ Code

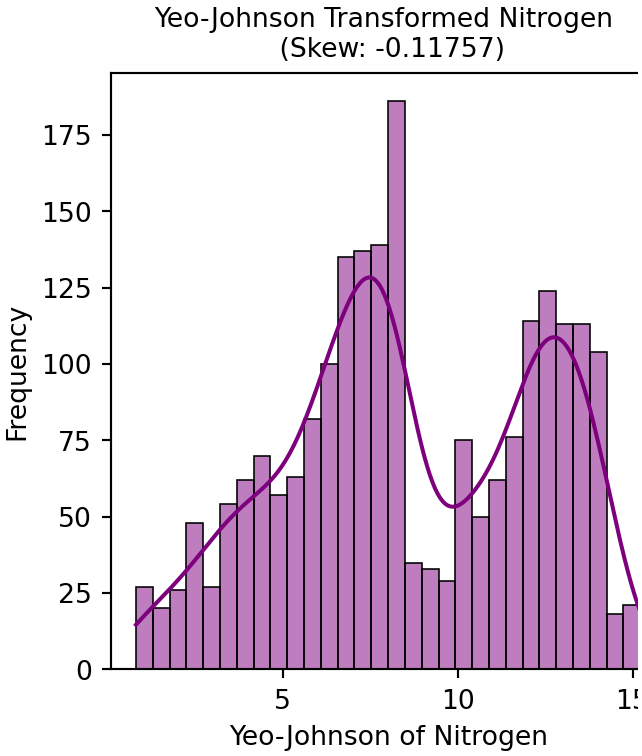
▶ Code

```
Values of Nitrogen were shifted to make them positive.
```

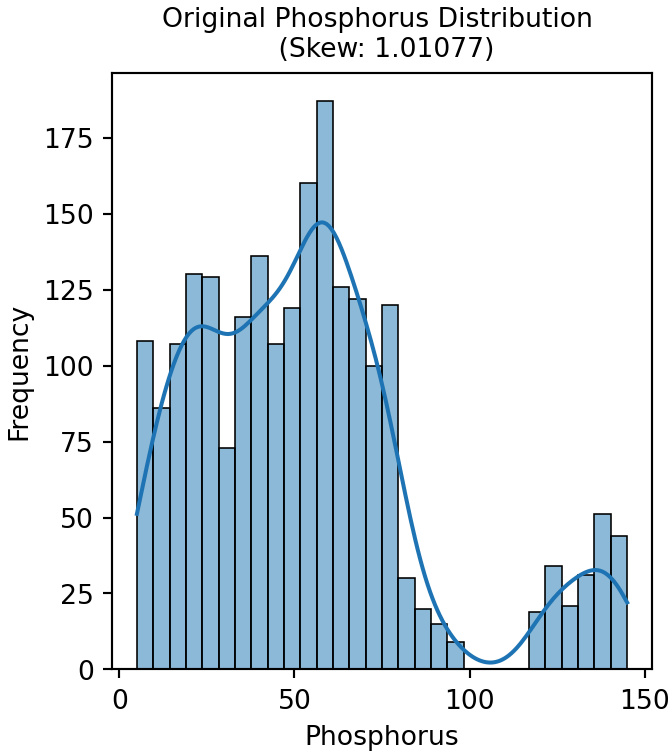

Original Nitrogen Distribution
(Skew: 0.50972)
Frequency
Nitrogen
0
25
50
75
100
125
150
175
0
50
100

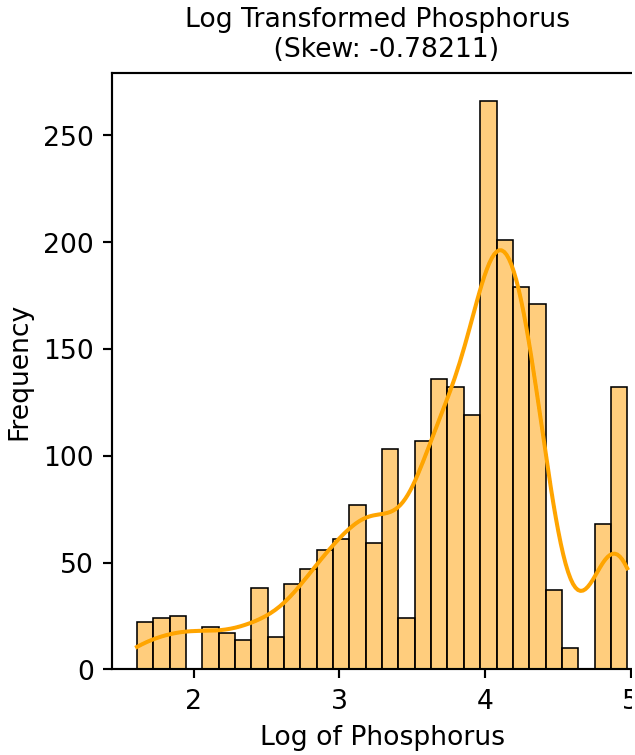

Log Transformed Nitrogen
(Skew: nan)
Frequency
Log of Nitrogen
0
50
100
150
200
250
0
1
2
3
4

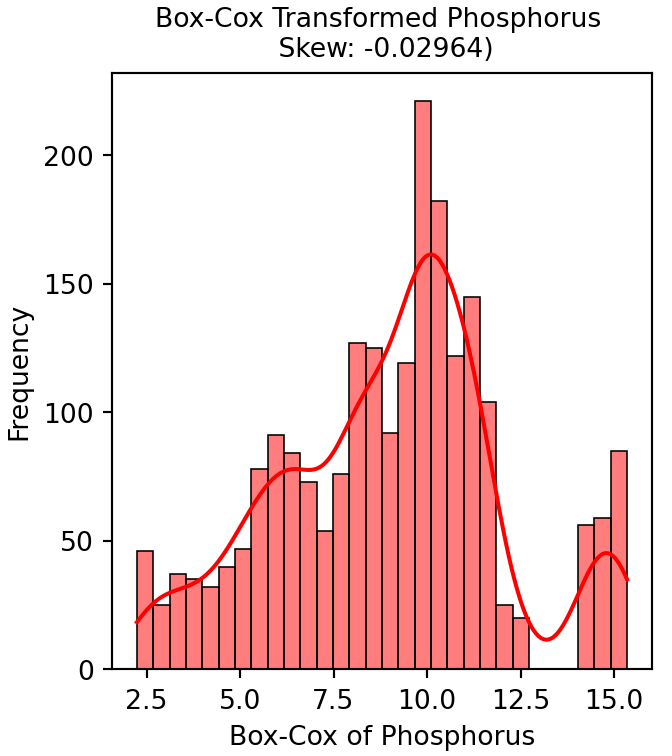

Box-Cox Transformed Nitrogen
Skew: -0.12974)
Frequency
Box-Cox of Nitrogen
0
25
50
75
100
125
150
175
0
5
10
15

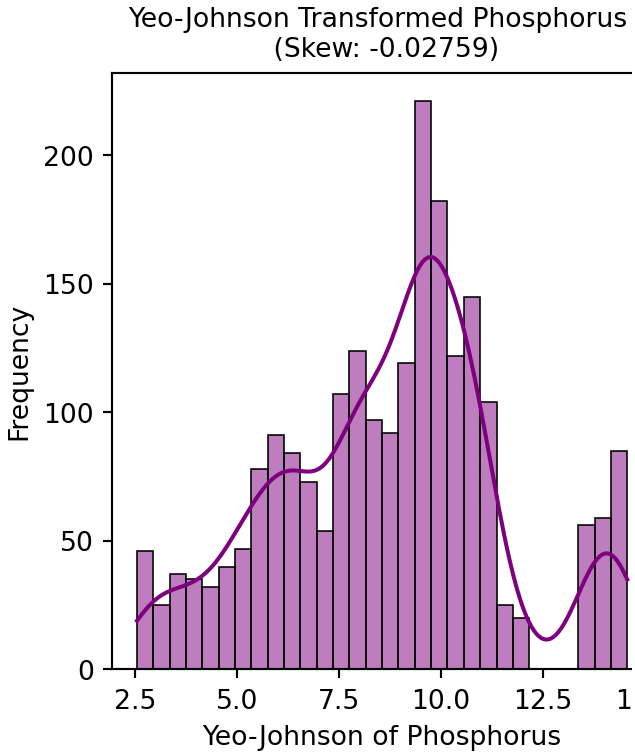

Yeo-Johnson Transformed Nitrogen
(Skew: -0.11757)
Frequency
Yeo-Johnson of Nitrogen
0
25
50
75
100
125
150
175
5
10

▶ Code

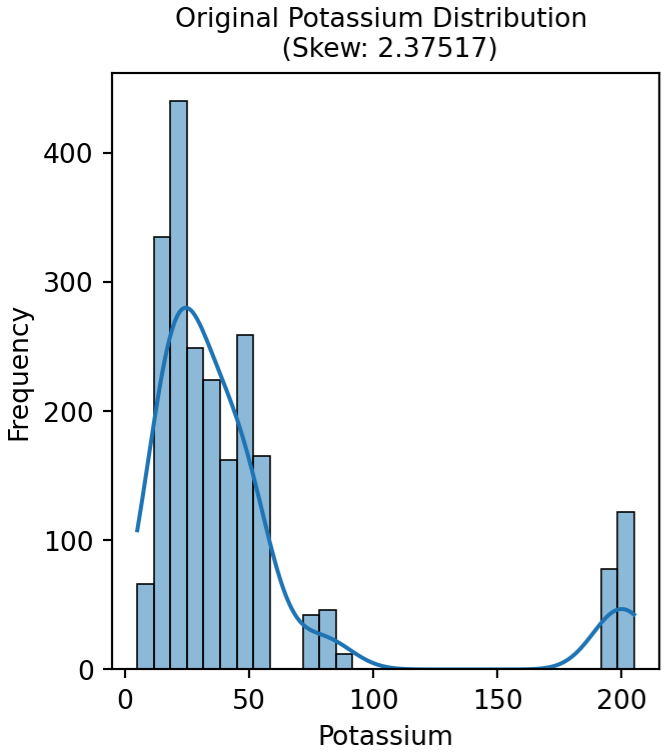

Original Phosphorus Distribution
(Skew: 1.01077)
Frequency
Phosphorus
0
25
50
75
100
125
150
175
0
50
100
150

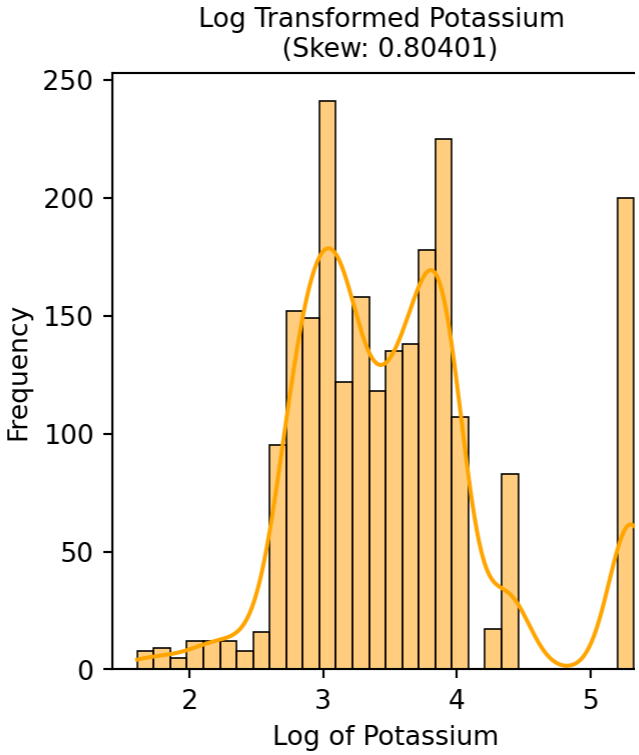

Log Transformed Phosphorus
(Skew: -0.78211)
Frequency
Log of Phosphorus
0
50
100
150
200
250
2
3
4

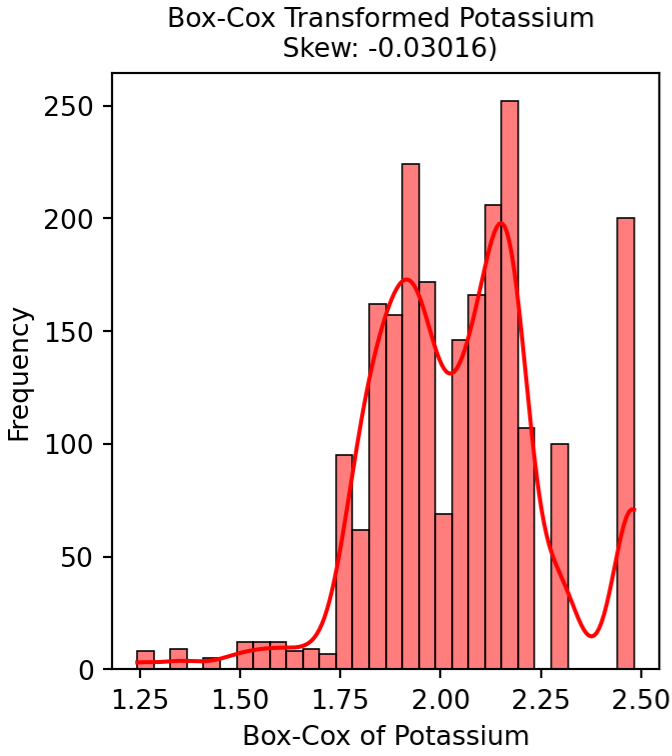

Box-Cox Transformed Phosphorus
Skew: -0.02964)
Frequency
Box-Cox of Phosphorus
0
50
100
150
200
2.5
5.0
7.5
10.0
12.5
15.0

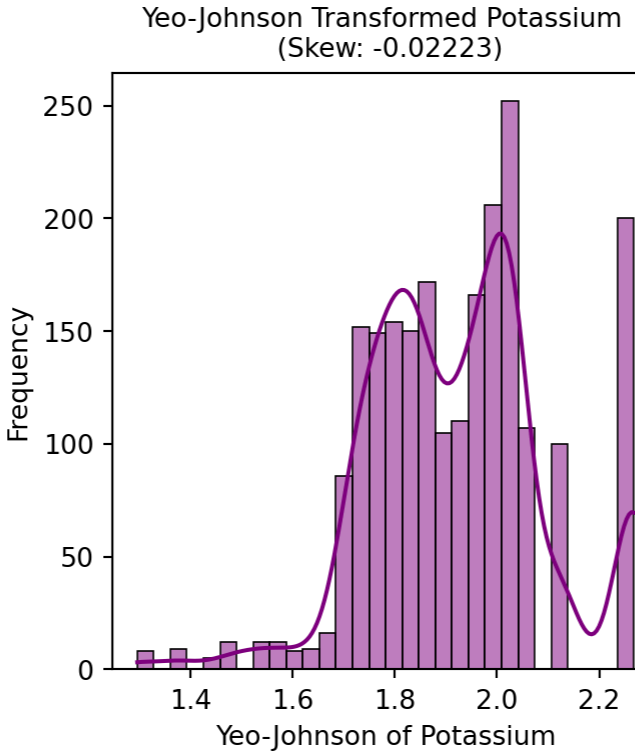

Yeo-Johnson Transformed Phosphorus
(Skew: -0.02759)
Frequency
Yeo-Johnson of Phosphorus
0
50
100
150
200
2.5
5.0
7.5
10.0
12.5

▶ Code

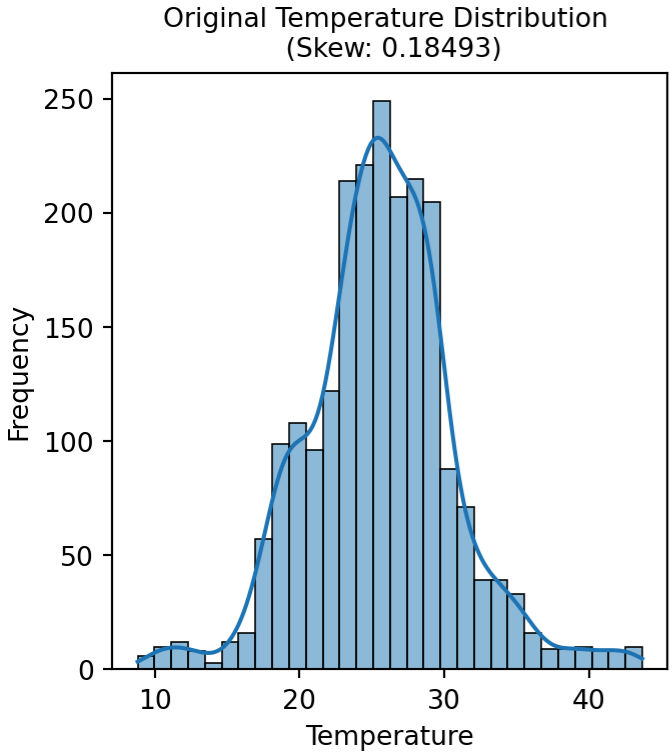

Original Potassium Distribution
(Skew: 2.37517)
Frequency
Potassium
0
100
200
300
400
0
50
100
150
200

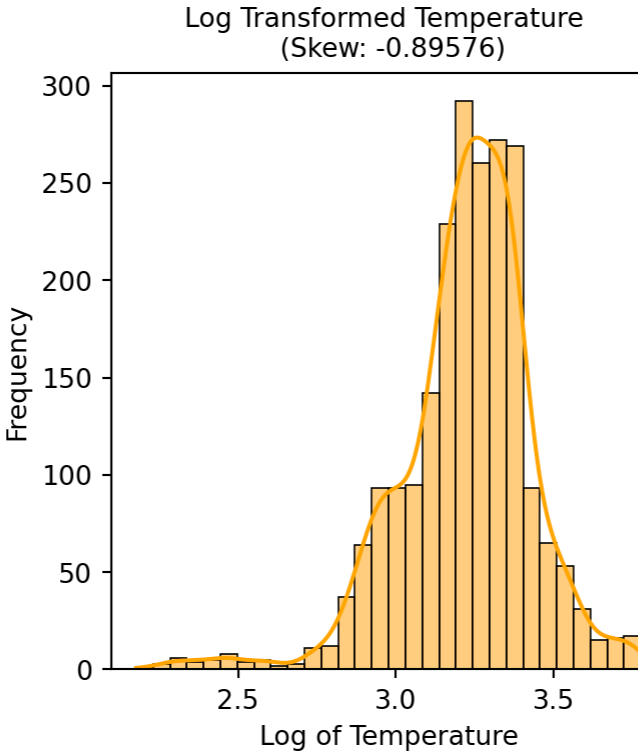

Log Transformed Potassium
(Skew: 0.80401)
Frequency
Log of Potassium
0
50
100
150
200
250
2
3
4
5

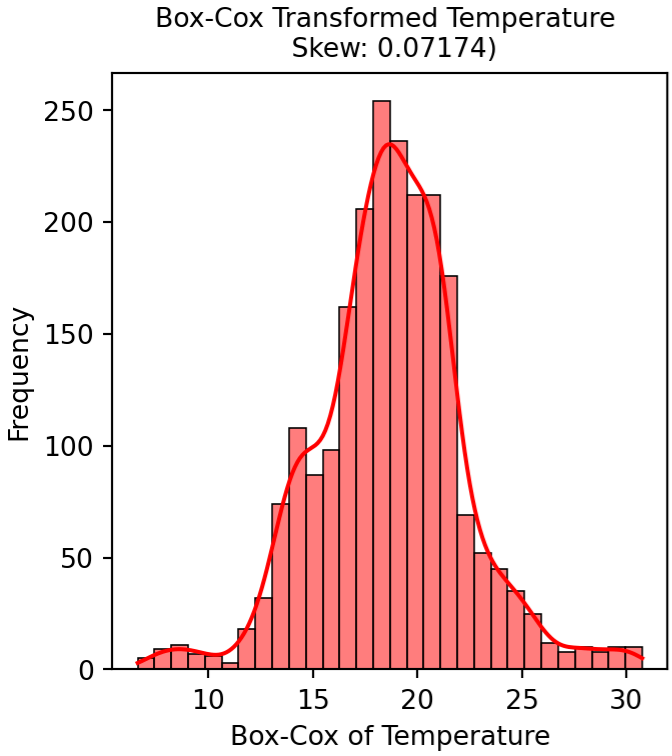

Box-Cox Transformed Potassium
Skew: -0.03016)
Frequency
Box-Cox of Potassium
0
50
100
150
200
250
1.25
1.50
1.75
2.00
2.25
2.50

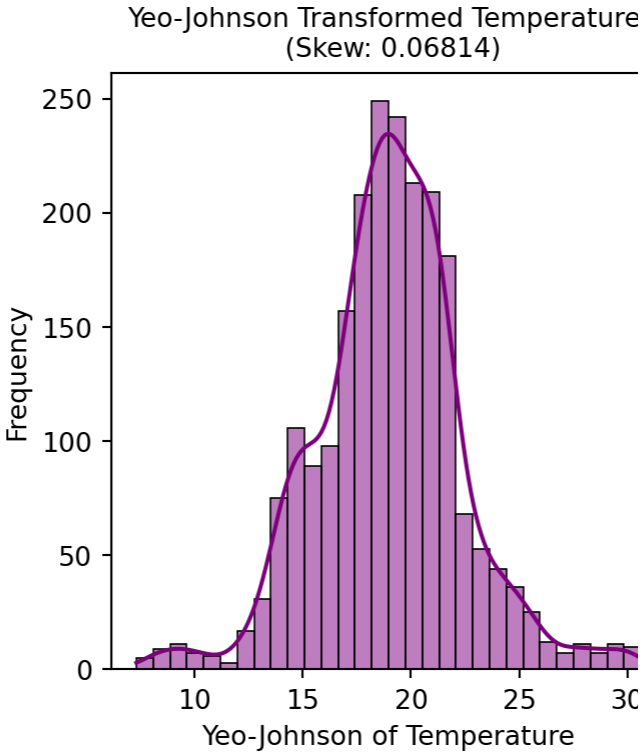

Yeo-Johnson Transformed Potassium
(Skew: -0.02223)
Frequency
Yeo-Johnson of Potassium
0
50
100
150
200
250
1.4
1.6
1.8
2.0
2.2

▶ Code

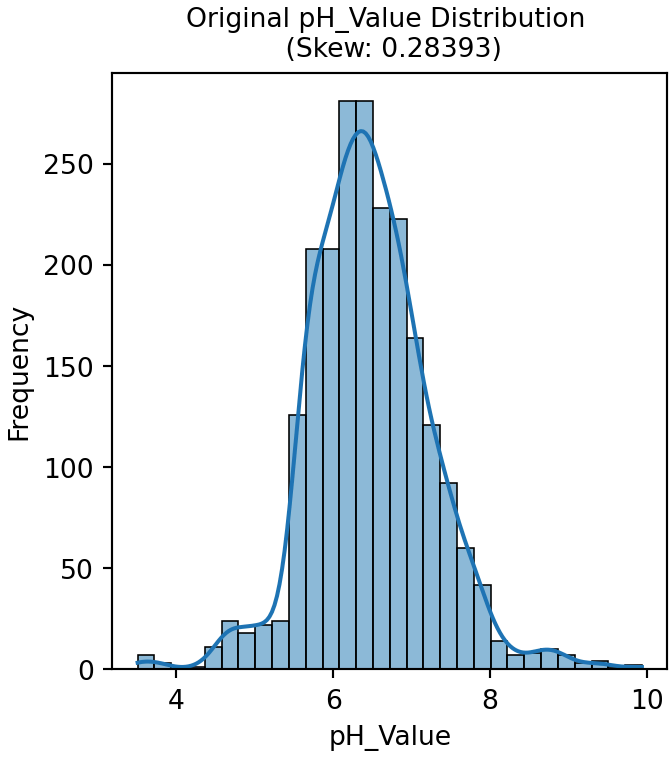

Original Temperature Distribution
(Skew: 0.18493)
Frequency
Temperature
0
50
100
150
200
250
10
20
30
40

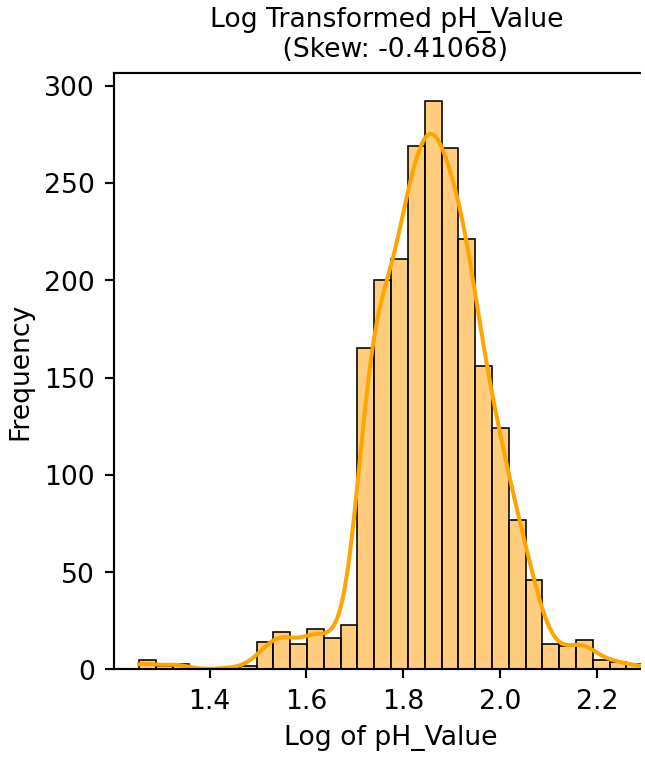

Log Transformed Temperature
(Skew: -0.89576)
Frequency
Log of Temperature
0
50
100
150
200
250
300
2.5
3.0
3.5

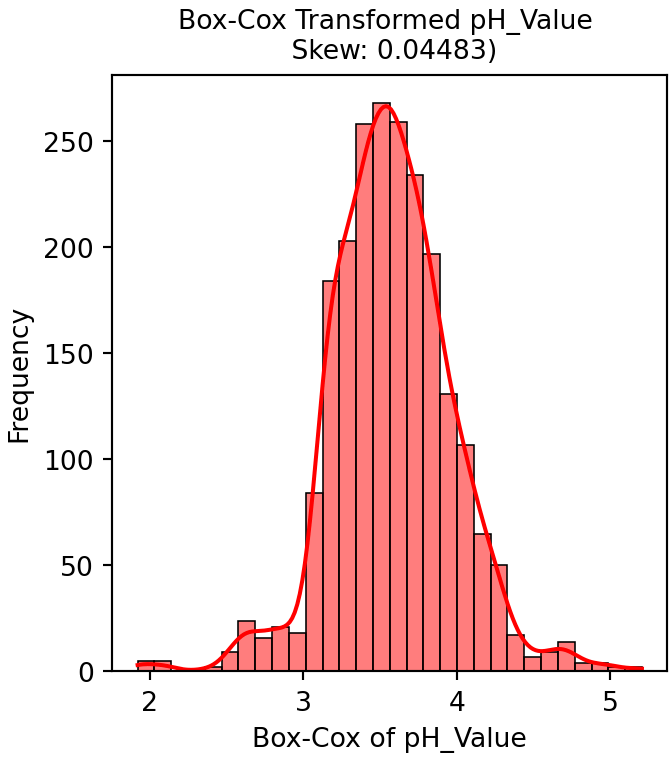

Box-Cox Transformed Temperature
Skew: 0.07174)
Frequency
Box-Cox of Temperature
0
50
100
150
200
250
10
15
20
25
30

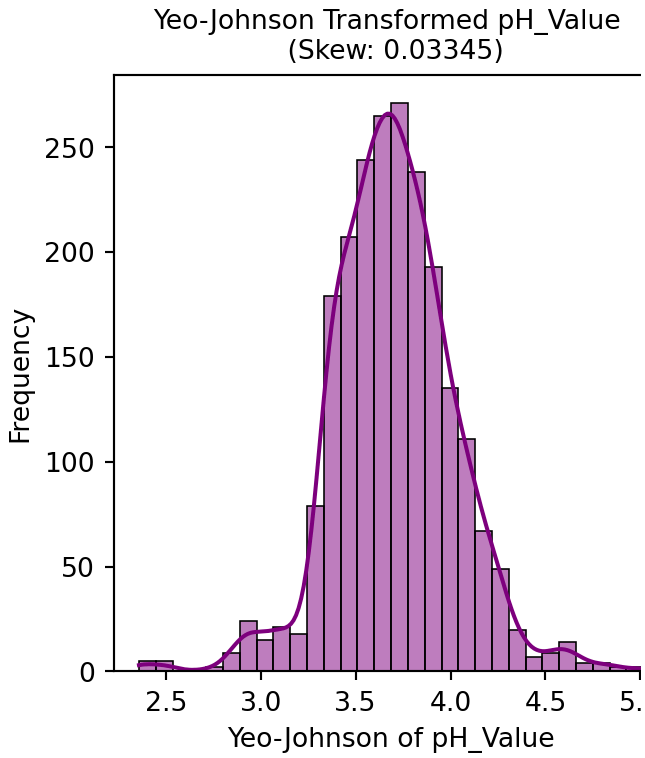

Yeo-Johnson Transformed Temperature
(Skew: 0.06814)
Frequency
Yeo-Johnson of Temperature
0
50
100
150
200
250
10
15
20
25
30

▶ Code





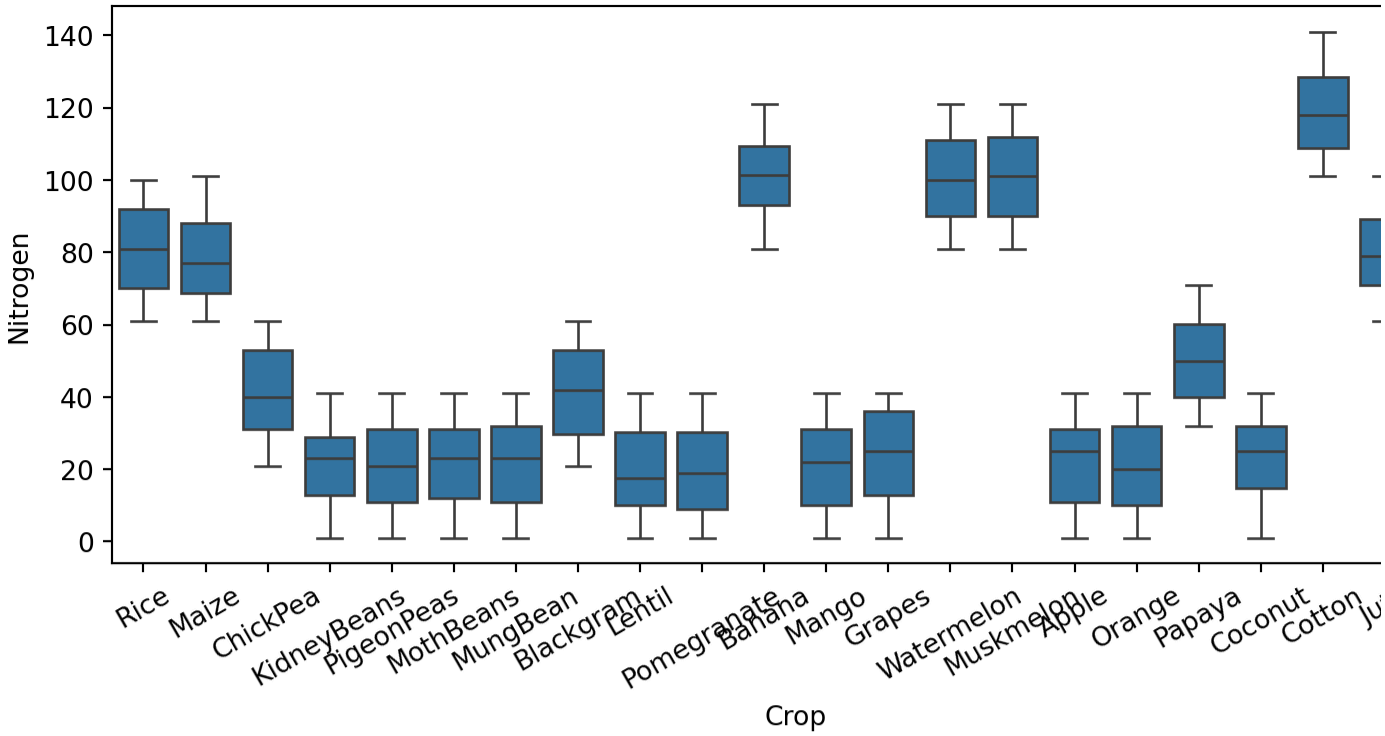


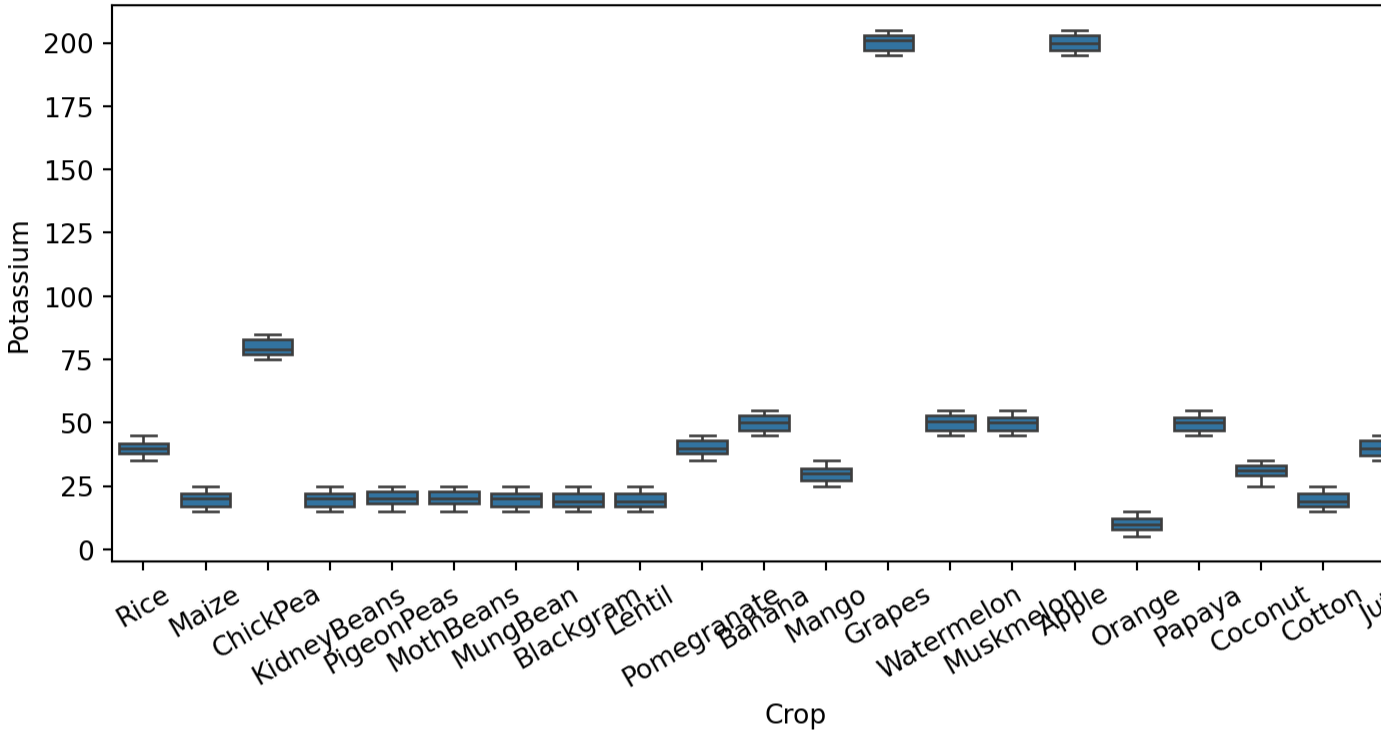


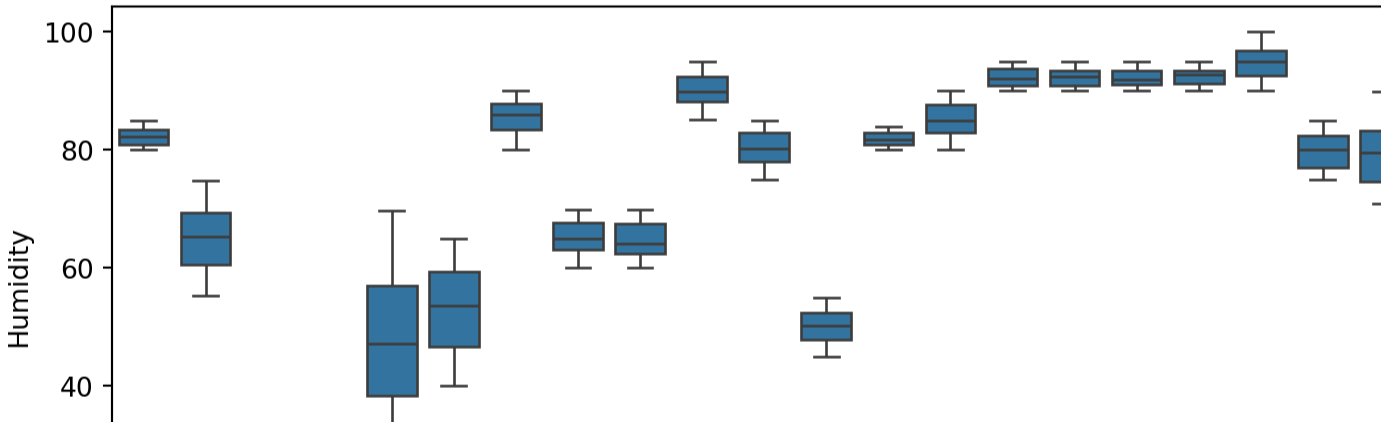
It is evident that the quantile transformation graph looks the best so we'll use quantile transformation on the needed data.

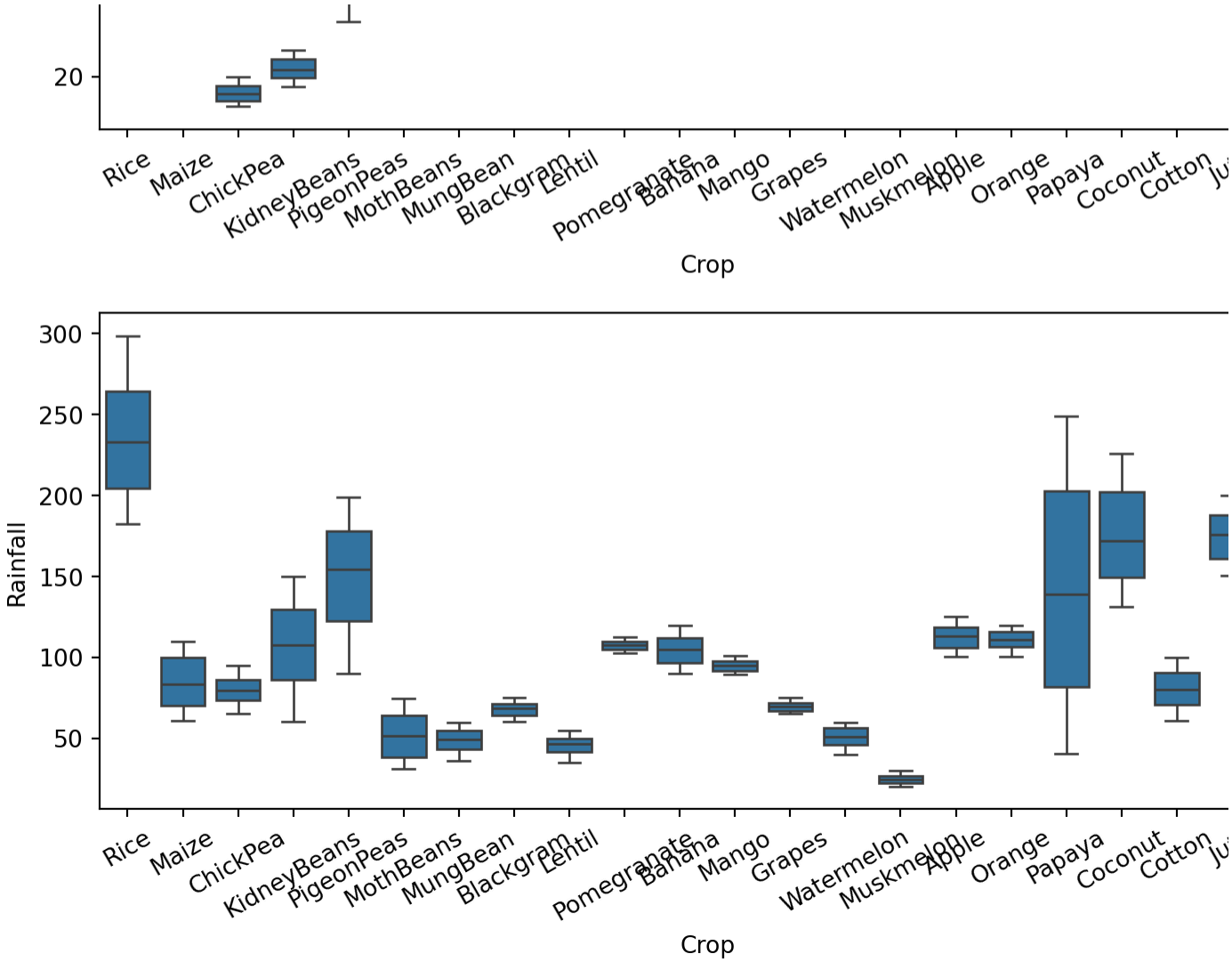
However in some of the graphs above, we see that some transformations result in double peaks. This might indicate that crops might be devided in two groups for each feature. To understand this assumption of ours, we can graphs that show crops vs features, i.e. how different crops behave to different values of features.

▶ Code


Nitrogen
0
20
40
60
80
100
120
140
Rice
Maize
ChickPea
KidneyBeans
PigeonPeas
MothBeans
MungBean
Blackgram
Lentil
Pomegranate
Banana
Mango
Grapes
Watermelon
Muskmelon
Apple
Orange
Papaya
Coconut
Cotton
Crop


Potassium
0
25
50
75
100
125
150
175
200
Rice
Maize
ChickPea
KidneyBeans
PigeonPeas
MothBeans
MungBean
Blackgram
Lentil
Pomegranate
Banana
Mango
Grapes
Watermelon
Muskmelon
Apple
Orange
Papaya
Coconut
Cotton
Crop


Humidity
40
60
80
100



From the above box plots it is now clear to us that crops are be divided into to groups for certain features.

For example, the Box-Cox transformation and Yeo-Johnson transformation for Nitrogen shows two peaks. This is further confirmed from the box plot above where almost half of the crops reuire low nitrogen and the other half requires high nitrogen.

This is also the case for certain other features like Potassium and Phosphorus.

Label Encoding

▶ Code

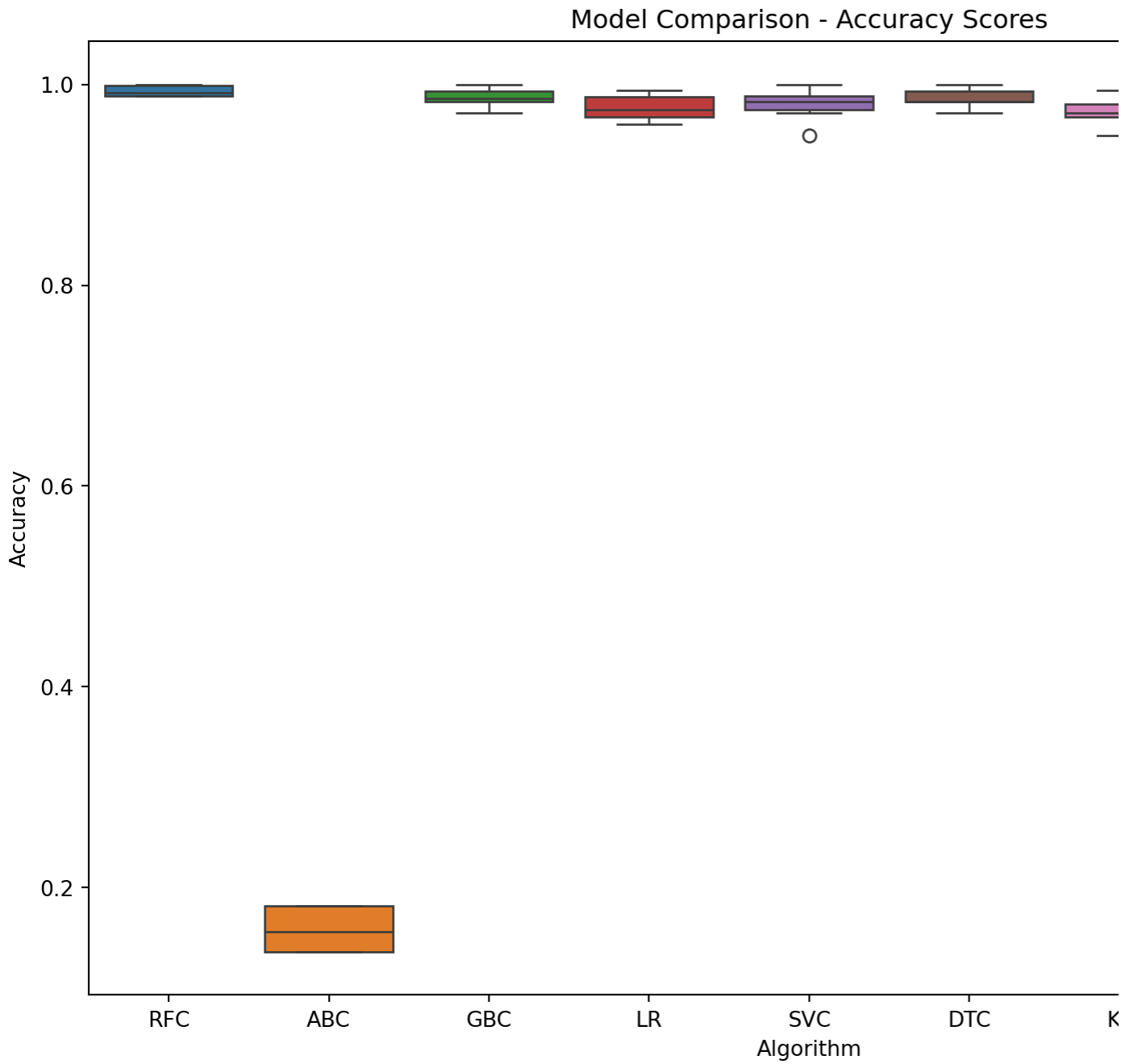
```
Encoded Value: 0, Original Label: Apple
Encoded Value: 1, Original Label: Banana
Encoded Value: 2, Original Label: Blackgram
Encoded Value: 3, Original Label: ChickPea
Encoded Value: 4, Original Label: Coconut
Encoded Value: 5, Original Label: Coffee
Encoded Value: 6, Original Label: Cotton
Encoded Value: 7, Original Label: Grapes
```

```
Encoded Value: 8, Original Label: Jute
Encoded Value: 9, Original Label: KidneyBeans
Encoded Value: 10, Original Label: Lentil
Encoded Value: 11, Original Label: Maize
Encoded Value: 12, Original Label: Mango
Encoded Value: 13, Original Label: MothBeans
Encoded Value: 14, Original Label: MungBean
Encoded Value: 15, Original Label: Muskmelon
Encoded Value: 16, Original Label: Orange
Encoded Value: 17, Original Label: Papaya
Encoded Value: 18, Original Label: PigeonPeas
Encoded Value: 19, Original Label: Pomegranate
Encoded Value: 20, Original Label: Rice
Encoded Value: 21, Original Label: Watermelon
```

Model comparison:

▶ Code

```
           Acc Mean  Acc STD
Algorithm
RFC           0.993    0.005
ABC           0.159    0.022
GBC           0.987    0.009
LR            0.976    0.012
SVC           0.982    0.014
DTC           0.986    0.008
KNN           0.972    0.013
GNB           0.994    0.005
XGB           0.929    0.018
results_acc_df:       0     1     2     3     4     5     6     7     8
0 0.989 0.136 0.972 0.972 0.983 0.983 0.966 0.994 0.915
1 0.994 0.136 0.983 0.989 0.989 0.994 0.972 0.989 0.938
2 1.000 0.182 0.989 0.989 1.000 0.983 0.994 1.000 0.949
3 0.994 0.182 0.983 0.994 0.983 0.983 0.972 0.994 0.955
4 1.000 0.182 1.000 0.983 0.989 0.989 0.972 1.000 0.949
5 0.989 0.136 0.989 0.966 1.000 0.994 0.983 0.989 0.920
6 0.989 0.136 0.977 0.960 0.983 0.983 0.983 0.994 0.926
7 1.000 0.182 1.000 0.972 0.972 1.000 0.972 1.000 0.903
8 0.989 0.176 0.994 0.977 0.972 0.972 0.955 0.989 0.903
9 0.989 0.136 0.983 0.960 0.949 0.977 0.949 0.989 0.932
```

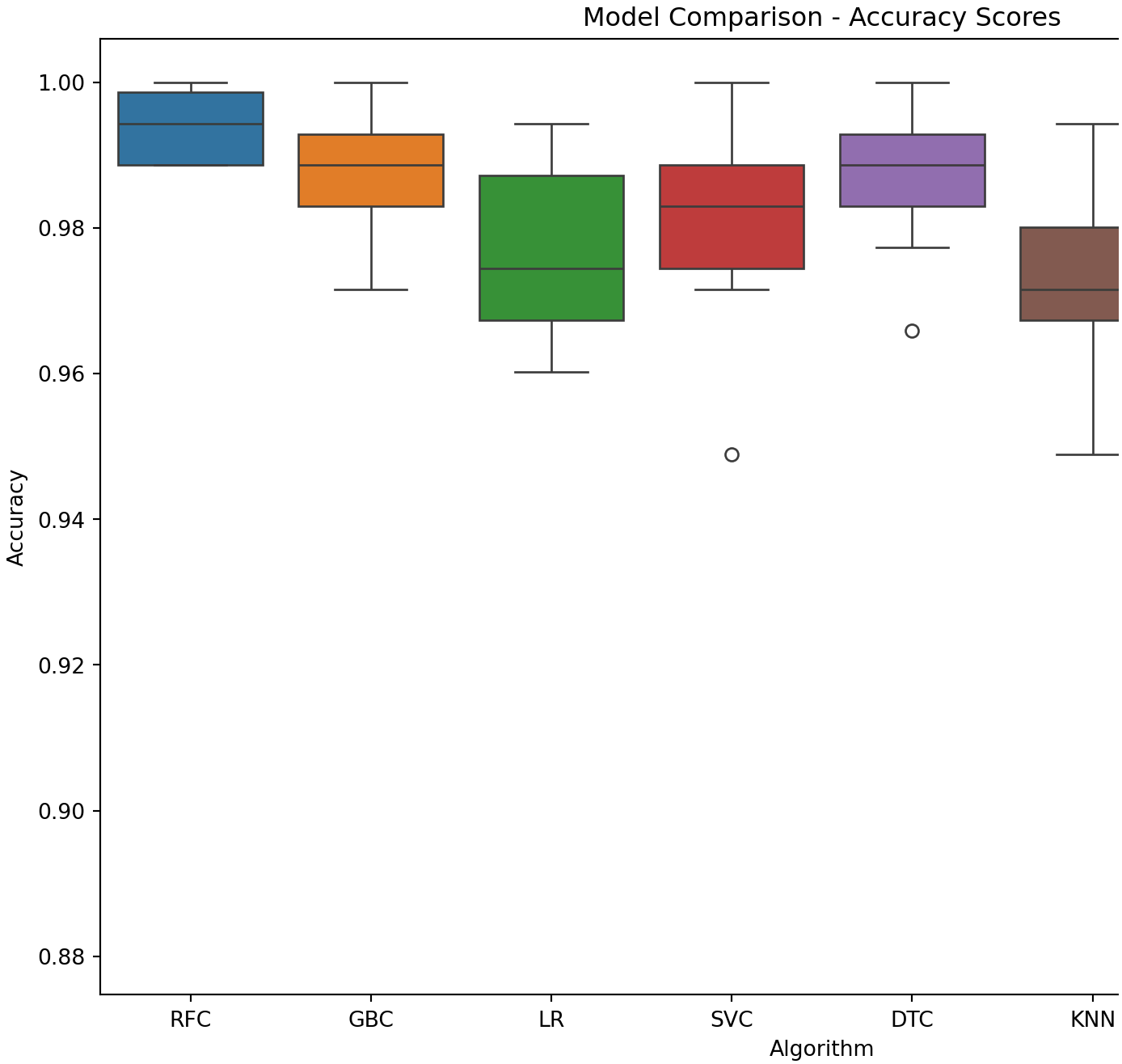


We can clearly discard Ada Boost Classifier.

▶ Code

Algorithm	Acc Mean	Acc STD
RFC	0.994	0.005
GBC	0.988	0.009
LR	0.976	0.012
SVC	0.982	0.014
DTC	0.986	0.009
KNN	0.972	0.013
GNB	0.994	0.005
XGB	0.922	0.022

```
results_acc_df:       0     1     2     3     4     5     6     7
0 0.989 0.972 0.972 0.983 0.989 0.966 0.994 0.903
1 0.994 0.989 0.989 0.989 0.994 0.972 0.989 0.938
2 1.000 0.989 0.989 1.000 0.989 0.994 1.000 0.915
3 0.994 0.983 0.994 0.983 0.983 0.972 0.994 0.949
4 1.000 1.000 0.983 0.989 0.989 0.972 1.000 0.955
5 0.989 0.989 0.966 1.000 0.994 0.983 0.989 0.903
6 0.989 0.977 0.960 0.983 0.983 0.983 0.994 0.943
7 1.000 1.000 0.972 0.972 1.000 0.972 1.000 0.920
8 0.989 0.994 0.977 0.972 0.966 0.955 0.989 0.881
9 0.994 0.983 0.960 0.949 0.977 0.949 0.989 0.915
```



After analyzing the above output, we can finalize Random Forets classifier model because of high accuracy and large size of our dataset.

Finalizing model:

▶ Code

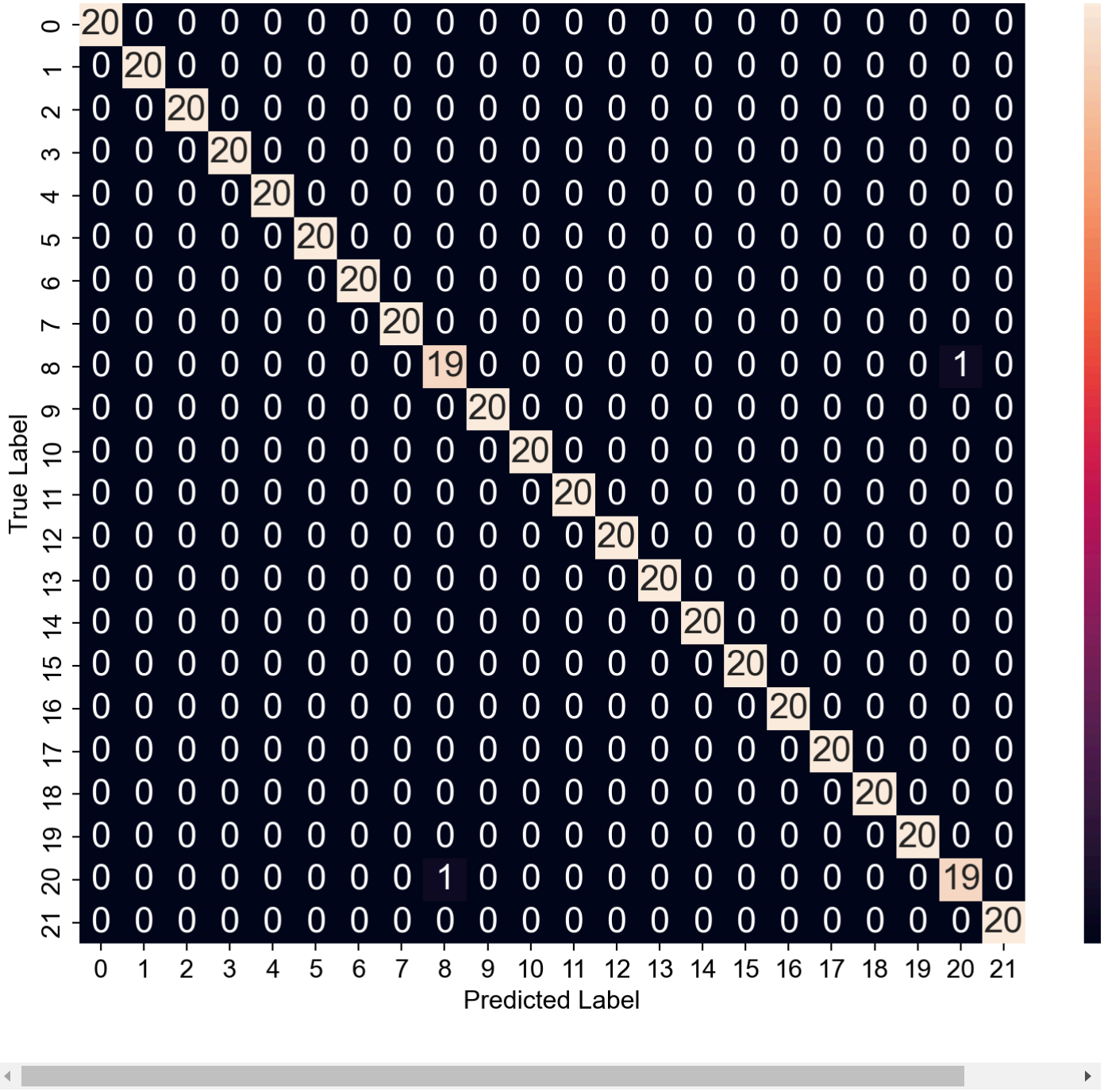
RandomForestClassifier

```
RandomForestClassifier()
```

▶ Code

	precision	recall	f1-score	support
0	1.00	1.00	1.00	20
1	1.00	1.00	1.00	20
2	1.00	1.00	1.00	20
3	1.00	1.00	1.00	20
4	1.00	1.00	1.00	20
5	1.00	1.00	1.00	20
6	1.00	1.00	1.00	20
7	1.00	1.00	1.00	20
8	0.95	0.95	0.95	20
9	1.00	1.00	1.00	20
10	1.00	1.00	1.00	20
11	1.00	1.00	1.00	20
12	1.00	1.00	1.00	20
13	1.00	1.00	1.00	20
14	1.00	1.00	1.00	20
15	1.00	1.00	1.00	20
16	1.00	1.00	1.00	20
17	1.00	1.00	1.00	20
18	1.00	1.00	1.00	20
19	1.00	1.00	1.00	20
20	0.95	0.95	0.95	20
21	1.00	1.00	1.00	20
accuracy			1.00	440
macro avg	1.00	1.00	1.00	440
weighted avg	1.00	1.00	1.00	440

▶ Code



Conclusion:

The primary objective of this analysis was to develop a robust predictive model for crop prediction. The Random Forest model achieved an accuracy of 99.4%. This results indicate that the model is effective in predicting crop yields. The model's performance was validated using cross-validation techniques, yielding consistent results across different folds.

Compared to other models such as Logistic Regression and Decision Trees, the Random Forest model demonstrated superior performance.

The implementation of this Random Forest model can significantly improve agricultural planning and decision-making, leading to better resource allocation. For instance, the model can help in accurately predicting crop yields, thereby enhancing operational efficiency. In conclusion, the Random Forest model developed in this analysis provides a powerful tool for predicting crop yields, with substantial potential for practical application in agriculture.

An app has also been developed which takes user input for features and then predicts the top five crops. The app also allows us to validate the model chosen by generating random values from the test dataset.