Intelligent Crop Recommendation Model

Contents

Introduction

Introduction

Agriculture is a critical sector of the economy and plays a key role in the livelihood of many individuals worldwide.

However, climate change, unpredictable weather patterns, and the increasing global population are presenting new challenges to agriculture.

To meet these challenges, farmers need to adopt innovative technologies and practices to improve productivity and profitability.

Abstract

The proposed Intelligent Crop Prediction System uses the historical data and machine learning algorithms to analyze various factors such as weather conditions and soil quality to predict crop yields for a given season.

The system recommends farmers to make informed decisions regarding crop selection and planting schedules leading to increased productivity and reduced risk of crop failure.

Objectives

Objectives

- To collect and explore suitable dataset for crop prediction.
- To preprocess the collected data suitable to Machine Learning.
- To build and validate a Machine Learning Model using Support Vector Machine(SVM), Random Forest(RF) and GradientBoost Techniques.
- To recommend the best suitable crops in the area so that the farmers get the maximum productivity.

LITERATURE REVIEW

1 Nischitha, K., Vishwakarma, D., Ashwini, M. N., & Manjuraju, M. R. (2020). Crop prediction using machine learning approaches. International Journal of Engineering Research & Technology (IJERT), 9(08), 23-26.

Observations :

- Chosen the SVM model for building crop prediction.
- Chosen the KNN model to perform classification.

2 Rao, M. S., Singh, A., Reddy, N. S., & Acharya, D. U. (2022). Crop prediction using machine learning. In Journal of Physics: Conference Series (Vol. 2161, No. 1, p. 012033). IOP Publishing.

Observations:

- Chosen Random Forest model to handle a large number of input variables like soil properties and weather conditions.
- To provide timely and accurate information on crop fields.

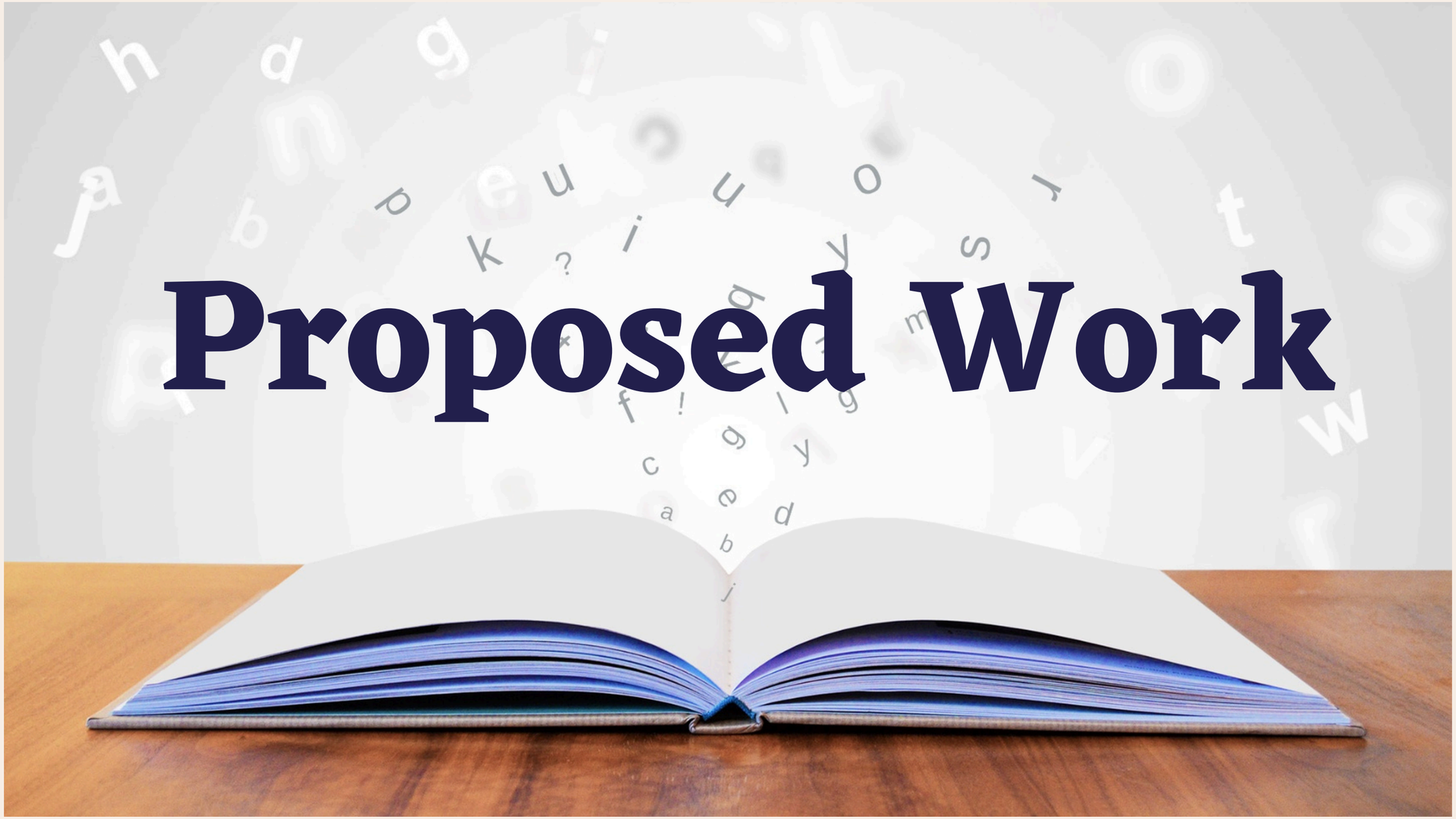
Modules

1. Crop Data collection.
2. Data Exploration.
3. Data Preprocessing.
4. Machine learning model building and testing.
5. Recommending Crop.
6. Experimental Analysis.

TIMELINE FOR DEVELOPMENT OF PROJECT

MODULE	MARCH	APRIL	MAY
M1			
M2			
M3			
M4			
M5			
M6			

Proposed Work



ARCHITECTURE

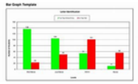
Crop Data Collection

Data collection

Dataset

Data Exploration

Pie chart

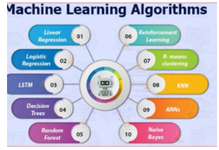


Bar graph

- Data preprocessing

ML Model Building

- SVM
- KNN
- Logistic Regression
- Gradient Boost
- Random Forest



ML Model Validation

70% Training

30% Testing

Experimental Analysis

- Accuracy
- Precision
- F1-Score
- Recall

Crop Recommendations

Coconut
Wheat
Papaya
Orange
Maize
Apple
Chickpeat
Watermelon
Kidney beans
Grapes
pigeonpeas
Mangot
Mothbeans
Banana
Mung beans
pomogranate
Black gram
Lentil
Cotton
Coffee
jute

1 Crop Data collection

Source :https://www.kaggle.com/datasets/chitrakumari25/smart-agricultural-production-optimizing-engine

Number of features : 7

1. Nitrogen
2. Phosphorus
3. Potassium
4. Temperature
5. Humidity
6. pH
7. Rainfall

- Number of Tuples : 2200
- Class : Accurate Crop Prediction
  - There are 23 crops in our dataset
- Type : .csv file

Hardware Requirements :

- Processor(i3)
- RAM(8GB)
- Hard Disk

Software Requirements :

- HTML
- CSS
- Flask
- Visual Studio code
- python

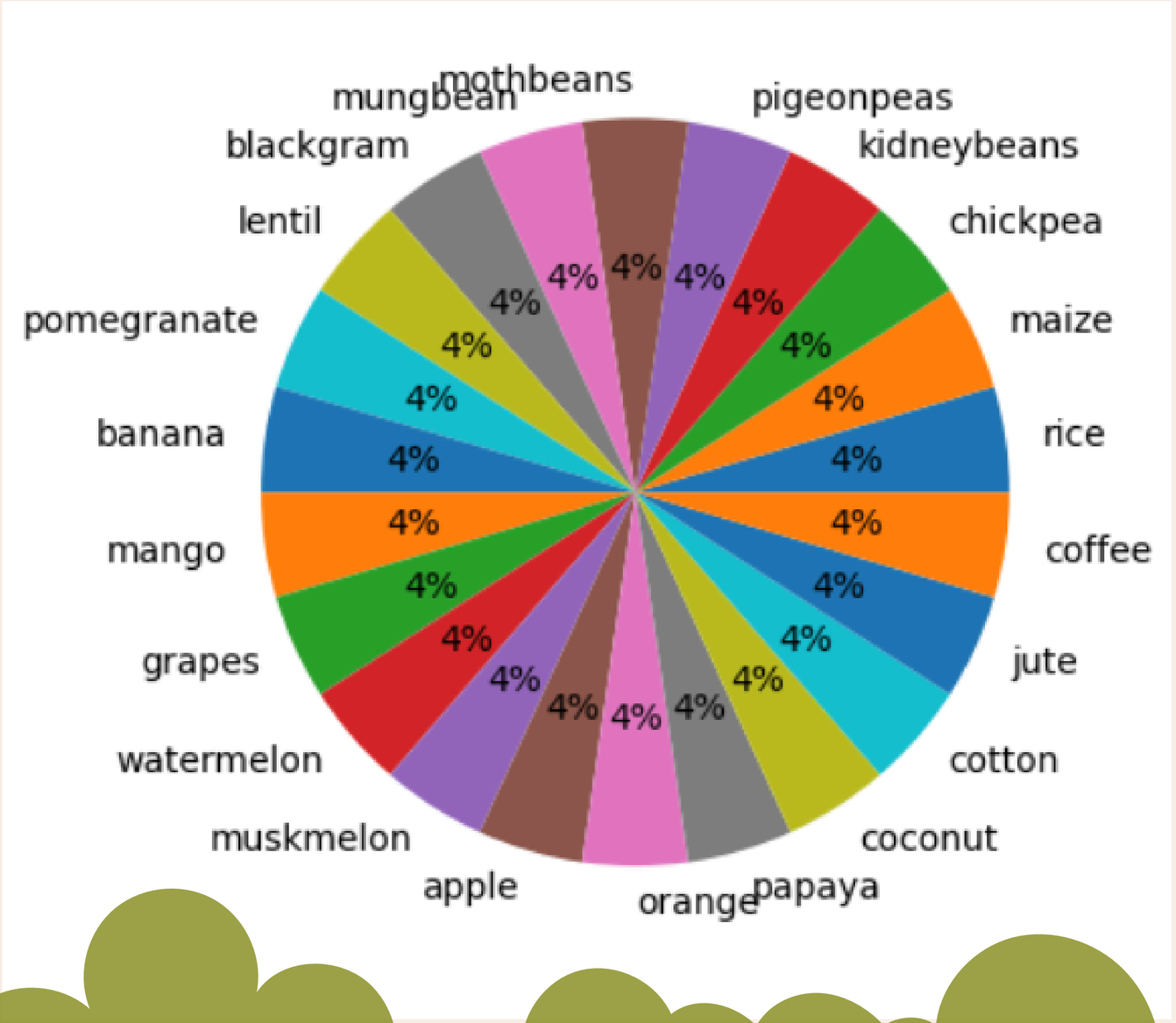
2 Data Visualisation

```
crops = df['label'].unique()
print(len(crops))
print(crops)
print(pd.value_counts(df['label']))
```

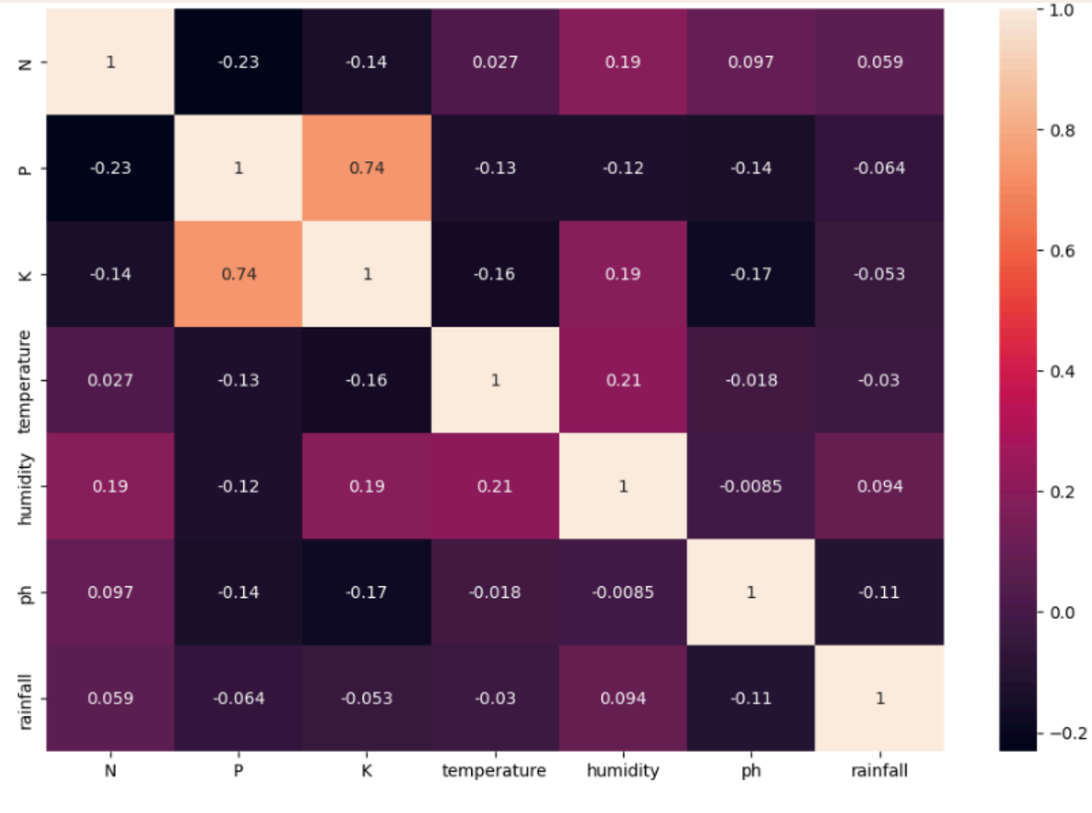
```
22
['rice' 'maize' 'chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans'
 'mungbean' 'blackgram' 'lentil' 'pomegranate' 'banana' 'mango' 'grapes'
 'watermelon' 'muskmelon' 'apple' 'orange' 'papaya' 'coconut' 'cotton'
 'jute' 'coffee']
rice           100
maize          100
jute           100
cotton         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: label, dtype: int64
```

Using pie chart :

1. This Dataset is properly categorised and is therefore more appropriate for machine learning techniques training.



Using Heat map :



1. A Heat map is a graphical representation of multivariate data that is structured as a matrix of columns and rows.
2. A heat map represents these coefficients to visualize the strengthof correlation among variables. It helps to find features that are best for Machine Learning model building

3 Data Preprocessing

- Null Values
- Normalization
- Standardization
- Discretization

```
[2]  import pandas as pd
     import numpy as np
     import seaborn as sns
     from matplotlib import pyplot as plt
```

```
df = pd.read_csv('/content/drive/MyDrive/dataset/Crop_recommendation.csv')
df.head(10)
```

	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
5	69	37	42	23.058049	83.370118	7.073454	251.055000	rice
6	69	55	38	22.708838	82.639414	5.700806	271.324860	rice
7	94	53	40	20.277744	82.894086	5.718627	241.974195	rice
8	89	54	38	24.515881	83.535216	6.685346	230.446236	rice
9	68	58	38	23.223974	83.033227	6.336254	221.209196	rice

```
[10] df.columns
```

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

```
[7] df.label.unique()
```

```
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)
```

```
[8] df.dtypes
```

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

```
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
```

```
[11] df.shape
```

```
(2200, 8)
```

```
[9] df.size
```

```
17600
```

```
[17] df.describe()
```

	N	P	K	temperature	humidity	ph	rainfall
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480	103.463655
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54.958389
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.211267
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64.551686
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94.867624
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124.267508
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.560117

NULL VALUES

```
print(df.isnull().sum())
```

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

4 Machine Learning model building and testing

- K-Nearest Neighbor
- Support Vector Machine
- Decision Tree
- Random Forest
- Logistic Regression
- GradientBoost

KNN

Knn is simple to understand and interpret, making it useful for nonlinear data and situations where the data distribution is unknown.

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(xtrain, ytrain)
```

SVM

The Support Vector Machine (SVM) model is a powerful machine learning algorithm. It finds a clear decision boundary between different classes by maximizing the margin between the support vectors

```
from sklearn.svm import SVC
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import confusion_matrix
norm = MinMaxScaler().fit(xtrain)
x_train_norm = norm.transform(xtrain)
# transform testing dataabs
x_test_norm = norm.transform(xtest)
SVM = SVC(kernel='poly', degree=3, C=1)
SVM.fit(x_train_norm,ytrain)
```

Decision Tree

Decision Tree uses a tree-like structure to make decisions by asking a series of questions about the data. Each question leads to a branch that eventually reaches a prediction.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.metrics import classification_report

dt = DecisionTreeClassifier(criterion="entropy",random_state=42,max_depth=5)
dt.fit(xtrain,ytrain)
```

Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees. Each decision tree is trained on a random subset of the training data.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix

RF = RandomForestClassifier(n_estimators=20, random_state=0)
RF.fit(xtrain,ytrain)
```

Logistic Regression

Logistic Regression predicts the probability of an instance belonging to a certain class based on input features. By applying a logistic function, it maps the predictions to a value between 0 and 1.

```
from sklearn.linear_model import LogisticRegression

LogReg = LogisticRegression(random_state=2)

LogReg.fit(xtrain,ytrain)
```

Gradient Boost

Gradient Boost ensemble of weak predictive models in a sequential manner.Each subsequent model in the ensemble is trained to correct the mistakes made by the previous models.

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(xtrain, ytrain)
```

SPLITTING INTO TRAIN AND TEST DATA

```
[23] from sklearn.model_selection import train_test_split
```

0s

```
[24] xtrain,xtest,ytrain,ytest = train_test_split(features,target,random_state=42,test_size=0.3)
```

0s

Making a prediction

In [32]:

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

```
['coffee']
```

5 Recommending Crop

```python
import pandas as pd
from sklearn.ensemble import RandomForestClassifier

# Load the dataset
data = pd.read_csv('/content/drive/MyDrive/dataset/Crop_recommendation.csv')

# Split the dataset into input features (X) and target variable (y)
X = data[['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall']]
y = data['label']

# Create a Random Forest Classifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the classifier
clf.fit(X, y)

def recommend_crop(input_data):
    # Create a DataFrame with the input parameters
    input_df = pd.DataFrame([input_data],
                            columns=['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall'])

    # Make predictions on the input data
    crop = clf.predict(input_df)

    return crop[0]

# Example usage
input_values = [44,55,25,29.63,65.91,7.42,71.16]
recommended_crop = recommend_crop(input_values)
print("Recommended Crop:", recommended_crop)
```

```
Recommended Crop: blackgram
```

6 Experimental Analysis

KNN

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	34
banana	1.00	1.00	1.00	26
blackgram	0.96	0.96	0.96	26
chickpea	1.00	1.00	1.00	34
coconut	1.00	1.00	1.00	33
coffee	1.00	0.97	0.98	30
cotton	0.96	0.96	0.96	28
grapes	1.00	1.00	1.00	23
jute	0.78	0.91	0.84	34
kidneybeans	0.97	1.00	0.99	36
lentil	0.88	1.00	0.94	22
maize	0.96	0.96	0.96	26
mango	1.00	1.00	1.00	32
mothbeans	0.97	0.91	0.94	34
mungbean	1.00	1.00	1.00	30
muskmelon	1.00	1.00	1.00	24
orange	1.00	1.00	1.00	25
papaya	1.00	0.97	0.99	37
pigeonpeas	1.00	0.95	0.97	37
pomegranate	1.00	1.00	1.00	38
rice	0.83	0.71	0.77	28
watermelon	1.00	1.00	1.00	23
accuracy			0.97	660
macro avg	0.97	0.97	0.97	660
weighted avg	0.97	0.97	0.97	660

Knn's Accuracy is : 0.9681818181818181

SVM

SVM's Accuracy is: 0.9681818181818181

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	34
banana	1.00	1.00	1.00	26
blackgram	0.93	1.00	0.96	26
chickpea	1.00	1.00	1.00	34
coconut	1.00	1.00	1.00	33
coffee	1.00	0.93	0.97	30
cotton	0.97	1.00	0.98	28
grapes	1.00	1.00	1.00	23
jute	0.73	0.97	0.84	34
kidneybeans	0.97	1.00	0.99	36
lentil	0.91	0.95	0.93	22
maize	1.00	0.96	0.98	26
mango	0.97	1.00	0.98	32
mothbeans	0.97	0.94	0.96	34
mungbean	1.00	1.00	1.00	30
muskmelon	1.00	1.00	1.00	24
orange	1.00	1.00	1.00	25
papaya	0.97	1.00	0.99	37
pigeonpeas	1.00	0.89	0.94	37
pomegranate	1.00	1.00	1.00	38
rice	1.00	0.64	0.78	28
watermelon	1.00	1.00	1.00	23
accuracy			0.97	660
macro avg	0.97	0.97	0.97	660
weighted avg	0.97	0.97	0.97	660

Decision Tree

DecisionTrees's Accuracy is: 0.8681818181818182

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	34
banana	1.00	1.00	1.00	26
blackgram	0.57	1.00	0.72	26
chickpea	1.00	0.97	0.99	34
coconut	1.00	1.00	1.00	33
coffee	1.00	1.00	1.00	30
cotton	1.00	1.00	1.00	28
grapes	1.00	1.00	1.00	23
jute	0.71	0.94	0.81	34
kidneybeans	0.00	0.00	0.00	36
lentil	0.50	1.00	0.67	22
maize	1.00	1.00	1.00	26
mango	1.00	1.00	1.00	32
mothbeans	0.00	0.00	0.00	34
mungbean	0.97	1.00	0.98	30
muskmelon	1.00	1.00	1.00	24
orange	1.00	1.00	1.00	25
papaya	0.97	0.92	0.94	37
pigeonpeas	0.57	1.00	0.73	37
pomegranate	1.00	1.00	1.00	38
rice	0.89	0.61	0.72	28
watermelon	1.00	1.00	1.00	23
accuracy			0.87	660
macro avg	0.83	0.88	0.84	660
weighted avg	0.81	0.87	0.83	660

Random Forest

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	34
banana	1.00	1.00	1.00	26
blackgram	1.00	1.00	1.00	26
chickpea	1.00	1.00	1.00	34
coconut	1.00	1.00	1.00	33
coffee	1.00	1.00	1.00	30
cotton	1.00	1.00	1.00	28
grapes	1.00	1.00	1.00	23
jute	0.87	1.00	0.93	34
kidneybeans	1.00	1.00	1.00	36
lentil	1.00	1.00	1.00	22
maize	1.00	1.00	1.00	26
mango	1.00	1.00	1.00	32
mothbeans	1.00	1.00	1.00	34
mungbean	1.00	1.00	1.00	30
muskmelon	1.00	1.00	1.00	24
orange	1.00	1.00	1.00	25
papaya	1.00	1.00	1.00	37
pigeonpeas	1.00	1.00	1.00	37
pomegranate	1.00	1.00	1.00	38
rice	1.00	0.82	0.90	28
watermelon	1.00	1.00	1.00	23
accuracy			0.99	660
macro avg	0.99	0.99	0.99	660
weighted avg	0.99	0.99	0.99	660

RF's Accuracy is: 0.9924242424242424

Logistic Regression

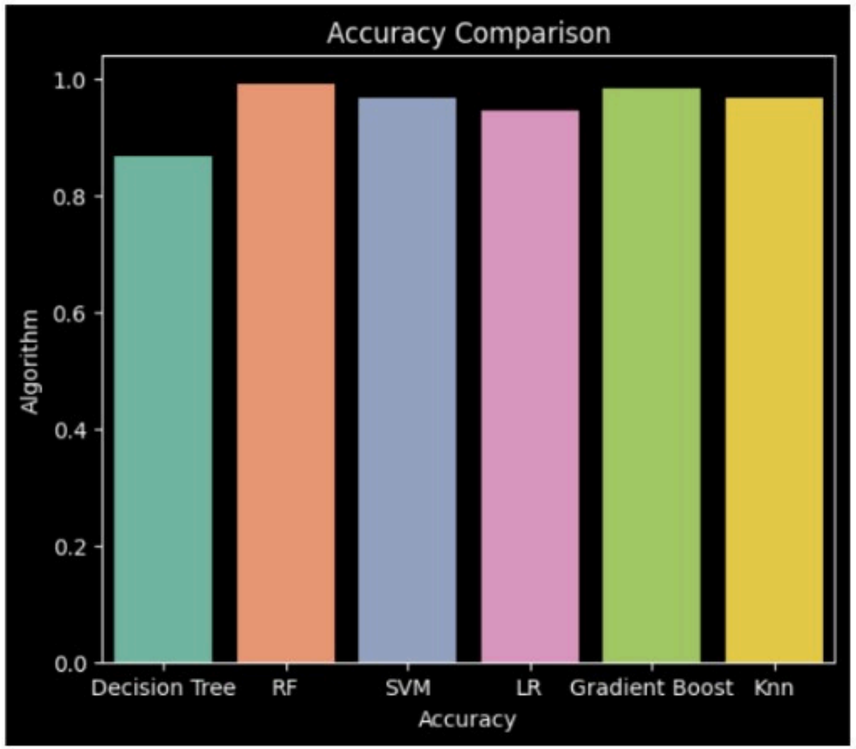
Logistic Regression's Accuracy is: 0.946969696969697

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	34
banana	0.96	1.00	0.98	26
blackgram	0.76	0.85	0.80	26
chickpea	1.00	1.00	1.00	34
coconut	1.00	1.00	1.00	33
coffee	0.97	1.00	0.98	30
cotton	0.87	0.96	0.92	28
grapes	1.00	1.00	1.00	23
jute	0.81	0.88	0.85	34
kidneybeans	1.00	0.97	0.99	36
lentil	0.91	0.91	0.91	22
maize	0.95	0.77	0.85	26
mango	0.97	1.00	0.98	32
mothbeans	0.90	0.82	0.86	34
mungbean	0.94	1.00	0.97	30
muskmelon	1.00	1.00	1.00	24
orange	1.00	1.00	1.00	25
papaya	0.94	0.92	0.93	37
pigeonpeas	0.95	0.95	0.95	37
pomegranate	1.00	1.00	1.00	38
rice	0.92	0.79	0.85	28
watermelon	1.00	1.00	1.00	23
accuracy			0.95	660
macro avg	0.95	0.95	0.95	660
weighted avg	0.95	0.95	0.95	660

COMPARISION OF ALGORITHMS

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=[6, 5])
plt.title('Accuracy Comparison')
plt.xlabel('Accuracy')
plt.ylabel('Algorithm')
sns.barplot(x = model, y = acc)

plt.show()
```



CROP RECOMMENDATION

Nitrogen in parts per million(ppm):

Nitrogen

Phosphorous in parts per million(ppm):

Phosphorous

Potassium in parts per million(ppm):

Potassium

Temperature in celsius:

Temperature

Humidity in percentage(%):

Humidity

pH(0-14):

Ph

Rainfall in millimeters(mm):

Rainfall

Predict

CROP RECOMMENDATION

Nitrogen in parts per million(ppm):

61

Phosphorous in parts per million(ppm):

41

Potassium in parts per million(ppm):

17

Temperature in celsius:

25.14

Humidity in percentage(%):

65.26

pH(0-14):

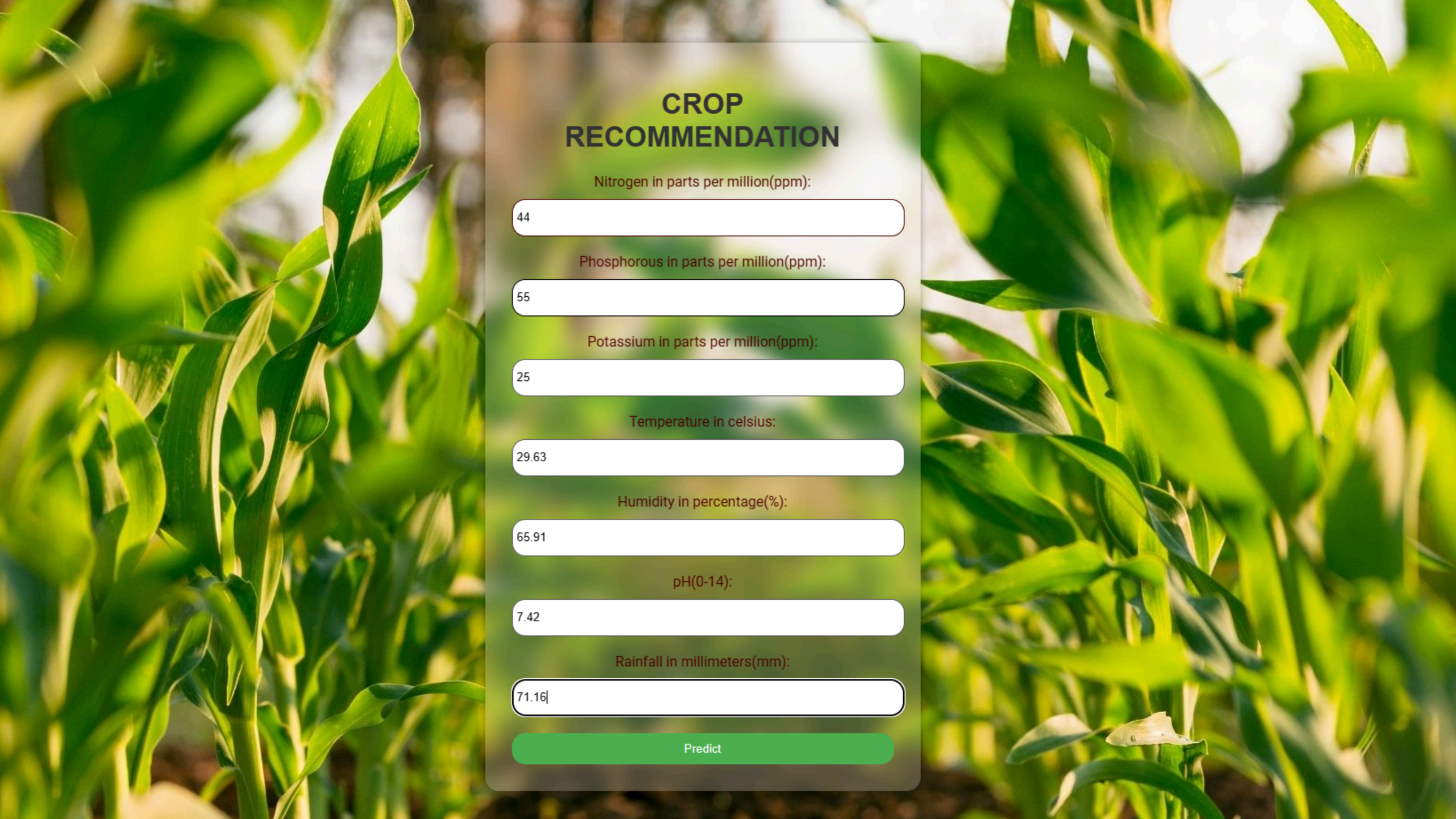
6.02

Rainfall in millimeters(mm):

76.68

Predict

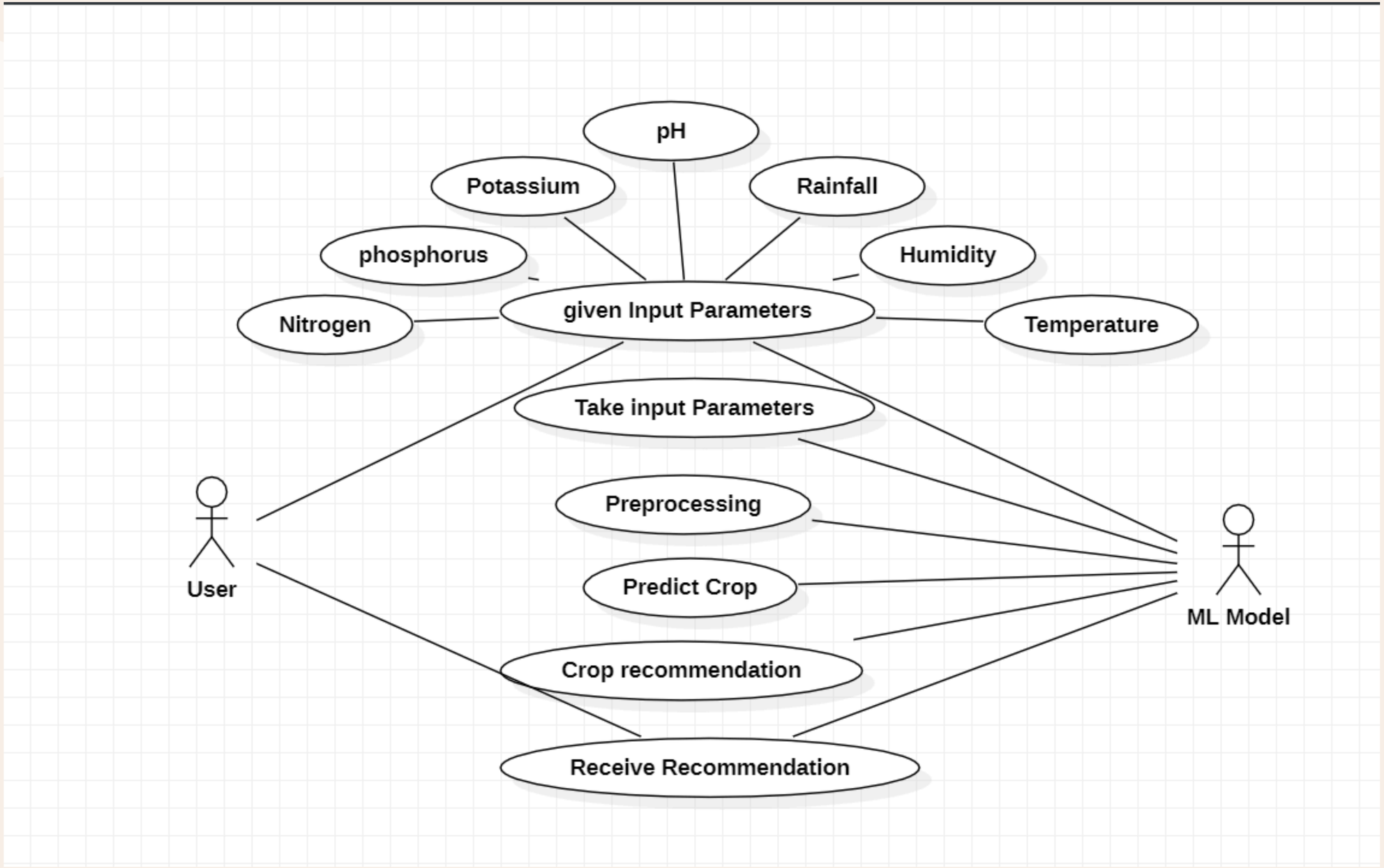

THE BEST
RECOMMENDED
CROP IS
coffee


CROP RECOMMENDATION
Nitrogen in parts per million(ppm):
44
Phosphorous in parts per million(ppm):
55
Potassium in parts per million(ppm):
25
Temperature in celsius:
29.63
Humidity in percentage(%):
65.91
pH(0-14):
7.42
Rainfall in millimeters(mm):
71.16
Predict

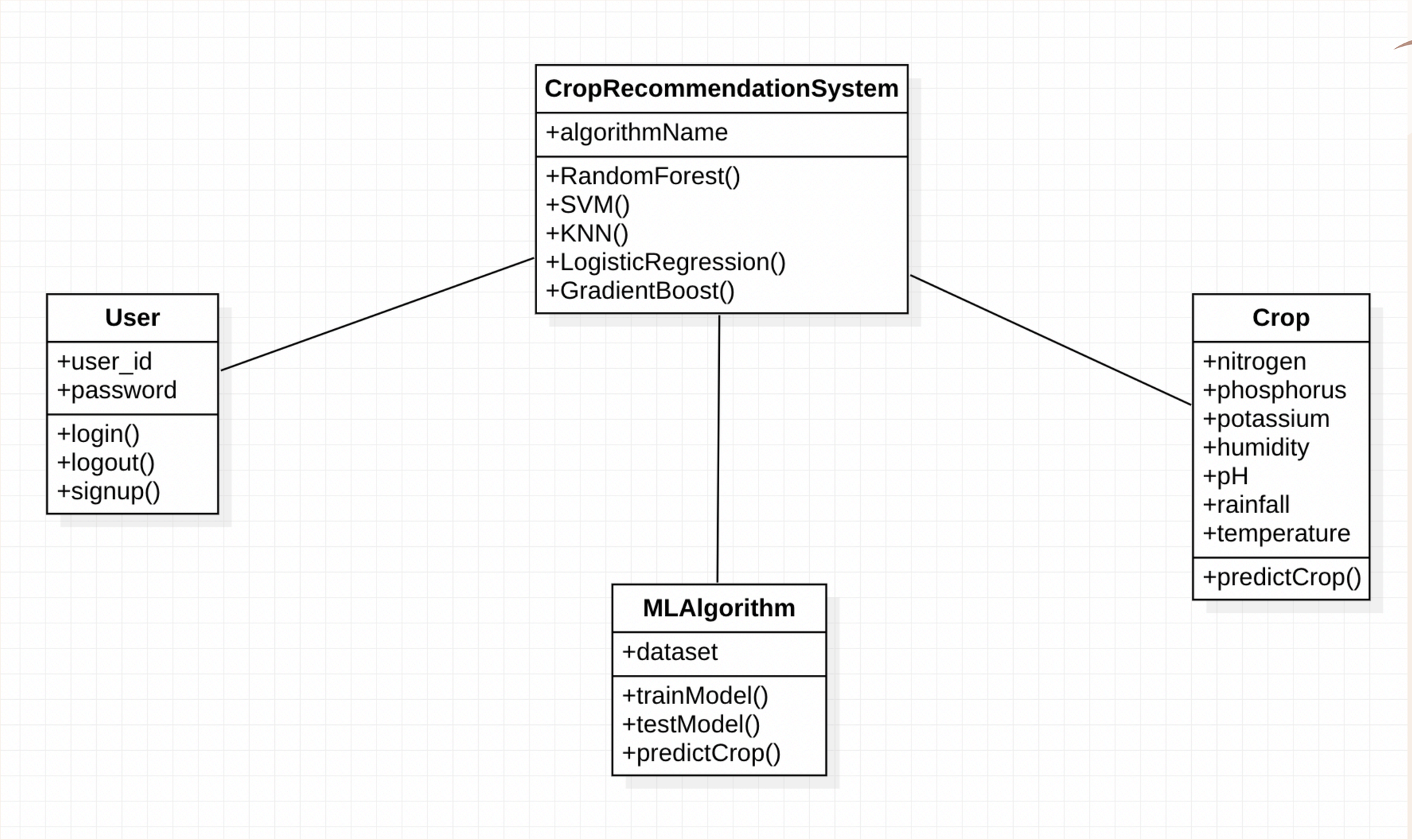

THE BEST
RECOMMENDED
CROP IS
blackgram

UML Diagrams

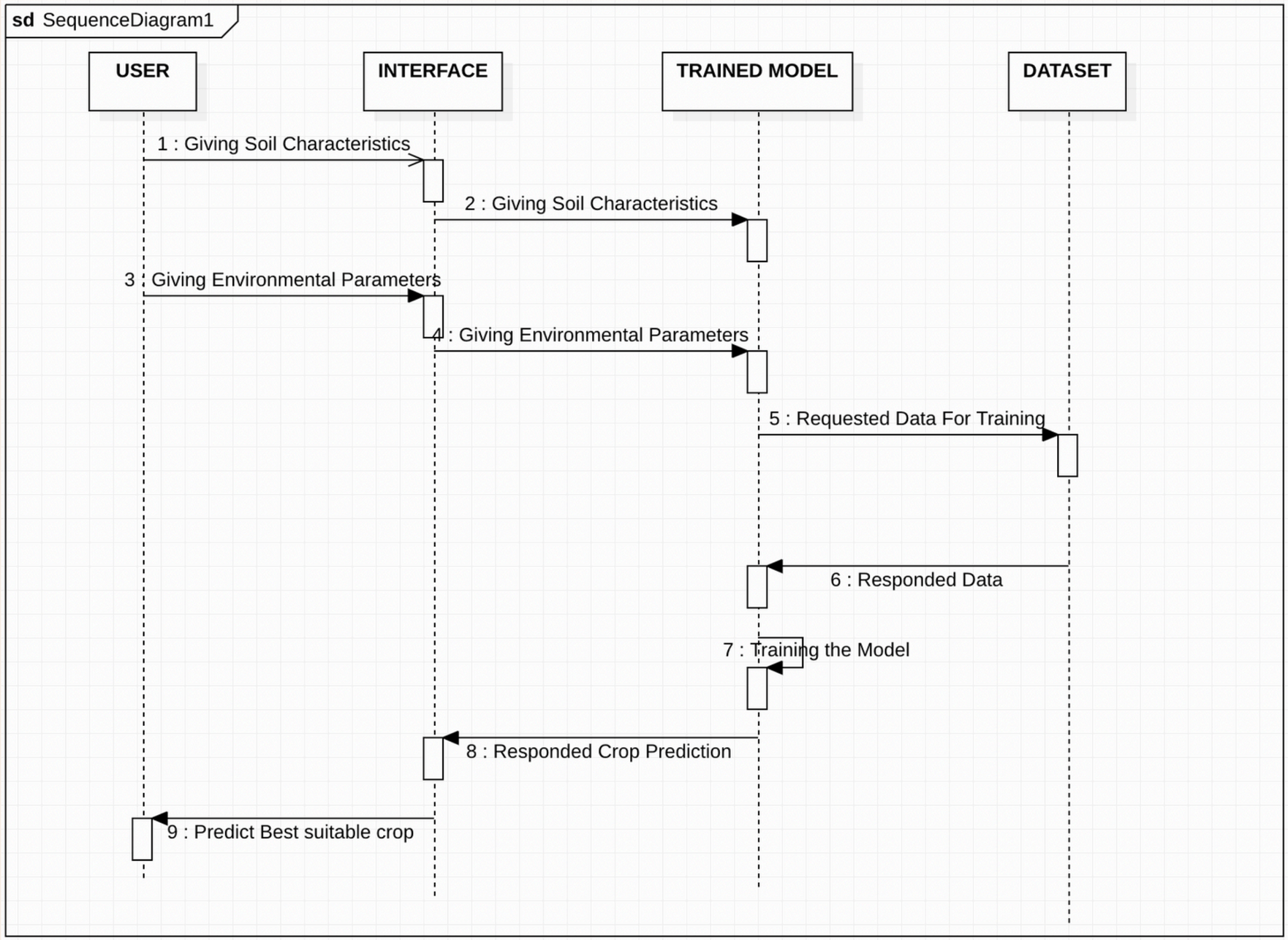
USECASE DIAGRAM



CLASS DIAGRAM



SEQUENCE DIAGRAM



Thank you