Mini Project Report on

Crop Recommendation System

Submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by:

Student Name: Rameshwari Kapoor **University Roll.No.:** 2119003

Under the Mentorship of

Ms. Preeti Chaudhary

Assistant Professor

Department of Computer Science and Engineering

Graphic Era Hill University

Dehradun, Uttarakhand

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the project report entitled **"Crop Recommendation System"** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering of the Graphic Era Hill University, Dehradun shall be carried out by myself under the mentorship of **Ms. Preeti Chaudhary, Assistant Professor**, Department of Computer Science and Engineering, Graphic Era Hill University, Dehradun.

Name: Rameshwari Kapoor **University Roll no.:** 2119003

Table of Contents

Chapter No.	Description	Page No.
Chapter 1	Introduction about Machine Learning	4
Chapter 2	Crop Recommendation System Briefing	6
Chapter 3	Methodology	9
Chapter 4	Code	11
Chapter 5	Need of Crop Recommendation System	19
Chapter 6	Application	20
Chapter 7	Conclusion	21

Chapter 1

Introduction

Machine Learning

The term 'Machine Learning' was coined by Arthur Samuel in 1959. Machine learning is a rapidly expanding field of artificial intelligence that allows computers to automatically infer information from historical data. Machine learning employs a variety of methods to create mathematical models and make predictions using knowledge or historical data. Currently, it is being used for various tasks such as house price prediction, crop recommendation system, image/speech recognition, email filtering, Facebook auto-tagging, recommender system, Natural Language Processing and many more.

- Classification

There are several types of machine learning, including supervised learning, unsupervised learning, and reinforcement learning.

- Supervised learning

Supervised learning, also known as supervised machine learning, is a subcategory of machine learning and artificial intelligence. It uses labelled datasets to train algorithms to classify data or predict outcomes accurately.

- Unsupervised learning

Unsupervised Machine Learning is a type of machine learning that utilizes unsupervised algorithms to classify and group un-labelled datasets. These algorithms can identify hidden patterns or clusters of data without human intervention.

- Reinforcement learning

RL is a form of machine learning in which an agent learns through trial and error based on feedback from its actions and experiences in an interactive setting.

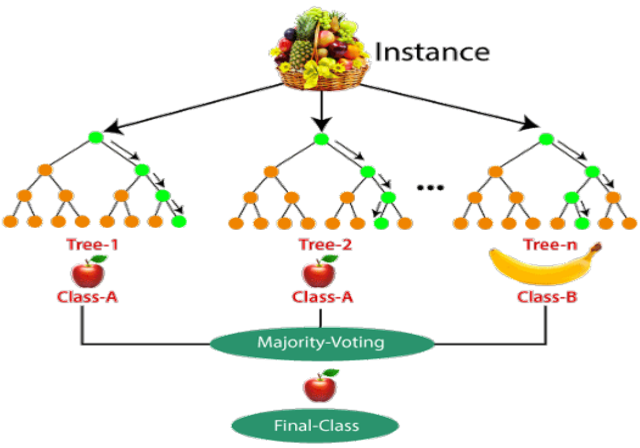
Chapter 2

Crop Recommendation System

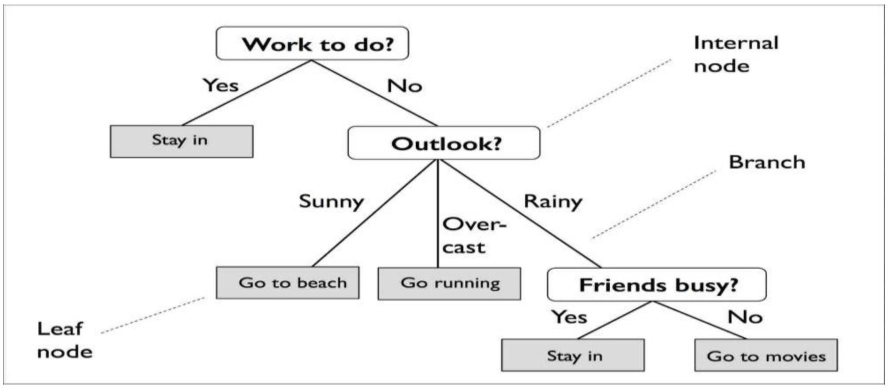
A Crop Recommendation System uses machine learning algorithms to provide recommendations to the farmers on the best crops to plant based on specific conditions. It considers various factors like amount of NPK (Nitrogen, Phosphorous, Potassium) present in the soil, temperature, and humidity in that area, ph of soil and the amount of rainfall received during that time. The system provides a connectivity to farmers via a mobile application. Farmers can use this system to improve their yield and thus improving their agricultural productivity and profitability.

In this project I have merged two crop recommendation datasets to increase the data and handle the problem of overfitting. The main objective/aim of this project is to use different ML Algorithms and check which algorithm(s) gives the best result. Some of the algorithms that I have used in this project for making model includes:

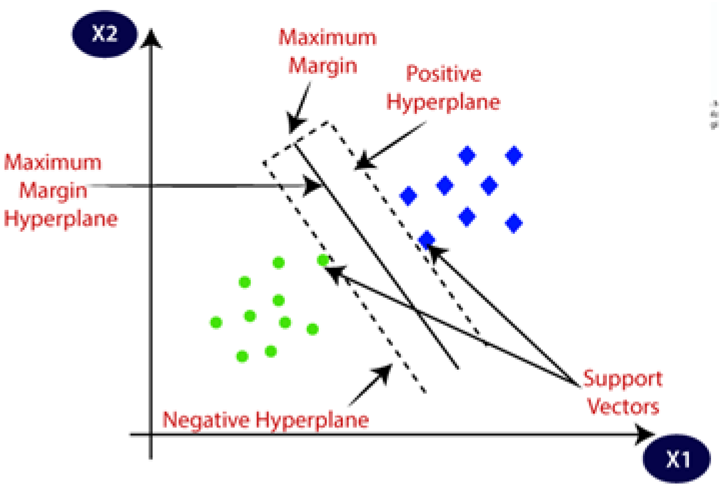
- **Random Forest: -** It is a popular supervised ML algorithm which can be used for both regression and classification. It produces predictions based on the majority votes from several decision trees rather than relying just on one.



- **<u>Decision Tree:-</u>** It is also a supervised ML algorithm that can be used for both regression and classification. In a decision tree the internal nodes represent the features, branches represent the decision rules and the leaf nodes represent the outcome. It simply asks a question and based on the answer(yes/no), it splits the tree into subtrees.

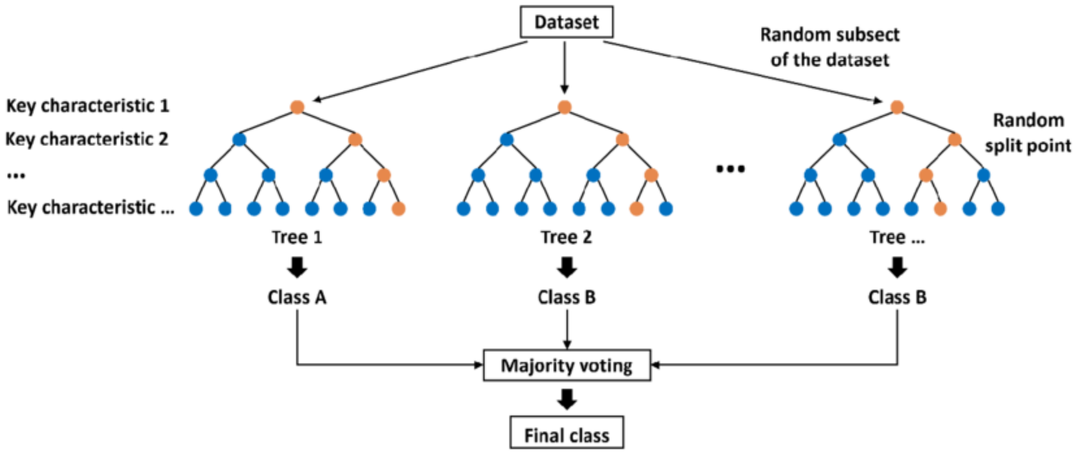


- **<u>Support Vector Classification: -</u>** SVC is a supervised learning algorithm used for classification tasks. It aims to find the best hyperplane that separates different classes of data points in an n-dimensional space. The best hyperplane is the one that has the largest margin, which is the distance between the hyperplane and the closest data points of each class, also known as support vectors.



- **LightGBM:-** Light Gradient Boosting Machine is a gradient boosting framework that is particularly used for tasks like classification and regression. It is based on decision trees. Its aim is to increase the efficiency of the model and reduce the memory usage.

- **ExtraTrees Classifier:-** The extra trees classifier is a powerful tool particularly used for classification tasks. It randomly considers a bunch of potential splits at each decision point.



Chapter 3

Methodology

- ## How the Model Works?

- **Importing Modules: -** The code starts by importing necessary Python modules and setting up the environment for data analysis and visualization.
- **Data Gathering and Displaying:** Two datasets (df1 and df2) are read using the Pandas library. Concatenation is then performed to combine these datasets into one. Duplicates are then removed from the combined dataset.
- **Exploratory Data Analysis (EDA):** The code performs EDA on the dataset. It includes checking the dataset’s shape. Columns, duplicated values, and basic statistics. Box plots are created to visualize the distribution of key features like nitrogen (N), phosphorous (P), potassium (K), temperature, humidity, pH, and rainfall.
- **Preprocessing:** Outliers in the 'rainfall' feature are identified and removed using the Interquartile Range (IQR) method. Correlation between different features is visualized using a heatmap.
- **Data Visualization:** Bar plots are created to visualize the distribution of nitrogen (N), phosphorous (P), and potassium (K) for each crop.
- **Model Training:** The dataset is split into training and testing sets. Several machine learning models are trained using the training set. The models used include LightGBM, Decision Tree, Random Forest, Support Vector Machine (SVM), and Extra Trees. Each model's accuracy is evaluated using the testing set.

- **Model Evaluation:** Confusion matrices and classification reports are generated to evaluate the performance of each machine learning model. The metrics include accuracy, precision, recall, and F1-score.
- **Model Testing**: The trained Random Forest model is tested with several sample inputs (test,test1, etc.) to make crop predictions based on input feature values.
- **Model Comparison:** The code compares the accuracy of different machine learning models (LightGBM, Decision Tree, Random Forest, SVM, and Extra Trees) using the testing set.

- **Model Saving:** The Support Vector Machine (SVM) model is saved using the pickle library (pickle.dump) for future use.

How the GUI works?

- **Importing Libraries:** The code starts by importing necessary libraries like streamlit, joblib, numpy, pickle, and warnings.
- **Loading the Machine Learning Model:** The code loads a machine learning model(‘modelSVC.pkl’) using the ‘pickle’ library.
- **User Input:** The code uses Streamlit sliders to allow the users to input values for Nitrogen, Phosphorous, Potassium, Temperature, Humidity, pH, and Rainfall.
- **Making Predictions:** The user inputs are combined into a list ‘p’, which is then used to make predictions using the loaded machine learning model.
- **Displaying Results:** The recommended crop is displayed using ‘st.write’.
- **Providing Recommendations and Information:** Based on the predicted crop, the code displays specific recommendations and information about the do’s and don’ts for cultivating that crop.

Chapter 4

Code

1. Collecting and displaying the data

Importing Modules

In [1]:
```
import IPython
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from IPython import get_ipython
import warnings
warnings.filterwarnings('ignore')
```

Data Gathering and Displaying

In [2]:
```
df1 = pd.read_csv('Crop_recommendation.csv')
df2 = pd.read_csv("C:\\Users\\Manan\\Downloads\\archive (5)\\Crop_recommendation.csv")
```

In [3]:
```
df1.head()
```

Out[3]:

	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

In [4]:
```
df1
```

Out[4]:

	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
...	...	...	...	...	...	...	...	...
1692	117	86	48	28.695620	82.541958	6.225225	116.161684	banana
1693	114	94	53	26.335449	76.853201	6.190757	118.685826	banana
1694	110	78	50	25.937302	78.898644	5.915569	98.217475	banana
1695	94	70	48	25.136865	84.883944	6.195152	91.464425	banana
1696	80	71	47	27.505277	80.797840	6.156373	105.077699	banana

1697 rows × 8 columns

2. EDA

Exploratory Data Analysis

```
In [6]: df = pd.concat([df1,df2])
```

```
In [7]: df
```

Out[7]:

	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
...	...	...	...	...	...	...	...	...
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

3897 rows × 8 columns

```
In [8]: df.shape
Out[8]: (3897, 8)
```

```
In [9]: df.columns
Out[9]: Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'], dtype='object')
```

```
In [10]: df.duplicated().sum()
Out[10]: 766
```

```
In [11]: df.drop_duplicates(inplace = True)
```

```
In [12]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3131 entries, 0 to 2199
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   N            3131 non-null   int64
 1   P            3131 non-null   int64
 2   K            3131 non-null   int64
 3   temperature  3131 non-null   float64
 4   humidity     3131 non-null   float64
```

```
In [13]: df.describe()
```

Out[13]:

	N	P	K	temperature	humidity	ph	rainfall
count	3131.000000	3131.000000	3131.000000	3131.000000	3131.000000	3131.000000	3131.000000
mean	48.890450	54.879272	47.251038	25.294654	67.405361	6.481340	100.492838
std	36.326147	31.408798	49.505415	5.037909	24.498982	0.849883	54.107730
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	5.314507
25%	21.000000	35.000000	20.000000	22.179319	52.842901	5.928963	63.507742
50%	36.000000	54.000000	30.000000	25.400592	77.729311	6.418743	91.638957
75%	83.000000	69.000000	49.000000	28.434307	86.920705	6.962386	120.541004
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.560117

```
In [14]: df.nunique()
Out[14]: N                137
         P                117
         K                 73
         temperature     2315
         humidity        2311
         ph              2644
         rainfall        2378
         label             27
         dtype: int64
```

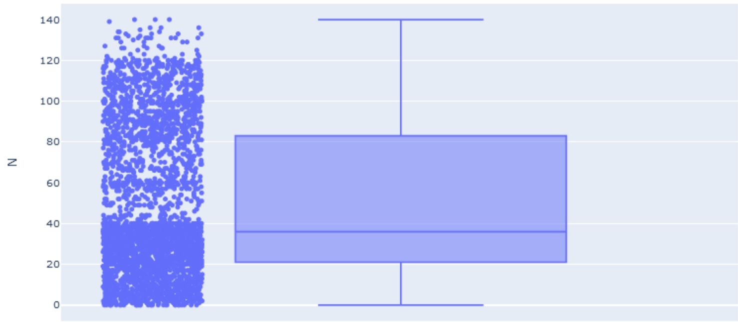
```
In [15]: df['label'].unique()
Out[15]: array(['rice', 'maize', 'Soyabeans', 'beans', 'peas', 'groundnuts',
                'cowpeas', 'banana', 'mango', 'grapes', 'watermelon', 'apple',
                'orange', 'cotton', 'coffee', 'chickpea', 'kidneybeans',
                'pigeonpeas', 'mothbeans', 'mungbean', 'blackgram', 'lentil',
                'pomegranate', 'muskmelon', 'papaya', 'coconut', 'jute'],
               dtype=object)
```

```
In [16]: df['N'].unique()
Out[16]: array([ 90,  85,  60,  74,  78,  69,  94,  89,  68,  91,  93,  77,  88,
                 76,  67,  83,  98,  66,  97,  84,  73,  92,  95,  99,  63,  62,
                 64,  82,  79,  65,  75,  71,  72,  70,  86,  61,  81,  80, 100,
                 87,  96,  40,  23,  39,  22,  36,  32,  58,  59,  42,  28,  43,
                 27,  50,  25,  31,  26,  54,  57,  49,  46,  38,  35,  52,  44,
                 24,  29,  20,  56,  37,  51,  41,  34,  30,  33,  47,  53,  45,
                 48,  13,   2,  17,  12,   6,  10,  19,  11,  18,  21,  16,   9,
                  1,   7,   8,   0,   3,   4,   5,  14,  15, 105, 108, 118, 101,
                106, 109, 117, 114, 110, 112, 111, 102, 116, 119, 107, 104, 103,
                120, 113, 115, 133, 136, 126, 121, 129, 122, 140, 131, 135, 123,
                125, 139, 132, 127, 130, 134,  55], dtype=int64)
```

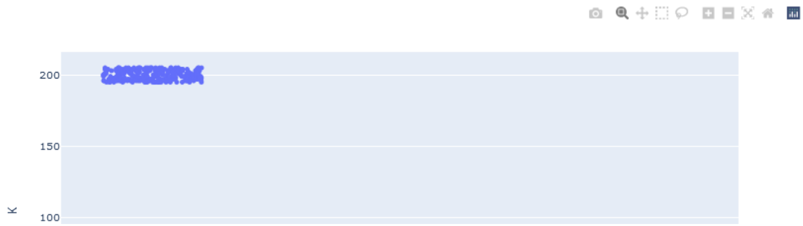
```
In [17]: df['P'].unique()
Out[17]: array([ 42,  58,  55,  35,  37,  53,  54,  46,  56,  50,  48,  38,  45,
```

In [24]:
```
import plotly.express as px
```

In [25]:
```
fig = px.box(df, y='N', points='all')
fig.show()
```

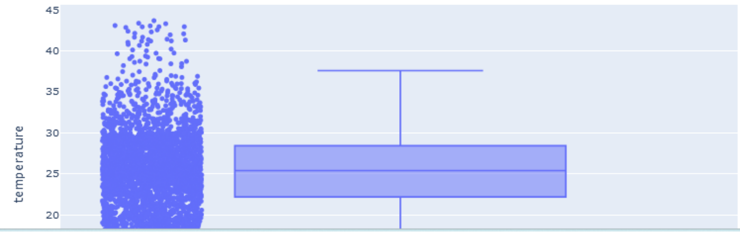


In [26]:
```
fig = px.box(df, y='K', points='all')
fig.show()
```



In [27]:
```
fig = px.box(df, y='P', points='all')
fig.show()
```

In [28]:
```
fig = px.box(df, y='temperature', points='all')
fig.show()
```



3. Preprocessing The Data

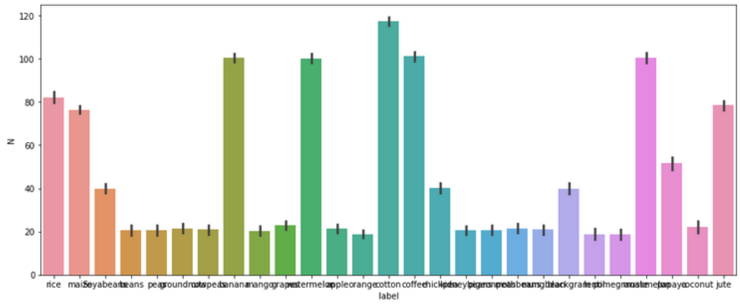
Preprocessing

In [33]:
```
data = df.copy()

# IQR
Q1 = np.percentile(data['rainfall'], 25, interpolation='midpoint')
Q3 = np.percentile(data['rainfall'], 75, interpolation='midpoint')
IQR = Q3 - Q1

print('Old shape = ', data.shape)

# Identifying outliers using IQR method
upperBound = np.where(data['rainfall'] >= Q3 + 1.5 * IQR)
lowerBound = np.where(data['rainfall'] <= Q1 - 1.5 * IQR)

# Reset index before dropping rows
data.reset_index(drop=True, inplace=True)

# Drop rows based on the indices
data.drop(upperBound[0], axis=0, inplace=True)
data.drop(lowerBound[0], axis=0, inplace=True)

print('New shape = ', data.shape)
```

```
Old shape =  (3131, 8)
New shape =  (2985, 8)
```

In [34]:
```
df = data
```

In [35]:
```
plt.figure(figsize=(15,6))
sns.barplot(y='N',x='label',data = df)
plt.show()
```



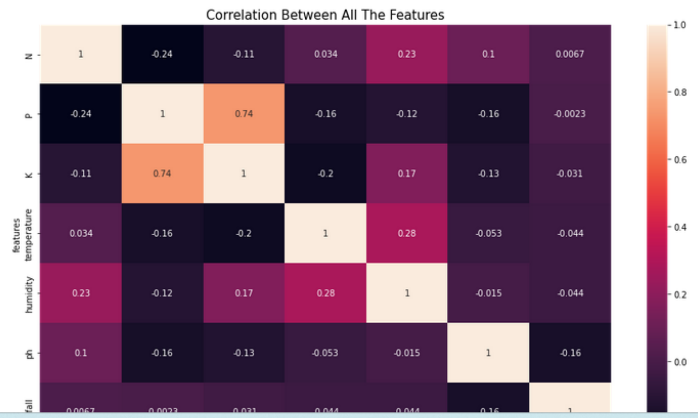
In [42]:
```
df_y = df['label']
df_x = df
df_x = df_x.drop('label', axis = 1)
```

In [43]:
```
df_x.corr()
```

Out[43]:

	N	P	K	temperature	humidity	ph	rainfall
N	1.000000	-0.240776	-0.112325	0.034084	0.230755	0.101718	0.006669
P	-0.240776	1.000000	0.735392	-0.161258	-0.116439	-0.164708	-0.002350
K	-0.112325	0.735392	1.000000	-0.195287	0.165346	-0.125126	-0.030854
temperature	0.034084	-0.161258	-0.195287	1.000000	0.281657	-0.052767	-0.044278
humidity	0.230755	-0.116439	0.165346	0.281657	1.000000	-0.015214	-0.043709
ph	0.101718	-0.164708	-0.125126	-0.052767	-0.015214	1.000000	-0.160364
rainfall	0.006669	-0.002350	-0.030854	-0.044278	-0.043709	-0.160364	1.000000

In [44]:
```
fig, ax = plt.subplots(1,1,figsize = (15,9))
sns.heatmap(df_x.corr(), annot = True)
ax.set(xlabel = 'features')
ax.set(ylabel = 'features')

plt.title('Correlation Between All The Features', fontsize = 15)
plt.show()
```



4. Building various Models

features

In [45]:
```
x = df_x
y = df_y
```

In [46]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.30, shuffle = True)
```

LightGBM

In [47]:
```
import lightgbm as lgb
modelLGB = lgb.LGBMClassifier()
modelLGB.fit(x_train, y_train)
```

Out[47]:
```
▾ LGBMClassifier
LGBMClassifier()
```

In [48]:
```
predLGB = modelLGB.predict(x_test)
```

In [49]:
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(predLGB , y_test)
print('Light GBM Model accuracy: ',accuracy)
```

```
Light GBM Model accuracy:  0.6941964285714286
```

Decision Tree

In [52]:
```
from sklearn.tree import DecisionTreeClassifier
modelDT = DecisionTreeClassifier()
modelDT.fit(x_train, y_train)
```

Out[52]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [53]:
```
predDT = modelDT.predict(x_test)
```

In [54]:
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(predDT , y_test)
print('Decision Tree Model accuracy: ',accuracy)
```

```
Decision Tree Model accuracy:  0.6953125
```

Random Forest

In [55]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
modelRF = RandomForestClassifier()
modelRF.fit(x_train , y_train)
```

Out[55]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [56]:
```
predRF = modelRF.predict(x_test)
```

In [57]:
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(predRF , y_test)
print('Random Forest Model accuracy: ',accuracy)
```

```
Random Forest Model accuracy:  0.7008928571428571
```

Support Vector Machine

In [58]:
```
from sklearn.svm import SVC
modelSVC = SVC()
modelSVC.fit(x_train, y_train)
```

Out[58]:
```
▾ SVC
SVC()
```

In [59]:
```
predSVC = modelSVC.predict(x_test)
```

In [60]:
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(predSVC , y_test)
print('Support Vector Model accuracy: ',accuracy)
```

```
Support Vector Model accuracy:  0.8180803571428571
```

Extra Trees

In [61]:
```
from sklearn.ensemble import ExtraTreesClassifier
modelETC = ExtraTreesClassifier()
modelETC.fit(x_train,y_train)
```

Out[61]:
```
▾ ExtraTreesClassifier
ExtraTreesClassifier()
```

In [62]:
```
predETC = modelETC.predict(x_test)
```

In [63]:
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(predETC , y_test)
print('Extra Trees Model accuracy: ',accuracy)
```

```
Extra Trees Model accuracy:  0.6986607142857143
```

5. Comparing all the Models

Comparison of all the Models

In [74]:
```
print("LightGBM Accuracy- ",accuracy_score(predLGB , y_test))
print("Decision Tree Accuracy- ",accuracy_score(predDT , y_test))
print("Random Forest Accuracy- ",accuracy_score(predRF , y_test))
print("Support Vector Machine Accuracy- ",accuracy_score(predSVC , y_test))
print("Extra Trees Accuracy- ",accuracy_score(predETC , y_test))
```

```
LightGBM Accuracy-  0.6941964285714286
Decision Tree Accuracy-  0.6953125
Random Forest Accuracy-  0.7008928571428571
Support Vector Machine Accuracy-  0.8180803571428571
Extra Trees Accuracy-  0.6986607142857143
```

6. Saving the model with the best accuracy

Saving The Best Model

In [75]:
```
import pickle
```

In [76]:
```
pickle.dump(modelSVC, open('modelSVC.pkl','wb'))
```

7. Creating GUI using Streamlit

```python
import streamlit as st
import joblib
import numpy as np
import pickle
import warnings
warnings.filterwarnings('ignore')
st.set_page_config(layout="wide")

model = pickle.load(open('modelSVC.pkl','rb'))

st.header('**CROP RECOMMENDATION SYSTEM**')
Nitrogen = st.slider('**N(Nitrogen):**',0.0,150.0)
Phosphorous = st.slider('**P(Phosphorous):**',0.0,150.0)
Potassium = st.slider('**K(Potassium):**',0.0,210.0)
Temperature = st.slider('**Temperature:**',5.0,50.0)
Humidity = st.slider('**Humidity:**',5.0,110.0)
ph = st.slider('**ph:**',6.0,8.0)
Rainfall = st.slider('**Amount of Rainfall:**',0.0, 1500.0)

st.markdown("""
<style>
.big-font {
    font-size: 20px !important;
}
</style>
""", unsafe_allow_html=True)

p = [[Nitrogen, Phosphorous, Potassium, Temperature, Humidity, ph, Rainfall]]
crop = model.predict(p)
st.write(f'**<p class="big-font">Recommended Crop is -** {crop[0]} </p>', unsafe_allow_html=True)

if(crop == 'rice'):
    st.markdown('**<p class="big-font">Do\'s:**</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Water Management:** Maintain proper water levels in the paddy fields.</p>', unsafe_allow_html=True)
    st.write(' **<p class="big-font">Soil Conditions:** Choose well-drained clayey soils for cultivation.</p>', unsafe_allow_html=True)
    st.write(' **<p class="big-font">Temperature:** Rice thrives in warm temperatures, so ensure the right climate.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Fertilization:** Apply nitrogenous fertilizers appropriately. </p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Dont\'s:**</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Overwatering:** Avoid excessive water as it may lead to diseases.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Poor Drainage:** Do not cultivate in poorly drained soils.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Late Planting:** Avoid late planting, as it may affect yield.</p>', unsafe_allow_html=True)
```

```python
elif crop == 'maize':
    st.write('**<p class="big-font">Do\'s:**')
    st.write('**<p class="big-font">Sunlight:** Maize requires full sunlight, so choose a sunny location</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Spacing:** Plant seeds at the recommended spacing.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Soil:** Well-drained loamy soils are ideal.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Fertilization:** Provide adequate nutrients, especially nitrogen.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Dont\'s:**</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Overcrowding:** Avoid planting too closely; give enough space between plants.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Waterlogging:** Maize doesn\'t tolerate waterlogged conditions.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Poor Soil:** Avoid poorly-drained or acidic soils.</p>', unsafe_allow_html=True)

elif crop == 'Soybeans':
    st.write('**<p class="big-font">Do\'s:**</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Crop Rotation:** Practice crop rotation for better yield</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Inoculation:** Use nitrogen-fixing bacteria for soil inoculation.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Planting Time:**  Plant during the recommended season.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Weed Control:**  Implement effective weed control measures.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Dont\'s:**')
    st.write('**<p class="big-font">Overcrowding:** Plant at recommended spacing to prevent overcrowding.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Late Planting:** Avoid late planting; soybeans are sensitive to day length.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Poor Drainage:** Soybeans prefer well-drained soils.</p>', unsafe_allow_html=True)

elif crop == 'beans':
    st.write('**<p class="big-font">Do\'s:**</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Support:** Provide support for climbing varieties.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Watering:** Maintain consistent soil moisture.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Spacing:** Plant seeds at the recommended spacing.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Fertilization:** Apply balanced fertilizers.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Dont\'s:**</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Overwatering:** Beans are susceptible to waterlogging; avoid overwatering.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Late Planting:** Plant on time for optimal yield.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Neglecting Weeds:** Ensure proper weed control.</p>', unsafe_allow_html=True)

elif crop == 'peas':
    st.write('**<p class="big-font">Do\'s:**</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Support:** Use supports for climbing varieties.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Cool Season:** Plant during cool seasons.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Well-Drained Soil:** Peas prefer well-drained soil.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Mulching:** Mulch to retain soil moisture.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Dont\'s:**</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Overwatering:** Peas are sensitive to waterlogging; avoid excessive watering.</p>', unsafe_allow_html=True)
```

```python
elif crop == 'cotton':
    st.write('**<p class="big-font">Do\'s:**</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Spacing:** Plant seeds at the recommended spacing.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Pest Control:** Implement pest control measures.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Well-Drained Soil:** Cotton prefers well-drained loamy soil.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Dont\'s:**</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Overwatering:** Cotton is sensitive to waterlogging.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Late Planting:** Plant on time for optimal yield.</p>', unsafe_allow_html=True)

elif crop == 'coffee':
    st.write('**<p class="big-font">Do\'s:**</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Altitude:** Choose the right altitude for coffee cultivation.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Shade:** Provide shade for young coffee plants.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Fertilization:** Apply organic fertilizers</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Pest Control:** Monitor and control pests.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Dont\'s:**</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Overwatering:** Maintain proper soil moisture without overwatering.</p>', unsafe_allow_html=True)
    st.write('**<p class="big-font">Waterlogging:** Coffee plants are sensitive to waterlogged conditions.</p>', unsafe_allow_html=True)

import base64

def get_base64(bin_file):
    with open(bin_file, 'rb') as f:
        data = f.read()
    return base64.b64encode(data).decode()

def set_background(png_file):
    bin_str = get_base64(png_file)
    page_bg_img = '''
    <style>
    .stApp {
    background-image: url("data:image/png;base64,%s");
    background-size: cover;
    }
    </style>
    ''' % bin_str
    st.markdown(page_bg_img, unsafe_allow_html=True)

set_background("farming2.jpg")
```

Chapter 5

Need of Crop Recommendation System

- **Increased yields:** By planting the right crops in the right conditions, farmers can increase their yields.
- **Improved profitability:** Farmers can make more money by planting crops that are in high demand and that are likely to sell for a good price.
- **Reduced risk:** Crop recommendation systems can help farmers to avoid planting crops that are not suited to the area, which can reduce the risk of crop failure.
- **Improved resource management:** Crop recommendation systems can help farmers to make the most of their land, water, and other resources.

Chapter 6

Applications of Crop Recommendation System

- **Enhanced Crop Selection :-** The system analyses various factors and then recommends the most suitable crop. This data-driven approach helps the farmers to avoid unsuitable choices and maximize their yield.
- **Increased Profitability :-** By choosing the right crops and managing resources efficiently, farmers can potentially boost their income and improve their livelihoods.
- **Promoting Sustainable Agriculture :-** The system can be integrated with other models to guide farmers towards choosing crops and practices that are less taxing on the environment. This includes recommending crops with lower water requirements, encouraging soil conservation practices, and suggesting integrated pest management techniques.
- **Reduced Risk of Crop Failure :-** Early warnings about potential pest outbreaks, diseases, or unfavourable weather conditions can be integrated into CRS, allowing farmers to take preventative measures and minimize crop losses.

Chapter 7

Conclusion

This project has delved into the fertile ground of precision agriculture, exploring the potential of data-driven decision-making for optimal crop selection. We recognized the limitations of traditional methods, often reliant on intuition and local knowledge, that can fall short in the face of climatic uncertainties and varying soil conditions.

Our response was to cultivate a personalized crop recommendation system, empowered by analysis of five key environmental factors: Nitrogen, Phosphorus, Potassium (NPK) levels, temperature, pH, rainfall, and humidity. By feeding this rich data into five diverse algorithms, we embarked on a rigorous selection process, ultimately identifying the [mention the best performing algorithm by name] as the champion of accurate and reliable recommendations.