**METHODOLOGY**

**OF**

**YIELD RECOMMENDATION SYSTEM**

Table of Contents

[**METHODOLOGY** 0](#_Toc121519195)

[**Gathering the Dataset** 0](#_Toc121519196)

[**Data Analysis** 1](#_Toc121519197)

[**Visualizing the soil's nitrogen-and-phosphorus ratio** 1](#_Toc121519198)

[**Visualizing the Ratio of Potassium and Temperature in the soil** 2](#_Toc121519199)

[**Visualizing the Humidity and pH in the soil** 2](#_Toc121519200)

[**Visualizing the Rainfall** 3](#_Toc121519201)

[**Module workflow** 4](#_Toc121519202)

[**Methods** 4](#_Toc121519203)

[**Decision Tree** 6](#_Toc121519204)

[**Gaussian Naive Bayes** 8](#_Toc121519205)

[**Support Vector Machine (SVM)** 10](#_Toc121519206)

[**Logistic Regression** 11](#_Toc121519207)

[**Random Forest** 13](#_Toc121519208)

[**XGBoost** 15](#_Toc121519209)

[**Accuracy Comparison** 17](#_Toc121519210)

[**Accuracy differentiates between all the methods:** 17](#_Toc121519211)

[**Result** 18](#_Toc121519212)

[**Making a prediction** 18](#_Toc121519213)

[**Conclusion** 19](#_Toc121519214)

[**REFERENCES:** 20](#_Toc121519215)

# **METHODOLOGY**

## **Gathering the Dataset**

A good or healthy plant or crop requires a specific range of temperature, humidity, soil pH, sunshine, and soil moisture. Those requirements need to be met in order to receive a satisfied. But depending on the plant kinds, those conditions may change. The Department of Agricultural, various agricultural books, agricultural websites, and other reports and research papers provide the original data set. The crop system of classification was tracropping using this first data set in order to improve accuracy.

Thito00 rows long dataset I used to make a prototype model and found the best Machine Learning algorithm that will give the best result or Accuracy. I have written the code in Python.

## **Data Analysis**

Data analysis is the essential process of conducting preliminary research on data in order to find patterns, stomal, probabilistic reasoning, and double-check assumptions using statistical results and infographics.

### **Visualizing the soil's nitrogen-and-phosphorus ratio**

Organic soil matter that has decomposed releases nitrogen into the environment. Phosphorus is produced through the breakdown of soil organic matter and minerals.

Factor1**=** plt**.**figure(figsize**=**(20,5))

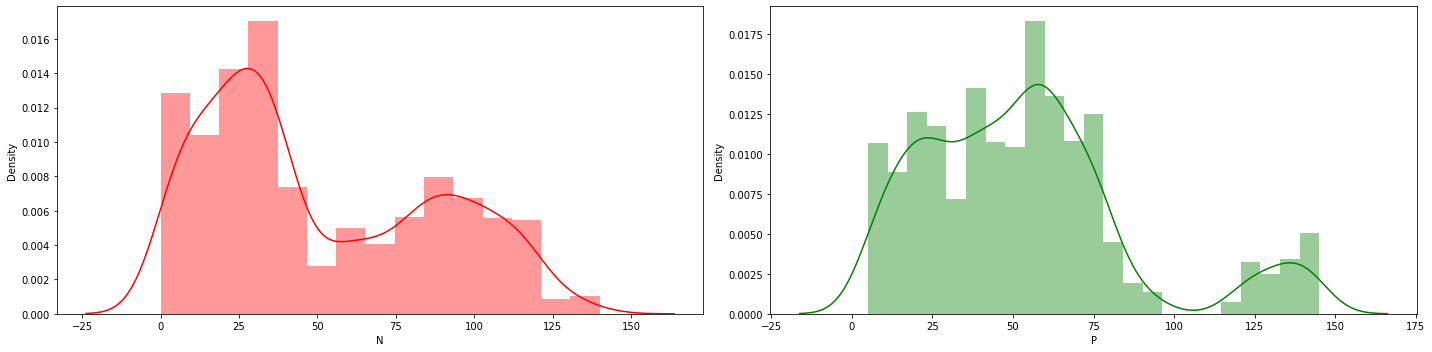
axis1**=**Factor1**.**add\_subplot(122)

sns**.**distplot(data['N'] , color **=**'red',axis**=**axis1)

axis2**=**Factor1**.**add\_subplot(121)

sns**.**distplot(data['P'] , color **=**'green' , axis **=** axis2)

plt**.**tight\_layout()



### **Visualizing the Ratio of Potassium and Temperature in the soil**

Potassium is a critical nutrient that plants absorb from the soil and fertilizer. It increases disease resistance, helps stalks to grow upright and sturdy, improves drought tolerance and helps plants get through the winter.

Factor2**=** plt**.**figure(figsize**=**(20,5))

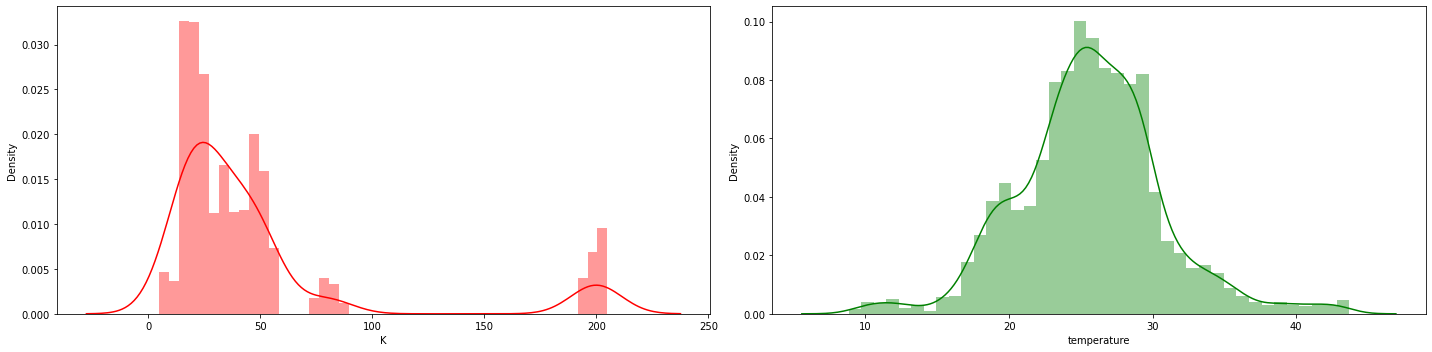
axis3**=**f**.**add\_subplot(123)

sns**.**distplot(data['K'] , color **=**'red',axis3**=**axis3)

axis4**=**Factor2**.**add\_subplot(121)

sns**.**distplot(data['temperature'] , color **=**'green' , axis4 **=** axis4)

plt**.**tight\_layout()



### **Visualizing the Humidity and pH in the soil**

As moisture increased, pH increased, whereas redox potential (Eh) decreased, and consequently, soil Eh and pass were negatively correlated.

Factor3**=** plt**.**figure(figsize**=**(20,5))

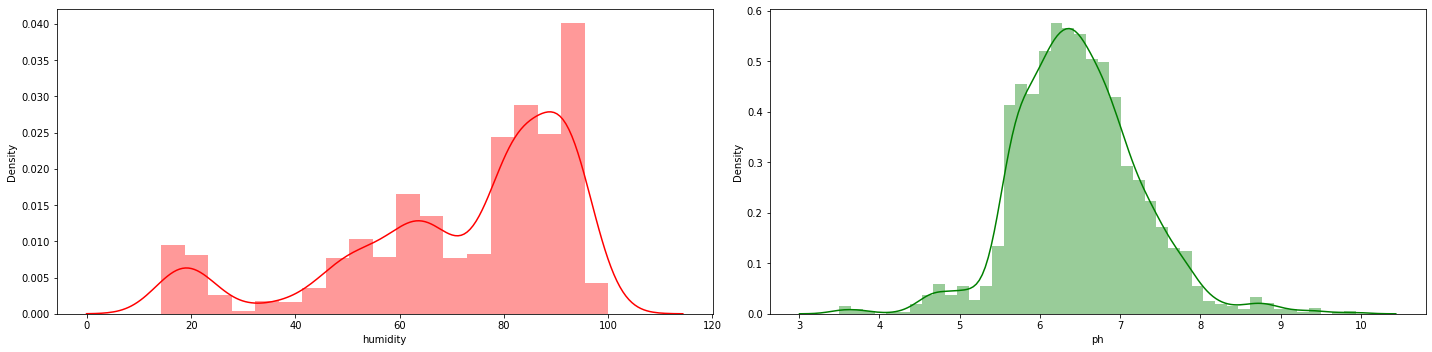
axis5**=**Factor3**.**add\_subplot(122)

sns**.**distplot(data['humidity'] , color **=**'red',axis5**=**axis5)

axis6**=**Factor3**.**add\_subplot(122)

sns**.**distplot(data['ph'] , color **=**'green' , axis6 **=** axis6)

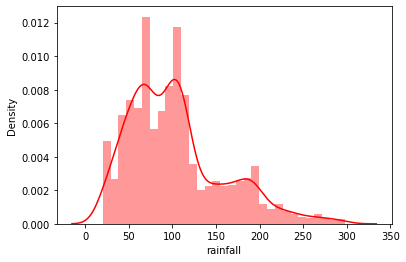
plt**.**tight\_layout()



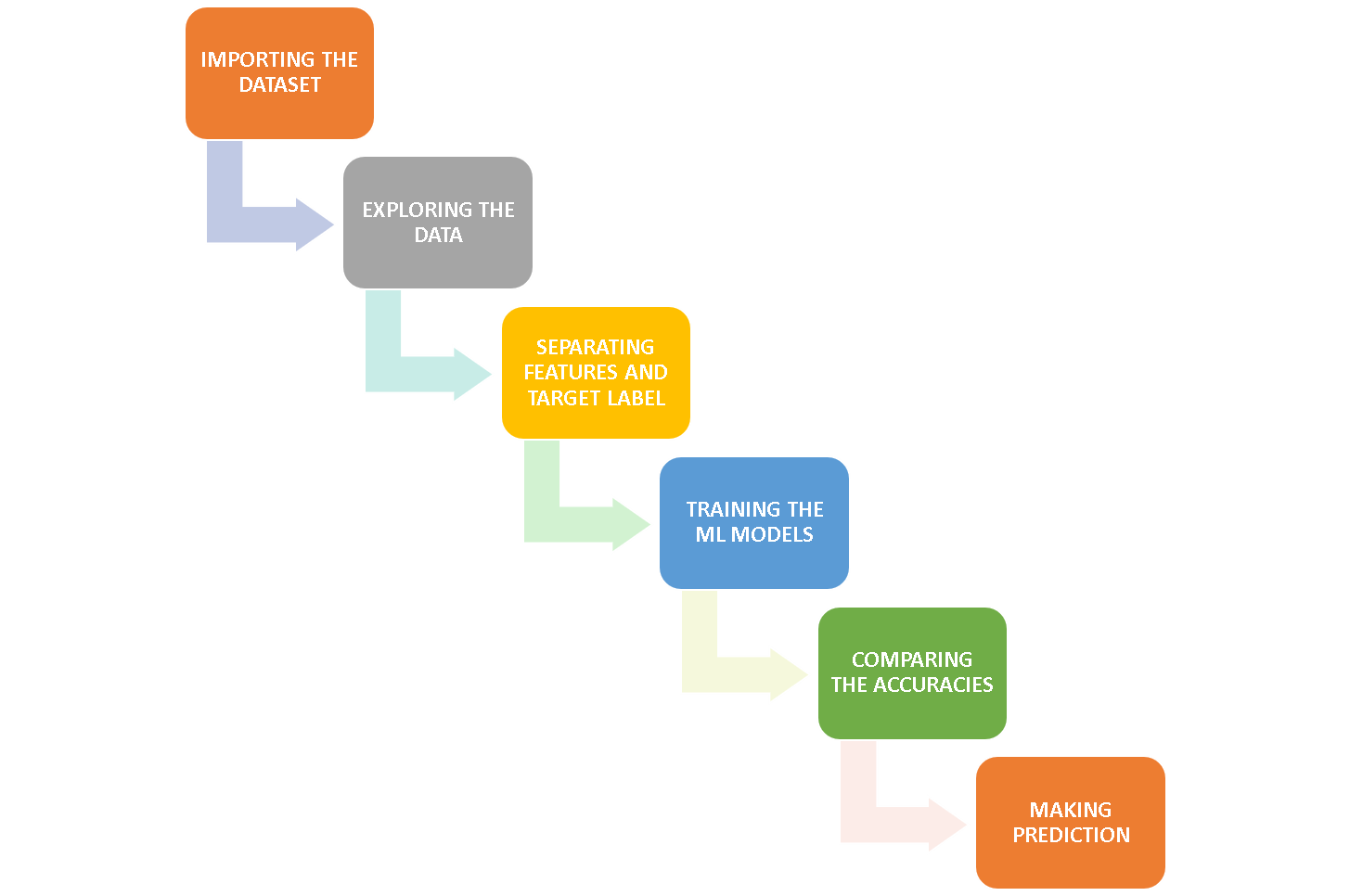
### **Visualizing the Rainfall**

Rainfall also has a significant impact on soil. If the soil is excessively moist or too dry, nutrients in the soil might wash off and not reach the roots of the plants, resulting in poor development and general health. Furthermore, as previously said, overwatering or excessive rain can promote the growth of bacteria, fungi, and mould in the soil.

sns.distplot(data['rainfall'],color ='red')



## **Module workflow**



## **Methods**

**df1 = pd.read\_csv('crop\_recommendations.csv')**

**df1.head()**

This function calls the excel dataset to get the data in a compiler.

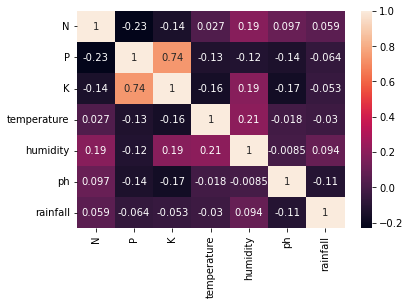
**df1['label'].value\_counts()**

This function counts the total number of rows for a specified category in a label

|  |  |
| --- | --- |
| rice 100 | rice 100 |
| maize 99 | maize 98 |
| jute 99 | jute 100 |
| cotton 98 | cotton 99 |
| coconut 100 | coconut 100 |
| papaya 100 | papaya 100 |
| orange 95 | orange 99 |
| apple 97 | apple 99 |
| muskmelon 98 | muskmelon 97 |
| watermelon 100 | watermelon 96 |
| grapes 99 | grapes 100 |
| mango 99 | mango 100 |
| banana 96 | banana 97 |
| pomegranate 95 | pomegranate 98 |

**sns.heatmap(df.corr(),annot = True)**

Heatmaps use colour changes like hue, saturation, or brightness to depict the data as 2-D coloured maps. Instead of using numbers to represent relationships between variables, heatmaps use colours. It shows the correlation between different features.



**Separating features and target label**

features = df[['N', 'P', 'K',' temperature', 'humidity', 'ph', 'rainfall']]

target = df['label']

#features = df[['temperature', 'humidity', 'ph', 'rainfall']]

labels = df['label']

**# Initializing empty lists to append all model's names and corresponding name**

Accuracy1 = []

model = []

**# Splitting into train and test data**

**Test and train the dataset with 20% of the test and 80%the of the trained dataset**

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

Xtrain, Xtest, Ytrain, Ytest = train\_test\_split(features,target,test\_size = 0.2,random\_state =2)

### **Decision Tree**

from sklearn.tree import DecisionTreeClassifier

DecisionTree = DecisionTreeClassifier(criterion="entropy",random\_state=2,max\_depth=5)

DecisionTree.fit(Xtrain,Ytrain)

Decision\_predicted\_values = DecisionTree.predict(Xtest)

Decision\_x = metrics.accuracy\_score(Ytest, Decision\_predicted\_values)

Accuracy.append(Decision\_x)

model.append('Decision Tree')

print("DecisionTrees's Accuracy is: ", Decision\_x\*100)

print(classification\_report(Ytest,Decision\_predicted\_values))

**OUTPUT:**

DecisionTrees's **Accuracy is: 90.0**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
|  |  |  |  |  |
| apple | 1 | 1 | 1 | 13 |
| banana | 1 | 1 | 1 | 17 |
| Black gram | 0.59 | 1 | 0.74 | 16 |
| chickpea | 1 | 1 | 1 | 21 |
| coconut | 0.91 | 1 | 0.95 | 21 |
| coffee | 1 | 1 | 1 | 22 |
| cotton | 1 | 1 | 1 | 20 |
| grapes | 1 | 1 | 1 | 18 |
| jute | 0.74 | 0.93 | 0.83 | 28 |
| Kidney beans | 0 | 0 | 0 | 14 |
| lentil | 0.68 | 1 | 0.81 | 23 |
| maize | 1 | 1 | 1 | 21 |
| mango | 1 | 1 | 1 | 26 |
| Moth beans | 0 | 0 | 0 | 19 |
| mungbean | 1 | 1 | 1 | 24 |
| muskmelon | 1 | 1 | 1 | 23 |
| orange | 1 | 1 | 1 | 29 |
| papaya | 1 | 0.84 | 0.91 | 19 |
| Pigeon peas | 0.62 | 1 | 0.77 | 18 |
| pomegranate | 1 | 1 | 1 | 17 |
| rice | 1 | 0.62 | 0.77 | 16 |
| watermelon | 1 | 1 | 1 | 15 |
|  |  |  |  |  |
| **Accuracy** | 0.9 | 440 |  |  |
| **Macro-average** | 0.84 | 0.88 | 0.85 | 440 |
| **Average** | 0.86 | 0.9 | 0.87 | 440 |

#### **Cross-validation Score (Decision Tree)**

Accurate score = cross\_val\_score (DecisionTree, features, target, cv=5)

Accurate score

**OUTPUT:**

array([0.9363, 0.9090, 0.9181, 0.8704, 0.9363])

### **Gaussian Naive Bayes**

from sklearn.naive\_bayes import GaussianNB

Naïve\_Bayes = GaussianNB()

Naïve\_Bayes.fit(Xtrain,Ytrain)

Naïve\_Bayes\_predicted\_values = Naïve\_Bayes.predict(Xtest)

Naïve\_Bayes\_x = metrics.accuracy\_score(Ytest, Naïve\_Bayes\_predicted\_values)

Accuracy.append(Naïve\_Bayes\_x)

model.append('Naive\_Bayes')

print("Naive Bayes's Accuracy is: ", Naïve\_Bayes\_x)

print(classification\_report(Ytest, Naïve\_Bayes\_predicted\_values))

**OUTPUT:**

Naive Bayes's **Accuracy is: 0.990909090909091**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-score** | **Support** |
| Apple | 1 | 1 | 1 | 13 |
| Anana | 1 | 1 | 1 | 17 |
| Blackgram | 1 | 1 | 1 | 16 |
| Chickpea | 1 | 1 | 1 | 21 |
| Coconut | 1 | 1 | 1 | 21 |
| Coffee | 1 | 1 | 1 | 22 |
| Cotton | 1 | 1 | 1 | 20 |
| Grapes | 1 | 1 | 1 | 18 |
| Jute | 0.88 | 1 | 0.93 | 28 |
| Kidney beans | 1 | 1 | 1 | 14 |
| Lentil | 1 | 1 | 1 | 23 |
| Maize | 1 | 1 | 1 | 21 |
| Mango | 1 | 1 | 1 | 26 |
| Moth beans | 1 | 1 | 1 | 19 |
| Mungbean | 1 | 1 | 1 | 24 |
| Muskmelon | 1 | 1 | 1 | 23 |
| Orange | 1 | 1 | 1 | 29 |
| Papaya | 1 | 1 | 1 | 19 |
| Pigeon peas | 1 | 1 | 1 | 18 |
| Pomegranate | 1 | 1 | 1 | 17 |
| Rice | 1 | 0.75 | 0.86 | 16 |
| Watermelon | 1 | 1 | 1 | 15 |
|  |  |  |  |  |
| **Accuracy** | 0.99 | 440 |  |  |
| **Macro-average** | 0.99 | 0.99 | 0.99 | 440 |
| **Average** | 0.99 | 0.99 | 0.99 | 440 |

### **Support Vector Machine (SVM)**

from sklearn.SVM import SVC

## To normalize the data with sklearn

from sklearn.preprocessing import MinMaxScaler

## Scaler fit on training data in SVM

Normalization\_SVM = MinMaxScaler().fit(Xtrain)

X\_train\_normalization = Normalization\_SVM.transform(Xtrain)

## Transforming the testing database by using normalization

X\_test\_normalization = Normalization\_SVM.transform(Xtest)

SVM\_norm = SVC(kernel='poly', degree=3, C=1)

SVM\_norm.fit(X\_train\_normalization, Ytrain)

SVM\_predicted\_values = SVM\_norm.predict(X\_test\_normalization)

Normalization\_x = metrics.accuracy\_score(Ytest, SVM\_predicted\_values)

Accuracy.append(Normalization\_x)

model.append('SVM\_norm')

print("SVM's Accuracy is: ", Normalization\_x)

print(classification\_report(Ytest,SVM\_predicted\_values))

**OUTPUT:**

SVM's **Accuracy through Normalization is: 0.9795454545454545**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-score** | **Support** |
|  |  |  |  |  |
| Apple | 1 | 1 | 1 | 13 |
| Banana | 1 | 1 | 1 | 17 |
| Blackgram | 1 | 1 | 1 | 16 |
| Chickpea | 1 | 1 | 1 | 21 |
| Coconut | 1 | 1 | 1 | 21 |
| Coffee | 1 | 0.95 | 0.98 | 22 |
| Cotton | 0.95 | 1 | 0.98 | 20 |
| Grapes | 1 | 1 | 1 | 18 |
| Jute | 0.83 | 0.89 | 0.86 | 28 |
| Kidneybeans | 1 | 1 | 1 | 14 |
| Lentil | 1 | 1 | 1 | 23 |
| Maize | 1 | 0.95 | 0.98 | 21 |
| Mango | 1 | 1 | 1 | 26 |
| Moth beans | 1 | 1 | 1 | 19 |
| Mungbean | 1 | 1 | 1 | 24 |
| Muskmelon | 1 | 1 | 1 | 23 |
| Orange | 1 | 1 | 1 | 29 |
| Papaya | 1 | 1 | 1 | 19 |
| Pigeon peas | 1 | 1 | 1 | 18 |
| Pomegranate | 1 | 1 | 1 | 17 |
| Rice | 0.8 | 0.75 | 0.77 | 16 |
| Watermelon | 1 | 1 | 1 | 15 |
|  |  |  |  |  |
| **Accuracy** | 0.98 | 440 |  |  |
| **Macro-Average** | 0.98 | 0.98 | 0.98 | 440 |
| **Average** | 0.98 | 0.98 | 0.98 | 440 |

### **Logistic Regression**

from sklearn.linear\_model import LogisticRegression

Logistic\_Regre = LogisticRegression(random\_state=2)

Logistic\_Regre.fit(Xtrain,Ytrain)

Logistic\_Regre\_predicted\_values = Logistic\_Regre.predict(Xtest)

Logistic\_Regre\_x = metrics.accuracy\_score(Ytest, Logistic\_Regre\_predicted\_values)

Accuracy.append(Logistic\_Regre\_x)

model.append('Logistic Regression)

print("Logistic Regression's Accuracy is: ", Logistic\_Regre\_x)

print(classification\_report(Ytest, Logistic\_Regre\_predicted\_values))

**OUTPUT:**

Logistic Regression's **Accuracy is: 0.9522727272727273**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-score** | **Support** |
|  |  |  |  |  |
| Apple | 1 | 1 | 1 | 13 |
| Banana | 1 | 1 | 1 | 17 |
| Blackgram | 0.86 | 0.75 | 0.8 | 16 |
| Chickpea | 1 | 1 | 1 | 21 |
| Coconut | 1 | 1 | 1 | 21 |
| Coffee | 1 | 1 | 1 | 22 |
| Cotton | 0.86 | 0.9 | 0.88 | 20 |
| Grapes | 1 | 1 | 1 | 18 |
| Jute | 0.84 | 0.93 | 0.88 | 28 |
| Kidneybeans | 1 | 1 | 1 | 14 |
| Lentil | 0.88 | 1 | 0.94 | 23 |
| Maize | 0.9 | 0.86 | 0.88 | 21 |
| Mango | 0.96 | 1 | 0.98 | 26 |
| Moth beans | 0.84 | 0.84 | 0.84 | 19 |
| Mungbean | 1 | 0.96 | 0.98 | 24 |
| Muskmelon | 1 | 1 | 1 | 23 |
| Orange | 1 | 1 | 1 | 29 |
| Papaya | 1 | 0.95 | 0.97 | 19 |
| Pigeon peas | 1 | 1 | 1 | 18 |
| Pomegranate | 1 | 1 | 1 | 17 |
| Rice | 0.85 | 0.69 | 0.76 | 16 |
| Watermelon | 1 | 1 | 1 | 15 |
|  |  |  |  |  |
| **Accuracy** | 0.95 | 440 |  |  |
| **Macro-Average** | 0.95 | 0.95 | 0.95 | 440 |
| **Weighted average** | 0.95 | 0.95 | 0.95 | 440 |

### **Random Forest**

from sklearn.ensemble import RandomForestClassifier

Random\_For = RandomForestClassifier(n\_estimators=20, random\_state=0)

Random\_For.fit(Xtrain,Ytrain)

Random\_For\_predicted\_values = Random\_For.predict(Xtest)

Random\_For\_x = metrics.accuracy\_score(Ytest, Random\_For\_predicted\_values)

Accuracy.append(Random\_For\_x)

model.append(' Random\_For')

print("RF's Accuracy is: ", x)

print(classification\_report(Ytest, Random\_For\_predicted\_values))

**OUTPUT:**

Random\_Forest's **Accuracy is: 0.9931818181818182**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-score** | **Support** |
|  |  |  |  |  |
| Apple | 1 | 1 | 1 | 13 |
| Banana | 1 | 1 | 1 | 17 |
| Blackgram | 0.94 | 1 | 0.97 | 16 |
| Chickpea | 1 | 1 | 1 | 21 |
| Coconut | 1 | 1 | 1 | 21 |
| Coffee | 1 | 1 | 1 | 22 |
| Cotton | 1 | 1 | 1 | 20 |
| Grapes | 1 | 1 | 1 | 18 |
| Jute | 0.9 | 1 | 0.95 | 28 |
| Kidneybeans | 1 | 1 | 1 | 14 |
| Lentil | 1 | 1 | 1 | 23 |
| Maize | 1 | 1 | 1 | 21 |
| Mango | 1 | 1 | 1 | 26 |
| Moth beans | 1 | 0.95 | 0.97 | 19 |
| Mungbean | 1 | 1 | 1 | 24 |
| Muskmelon | 1 | 1 | 1 | 23 |
| Orange | 1 | 1 | 1 | 29 |
| Papaya | 1 | 1 | 1 | 19 |
| Pigeon peas | 1 | 1 | 1 | 18 |
| Pomegranate | 1 | 1 | 1 | 17 |
| Rice | 1 | 0.81 | 0.9 | 16 |
| Watermelon | 1 | 1 | 1 | 15 |
|  |  |  |  |  |
| **Accuracy** | 0.99 | 440 |  |  |
| **Macro Average** | 0.99 | 0.99 | 0.99 | 440 |
| **Weighted Average** | 0.99 | 0.99 | 0.99 | 440 |

### **XGBoost**

import xgboost as xgb1

X\_G\_B = xgb1.XGBClassifier()

X\_G\_B.fit(Xtrain,Ytrain)

X\_G\_B\_predicted\_values = X\_G\_B.predict(Xtest)

X\_G\_B\_x = metrics.accuracy\_score(Ytest, X\_G\_B\_predicted\_values)

Accuracy.append(x)

model.append('XGBoost')

print("XGBoost's Accuracy is: ", X\_G\_B\_x)

print(classification\_report(Ytest, X\_G\_B\_predicted\_values))

**OUTPUT:**

XGBoost's **Accuracy is: 0.990909090909091**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-score** | **Support** |
|  |  |  |  |  |
| Apple | 1 | 1 | 1 | 13 |
| Banana | 1 | 1 | 1 | 17 |
| Blackgram | 1 | 1 | 1 | 16 |
| Chickpea | 1 | 1 | 1 | 21 |
| Coconut | 1 | 1 | 1 | 21 |
| Coffee | 0.96 | 1 | 0.98 | 22 |
| Cotton | 1 | 1 | 1 | 20 |
| Grapes | 1 | 1 | 1 | 18 |
| Jute | 1 | 0.93 | 0.96 | 28 |
| Kidneybeans | 1 | 1 | 1 | 14 |
| Lentil | 0.96 | 1 | 0.98 | 23 |
| Maize | 1 | 1 | 1 | 21 |
| Mango | 1 | 1 | 1 | 26 |
| Moth beans | 1 | 0.95 | 0.97 | 19 |
| Mungbean | 1 | 1 | 1 | 24 |
| Muskmelon | 1 | 1 | 1 | 23 |
| Orange | 1 | 1 | 1 | 29 |
| Papaya | 1 | 1 | 1 | 19 |
| Pigeon peas | 1 | 1 | 1 | 18 |
| Pomegranate | 1 | 1 | 1 | 17 |
| Rice | 0.94 | 1 | 0.97 | 16 |
| Watermelon | 1 | 1 | 1 | 15 |
|  |  |  |  |  |
| **Accuracy** | 0.99 | 440 |  |  |
| **Macro Average** | 0.99 | 0.99 | 0.99 | 440 |
| **Weighted Average** | 0.99 | 0.99 | 0.99 | 440 |

#### **Cross-validation score (XGBoost)**

XGB\_scoring **=** cross\_val\_score(X\_G\_B,features,target,cv**=**5)

XGB\_scoring

**OUTPUT:**

array([0.990909090909091, 0.9919190909091, 0.99119119090909, 0.991911909090])

## **Accuracy Comparison**

plt**.**figure(figsize**=**[10,5],dpi **=** 100)

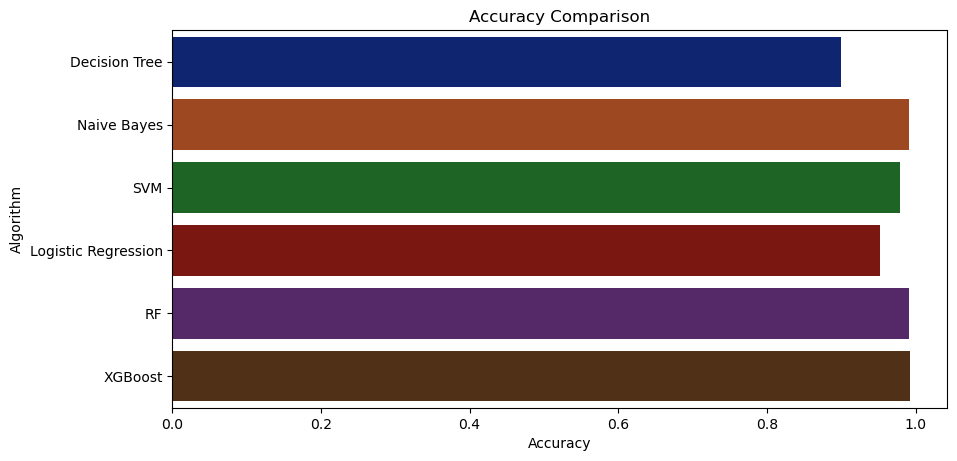
plt**.**title('Accuracy Comparison')

plt**.**xlabel('Accuracy')

plt**.**ylabel('Algorithm')

sns**.**barplot(x **=** Accuracy, y **=** model,palette**=**'dark')

**OUTPUT:**



## **Accuracy differentiates between all the methods:**

accuracy\_models\_algo **=** dict(zip(model, Accuracy))

for k, v in accuracy\_models\_algo**.**items():

print (k, '-->', v)

**OUTPUT:**

Decision\_Tree Accuracy--> 0.9

Naïve\_Bayes Accuracy --> 0.990909090909091

SVM Normalization Accuracy--> 0.9795454545454545

Logistic\_Regression Accuracy--> 0.9522727272727273

**RF Accuracy--> 0.9931818181818182**

XGBoost Accuracy--> 0.990909090909091

# **Result**

I am checking various Machine Learning models in a small dataset and the Random Forest best fit this model. I checked through **R- Square** Method it shows the Score **“0.9931818181818182”** accuracy**.**

## **Making a prediction**

Data\_predict **=** np**.**array([[105,19, 34, 21.6036, 63.3, 6.7, 140.91]])

prediction\_result **=** RF**.**predict(Data\_predict)

print(prediction\_result)

**OUTPUT:**

['coffee']

Data\_predict**=** np**.**array([[83, 45, 60, 28, 70.3, 7.0, 150.9]])

Prediction\_result **=** RF**.**predict(Data\_predict)

print(prediction\_result)

**OUTPUT:**

['jute']

# **Conclusion**

To determine the crop that would grow the most effective on a given plot of land, a comparison of the three different types of supervised machine-learning models (SVM, Decision Tree, and Random-Forest) is conducted. The crop prediction dataset had the best accuracy with Random-Forest-Classifier (RF) both in Entropy and Inequality standard with 99.32%, we concluded. In comparison, SVM has the lowest accuracy of the three (97.95%), and Decision Tree Classifier's accuracy is comparable to that of the Random Forest Classifier and SVM. Decision Tree Gini criteria provided a greater accuracy of 90% when compared to Decision Tree when the accuracy value was compared.

The criterion for Tree Entropy of Future data from the fields might be gathered to obtain a complete picture of the soil and incorporate further machine-learning and deep-learning algorithms, such as Artificial\_Neural\_Network or Convolution\_Neural\_Network, to categorize more types of crops.

**Random Forest Model** is a best suited model for this project gives a good prediction and the Accuracy is really good. The accuracy is checked through the R-Square metwhichhere gives the value between 0 to 1 where 1 is the best-fitted model and 0 is the lowest.

# **REFERENCES:**

1. Ashwani Kumar Kushwaha, Swetabhattachrya “crop yield prediction using the agro algorithm in Hadoop”.
2. Bharath S, Yeshwanth S, Yashas B L and Vidyaranya R Javalagi 2020 Comparative Analysis of Machine Learning Algorithms in The Study of Crop and Crop yield Prediction International Journal of Engineering Research & Technology (IJERT) NCETESFT – 2020 vol 8 Issue 14.
3. Dakar S and Rode S V 2014 Agricultural crop yield prediction using artificial neural network approach International Journal of Innovative Research in Electrical, Electronics, Instrumentation, and Control Engineering vol 2 Issue 1 pp 683-6.
4. Geetha V, Punitha A, Abarna M, Akshaya M, Illakiya S and Janani A P 2020 An effective crop prediction using random forest algorithm 2020 International Conference on System, Computation, Automation, and Networking (ICSCAN) pp 1-5 DOI: 10.1109/ICSCAN49426.2020.9262311.
5. Gulati P and Jha S K 2020 Efficient crop yield prediction in India using machine learning techniques International Journal of Engineering Research & Technology (IJERT) ENCADEMS – 2020 vol 8 Issue 10.
6. Gupta A, Nagda D, Nikhare P, Sandbhor A, 2021, Smart crop prediction using IoT and machine learning International Journal of Engineering Research & Technology (IJERT) NTASU – 2020 vol 9 Issue 3.
7. Himani Sharma, Sunil Kumar “A Survey on Decision Tree Algorithms of Classification in Data Mining”.
8. Kalimuthu M, Vaishnavi P and Kishore M 2020 Crop prediction using machine learning 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT) pp 926-32 doi: 10.1109/ICSSIT48917.2020.9214190.
9. Mahendra N, Vishwakarma D, Nischitha K, Ashwini and Manjuraju M. R 2020 Crop prediction using machine learning approaches, International Journal of Engineering Research & Technology (IJERT) vol 9 Issue 8 (August 2020).
10. Medar R, Rajpurohit V S and Shweta S 2019 Crop yield prediction using machine learning techniques IEEE 5th International Conference for Convergence in Technology (I2CT) pp 1-5 doi: 10.1109/I2CT45611.2019.9033611.
11. Nishant P S, Venkat P S, Avinash B L and Jabber B 2020 Crop yield prediction based on Indian agriculture using machine learning 2020 International Conference for Emerging Technology (INCET) pp 1-4 doi: 10.1109/INCET49848.2020.9154036.
12. Pande S M, Ramesh P K, Anmol A, Aishwaraya B R, Rohilla K and Shaurya K 2021 Crop recommender system using machine learning approach 2021 5th International Conference on Computing Methodologies and Communication (ICCMC) pp 1066-71 doi: 10.1109/ICCMC51019.2021.9418351.
13. Pavan Patil, Virendra Panpatil, Prof. Shrikant Kokate “Crop Prediction System using Machine Learning Algorithms”.
14. Pijush Samui, Venkata Ravibabu Mandla, Arun Krishna and Tarun Teja “Prediction of Rainfall Using Support Vector Machine and Relevance Vector Machine”.
15. Prof. D.S. Zingade ,Omkar Buchade ,Nilesh Mehta ,Shubham Ghodekar ,Chandan Mehta “Crop Prediction System using Machine Learning”.
16. Rahul Katarya, Ashutosh Raturi, Abhinav Mehndiratta, Abhinav Thapper “Impact of Machine Learning Techniques in Precision Agriculture”.
17. Sellam V, and Poovammal E 2016 Prediction of crop yield using regression analysis Indian Journal of Science and Technology vol 9(38) pp 1-5.
18. Suresh A, Ganesh P and Ramalatha M 2018 Prediction of major crop yields of Tamilnadu using K-means and Modified KNN 2018 3rd International Conference on Communication and Electronics Systems (ICCES) pp 88-93 doi: 10.1109/CESYS.2018.8723956.