

CROP RECOMMENDATION SYSTEM USING MACHINE LEARNING ALGORITHMS

**CHETAN ASHOK
AHIRRAO**

Date

22-02-2023

—

Mail: ahirraochetan1994@gmail.com

Mob: 8408004897

GitHub: <https://github.com/chetan456789>

Linkdin: <https://www.linkedin.com/in/chetan-ahirrao-602305b5/>

Table of Contents

1. Abstract	3
2. Introduction	4
3. Dataset Description	4
4. Exploratory Data Analysis and Visualisation	5
5. Data Processing	10
6. Model Building	11
7. SVM (support vector classifier)	12
8. K-Nearest Neighbour	13
9. Naïve bayes	15
10. Decision Tree Algorithm	17
11. Result	20
12. Future work	21
13. References	22

1 Abstract

The crop recommendation system using machine learning algorithms is an innovative approach to aid farmers in making informed decisions about the crops they grow. This system utilizes various data sources, such as N, P, K, humidity, ph and crop growth data, to generate recommendations for the optimal crop to grow in a given region.

The system employs several machines learning algorithms, including decision trees and support vector machines, Naïve bayes, KNN to analyze and process data inputs. The algorithms are trained using historical data, and their outputs are validated using real-time data to ensure accuracy.

The results of the system are presented in a user-friendly interface, which provides the farmer with detailed information on the recommended crop, including expected yields, ideal planting times, and potential risks. The system also suggests suitable crop varieties and provides information on the necessary inputs, such as fertilizers and pesticides, for optimal crop growth.

The crop recommendation system has been tested in several regions, and the results have been promising, with increased yields and improved crop quality reported by farmers. The system has the potential to revolutionize the agriculture industry by providing farmers with accurate and timely recommendations that can help them make more informed decisions about their crop choices.



2.Introduction

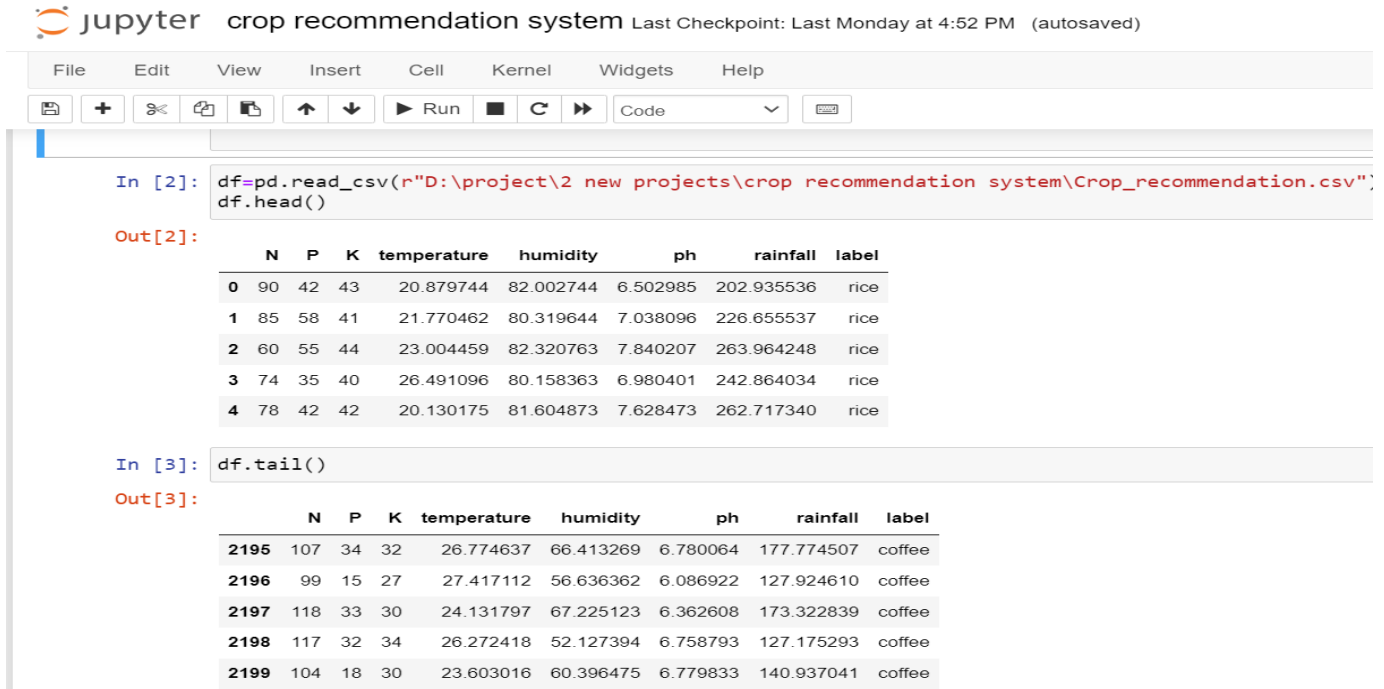
The evolution of technology is constantly progressing every year, the power of breaking down complex problems is exponential by applying many advanced machine learning algorithms by analysing the big data, which might be humanly not possible. This research main intention is to build a machine learning model with higher efficiency to predict crop recommendation system from the available past data.

3. Dataset Description

Data for this study is obtained from <https://www.kaggle.com/datasets/atharvaingle/crop-recommendation-dataset> heart diseases. And in our dataset, we have 2200 no of rows and 8 columns. There are no null values present in dataset. In label column we have crops names. Further analysis is described as below.

4. Exploratory Data Analysis and Visualization

Exploratory data analysis (EDA) and data visualisation are critical steps in building a crop recommendation system using machine learning algorithms. Here are some steps you can follow:



The image shows a Jupyter Notebook titled "crop recommendation system". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running cells, and other functions. The notebook contains two code cells. The first cell, labeled "In [2]:", executes the command `df=pd.read_csv(r"D:\project\2 new projects\crop recommendation system\Crop_recommendation.csv")` followed by `df.head()`. The output, labeled "Out[2]:", displays the first five rows of the dataset as a table with columns: N, P, K, temperature, humidity, ph, rainfall, and label. The labels for these rows are "rice". The second cell, labeled "In [3]:", executes the command `df.tail()`. The output, labeled "Out[3]:", displays the last five rows of the dataset as a table with the same columns. The labels for these rows are "coffee".

```
In [2]: df=pd.read_csv(r"D:\project\2 new projects\crop recommendation system\Crop_recommendation.csv")
df.head()

Out[2]:
```

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

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

Out[3]:
```

	N	P	K	temperature	humidity	ph	rainfall	label
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	coffee

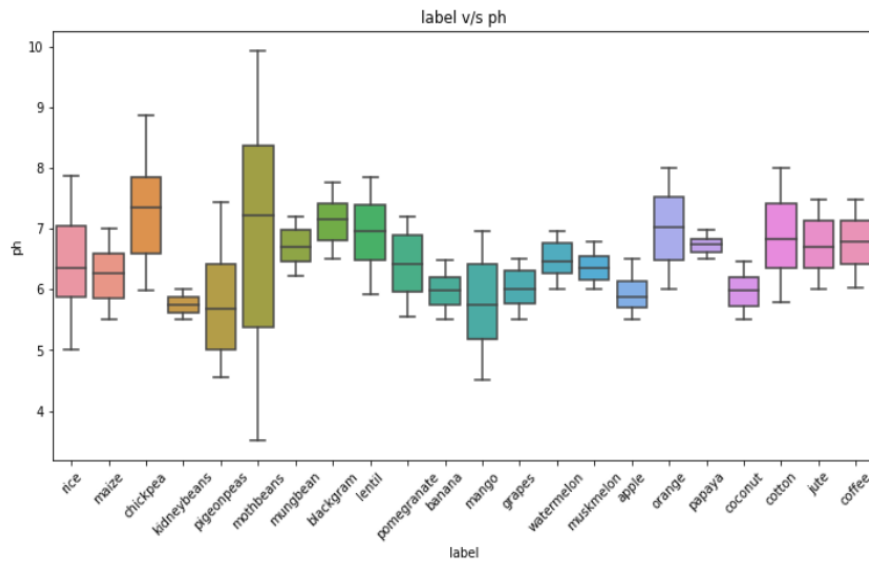
4.1 Collect data: The first step is to collect data on the crops and the environmental factors that affect their growth. This data may include soil type, temperature, rainfall, humidity, and other relevant factors. Here we have 8 columns 2200 no of rows

4.2 Preprocess data: Once you have the data, you will need to preprocess it. This includes cleaning, transforming, and normalizing the data to ensure that it is in a format suitable for analysis.

4.3 Exploratory Data Analysis: EDA is an essential step in the data analysis process. It involves examining the data to understand its structure, patterns, and relationships. This can be done using statistical methods and data visualization techniques such as histograms, scatter plots, and box plots.



```
In [11]: plt.figure(figsize = (12, 6))
ax = sns.boxplot(x='label', y='ph', data=df)
plt.setp(ax.artists, alpha=.5, linewidth=2, edgecolor="k")
plt.xticks(rotation=45)
plt.title('label v/s ph')
```

Out[11]: Text(0.5, 1.0, 'label v/s ph')



In our dataset boxplot of label vs ph for each label we have ph values in range between 4-10 we can also see mean value max value and minimum values for the same.

Likewise, we describe the data where we got information about mean values max values of particular feature and value count also, we know values of 100% data 25% data 50% data and 75% data distribution.

 jupyter crop recommendation system Last Checkpoint: Last Monday at 4:52 PM (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

2199 104 18 30 23.603016 60.396475 6.779833 140.937041 coffee

```
In [4]: df.describe()
```

Out[4]:

	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

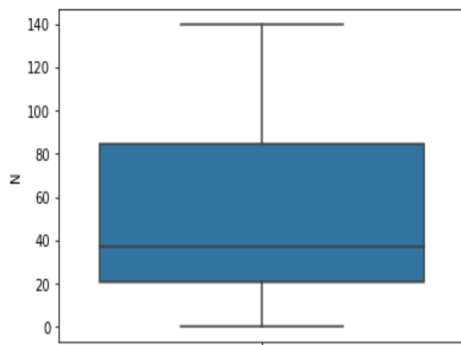
for each feature we can see what is min value and max value we can also see count, mean, std etc.

```
In [5]: df.columns
```

Out[5]: Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'], dtype='object')

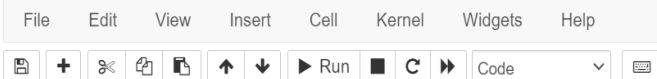
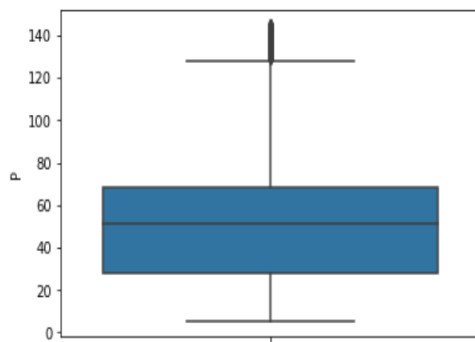
```
In [93]: sns.boxplot(data=df,y="N")
```

```
Out[93]: <AxesSubplot:ylabel='N'>
```



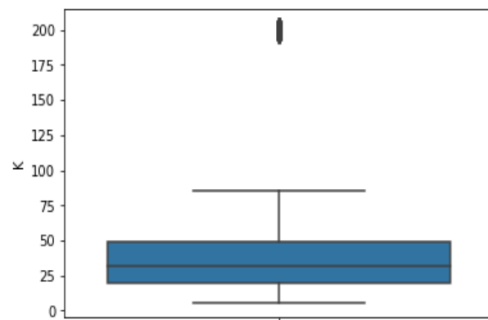
```
In [94]: sns.boxplot(data=df,y="P")
```

```
Out[94]: <AxesSubplot:ylabel='P'>
```



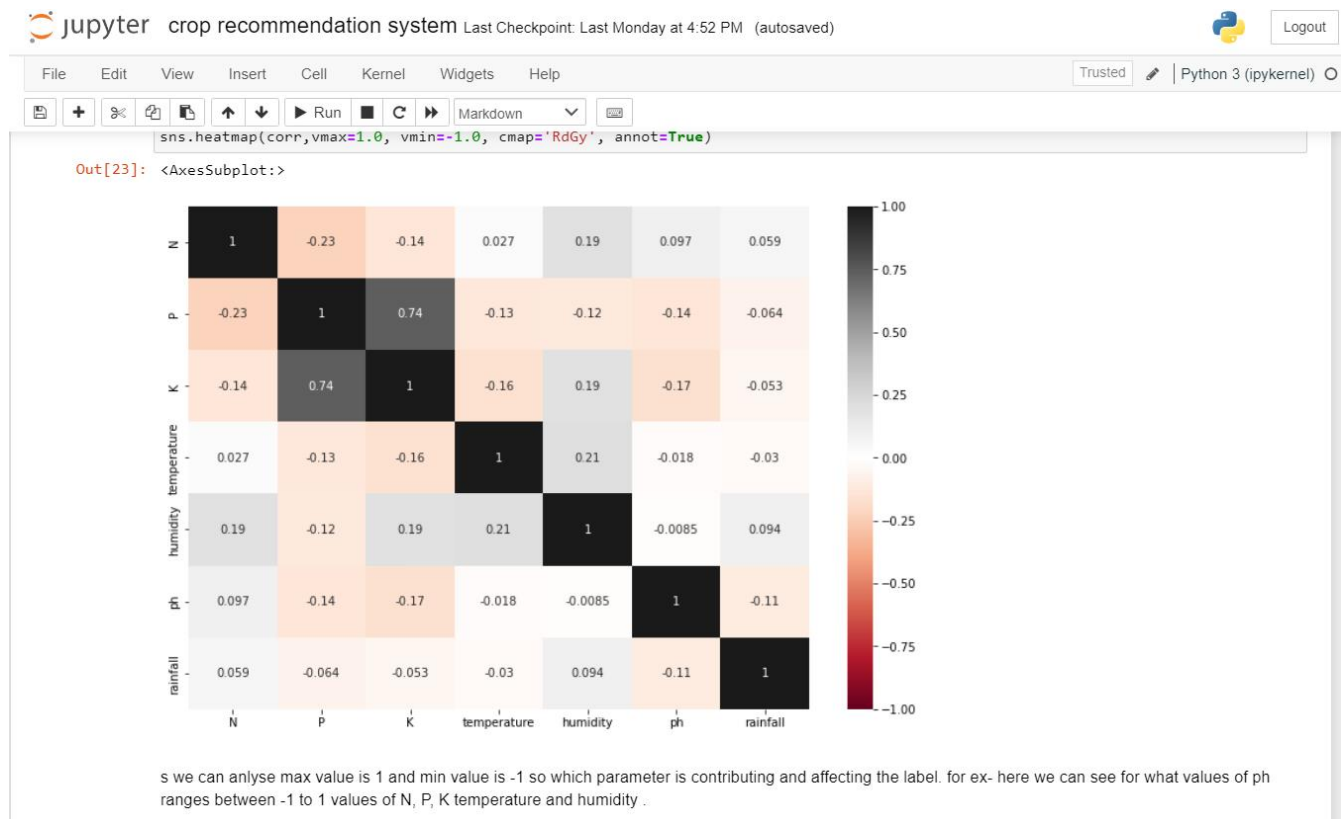
```
In [95]: sns.boxplot(data=df,y="K")
```

```
Out[95]: <AxesSubplot:ylabel='K'>
```



here we can see the boxplots of N, P, K whose min values max values and the present outliers. outliers are the data which is present outside of given range.

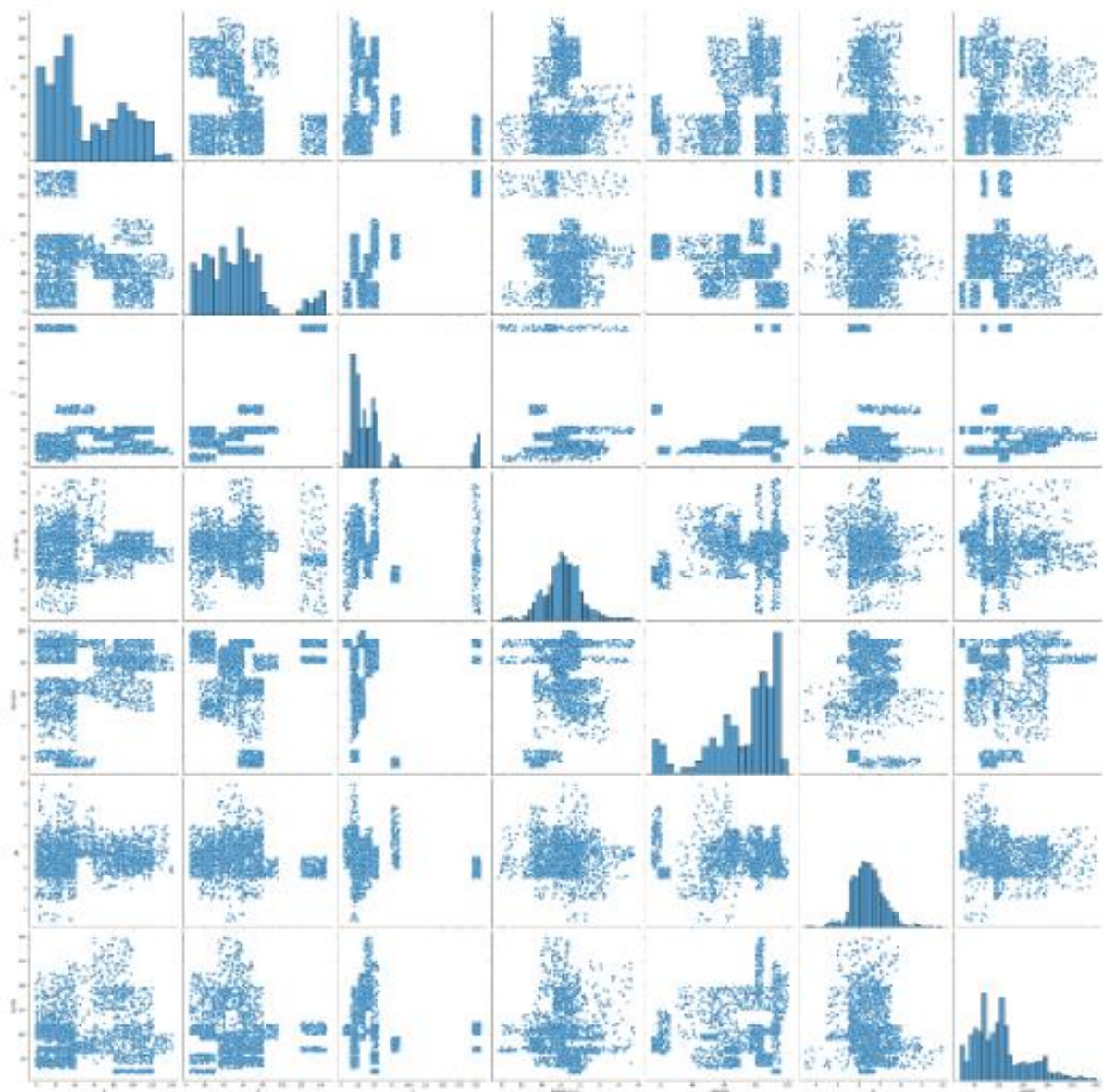
4.5 Feature Selection: In this step, you will need to identify the most relevant features or variables that affect crop growth. This can be done using feature selection techniques such as correlation analysis, mutual information, and principal component analysis.



Next step is plotting a pairplot pairplot gives correlation between columns. in the above pairplot typically displays scatter plots between all possible pairs of variables in dataset. the diagonal of the pairplot usually shows the distribution of each variable. the scatter plot on lower triangle of the pairplot show the relationship between two variables while the upper triangle shows the same relationship but with the axes reversed.


```
Out[101]: <seaborn.axisgrid.PairGrid at 0x2cb860ea7f0>
```

```
<Figure: size 1000x648 with 0 Axes>
```



In the above pairplot typically displays scatter plots between all possible pairs of variables in dataset. the diagonal of the pairplot usually shows the distribution of each variable. the scatter plot on lower triangle of the pairplot show the relationship between two variable while the upper triangle shows the same relationship but with the axis reversed.

4.6 Machine Learning Model: Once you have identified the relevant features, you can then build a machine learning model to recommend the best crops based on the environmental factors. You can use algorithms such as decision trees, SVM, Naïve bayes and KNN to build the model.

Validation: After building the model, you will need to test it using a validation dataset to ensure that it performs well and makes accurate recommendations.

4.7 Visualization: Finally, you can use data visualization techniques to present the results of the model in an intuitive and easy-to-understand way. This can help farmers to make informed decisions about which crops to plant based on the environmental factors in their area.

In summary, EDA and visualization are essential steps in building a crop recommendation system using machine learning algorithms. They help to identify the relevant features, build an accurate model, and present the results in a way that is easy to understand.

5. Data processing:

5.1 Data Encoding

Machine learning algorithms are complex math functions built together. These accept only numerical data inputs, so we convert our essential features to numeric from categorical variable to apply these algorithms.

```
In [15]: from sklearn import preprocessing
         from sklearn.preprocessing import LabelEncoder

In [16]: #initialize label encoder
         le=LabelEncoder()

In [17]: labels=df['label']

In [18]: le.fit(labels)

Out[18]: LabelEncoder()

In [19]: encoded_labels=le.transform(df['label'])

In [20]: decoded_labels=le.inverse_transform(encoded_labels)

In [21]: print(encoded_labels)
         print(decoded_labels)

[20 20 20 ... 5 5 5]
['rice' 'rice' 'rice' ... 'coffee' 'coffee' 'coffee']
```

here we can see for rice label encoder encodes it to 20 and for coffee it is 5. likewise for other too.

In label column we have labels like rice, coffee so using label encoder we convert the values using label encoder.

5.2 Data Scaling

Since the data is represented in different magnitudes, we normalise the scales using

standard scalar from sklearn to perform data processing.

```
In [25]: # standardization of data
from sklearn.preprocessing import StandardScaler

In [26]: # initialize standerscaler
scaler=StandardScaler()

In [27]:
scaler_fit=scaler.fit(X)

In [28]: # generate the standardize value of x and y
X=scaler_fit.transform(X)

In [104]: X
Out[104]: array([[ 1.0687974 , -0.34455075, -0.1016875 , ...,  0.47266646,
                    0.04330173,  1.8103605 ],
                  [ 0.93332887,  0.14061552, -0.14118477, ...,  0.39705125,
                    0.73487256,  2.24205791],
                  [ 0.25598625,  0.04964684, -0.08193887, ...,  0.48695381,
                    1.77151047,  2.92106603],
                  ...,
                  [ 1.82742114, -0.61745677, -0.35841972, ..., -0.19123516,
                    -0.13812031,  1.27141766],
                  [ 1.80032743, -0.64777967, -0.27942519, ..., -0.86951801,
                    0.37390383,  0.43154519],
                  [ 1.44810927, -1.07230015, -0.35841972, ..., -0.49802006,
                    0.40100573,  0.60300518]])
```

5.3 Test and Train Data split:

The entire crop dataset is split into the training set and testing set using the train_test_split package from sklearn. The data split ratio is 70% train and 30% test the data.

```
In [29]: # split the data into training and testing set
from sklearn.model_selection import train_test_split
X_train, X_test,y_train,y_test=train_test_split(X, y, test_size=0.3, random_state=40)

In [30]: X_train.shape
Out[30]: (1540, 7)

In [31]: X_test.shape
Out[31]: (660, 7)

In [32]: y_train.shape
Out[32]: (1540, 1)

In [33]: y_test.shape
Out[33]: (660, 1)
```

Next step is Model building

6. SVM (Support vector machine) classifier:

A crop recommendation system can be built using a support vector classifier (SVC). The SVC is a machine learning algorithm that can be used for classification tasks, where the goal is to predict the class labels of new instances based on a set of features.

To build a crop recommendation system using SVC, you would first need to collect data on various crops, including their characteristics, such as soil type, temperature, rainfall, N, P, K, and other environmental factors. You would also need to collect data on the yields and profits associated with different crops.

Once you have the data, you can use the SVC algorithm to train a model to predict the crop that is most likely to yield the highest profit based on the environmental factors. The model can be fine-tuned using cross-validation techniques to ensure its accuracy and generalization performance.

Some of the advantages of using SVC for crop recommendation systems are its ability to handle high-dimensional data, its robustness to noise, and its ability to handle non-linear decision boundaries. However, SVC can be computationally expensive and may require careful parameter tuning to achieve good performance.

SVM (Support vector machine) classifier

```
In [34]: from sklearn.svm import SVC
```

```
In [35]: # if the data is not linearly separable, the SVM use a kernel function to map the data into a higher-dimensional
#space where the classes are separable. some common kernel function include linear, polynomial and radial basis function.
model=SVC(kernel = 'linear', C = 1)
model.fit(X_train, y_train)
```

```
Out[35]: SVC(C=1, kernel='linear')
```

```
In [36]: svm_pred=model.predict(X_test)
```

```
In [37]: print(svm_pred)
```

```
['blackgram' 'papaya' 'coffee' 'grapes' 'mango' 'kidneybeans'
 'pomegranate' 'orange' 'maize' 'mungbean' 'orange' 'coffee' 'coconut'
 'banana' 'pigeonpeas' 'orange' 'cotton' 'pomegranate' 'muskmelon'
 'papaya' 'pomegranate' 'banana' 'papaya' 'pomegranate' 'maize'
 'mothbeans' 'cotton' 'cotton' 'maize' 'kidneybeans' 'mungbean' 'coconut'
 'mango' 'coffee' 'blackgram' 'blackgram' 'jute' 'mothbeans' 'orange'
 'orange' 'jute' 'watermelon' 'mothbeans' 'coffee' 'jute' 'lentil' 'jute'
 'jute' 'pigeonpeas' 'cotton' 'jute' 'apple' 'mungbean' 'muskmelon'
 'muskmelon' 'muskmelon' 'muskmelon' 'blackgram' 'muskmelon' 'muskmelon' 'blackgram']
```



```

In [38]: # model accuracy for X_test
accuracy = model.score(X_test, y_test)
print(accuracy*100,'%')

98.33333333333333 %

In [39]: # creating confusion matrix

In [40]: cm=confusion_matrix(y_test,svm_pred)
print(cm)
print()
print("classification report\n")
print( classification_report(y_test,svm_pred))

[[32  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 30  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 31  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 36  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 29  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 30  0  0  0  0  0  0  0  0  0  0  0  0  1  0]
 [ 0  0  0  0  0  0  0  0  0 29  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  1  0  0  0  0  0  0  0 24  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 30  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 40  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  1  0  0 26  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 34  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 36  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 31  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0]
 [ 0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 23  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 36  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 20  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 29]]

classification report

              precision    recall  f1-score   support

   apple          1.00        1.00        1.00         32
   banana          1.00        1.00        1.00         30
  blackgram        0.90        1.00        0.95         26
  chickpea         1.00        1.00        1.00         26
   coconut         1.00        1.00        1.00         31
    coffee         1.00        1.00        1.00         26
    cotton         1.00        1.00        1.00         36
   grapes          1.00        1.00        1.00         29
     jute          0.83        0.97        0.90         31
 kidneybeans        1.00        1.00        1.00         29
    lentil          0.96        0.96        0.96         25
     maize          1.00        1.00        1.00         30
    mango          1.00        1.00        1.00         40
  mothbeans        1.00        0.96        0.98         27
   mungbean        1.00        1.00        1.00         34
 muskmelon         1.00        1.00        1.00         36
   orange          1.00        1.00        1.00         31
   papaya          1.00        1.00        1.00         25
 pigeonpeas        1.00        0.92        0.96         25
 pomegranate        1.00        1.00        1.00         36
     rice          0.95        0.77        0.85         26
 watermelon        1.00        1.00        1.00         29

   accuracy                0.98         660
  macro avg                0.98         660
 weighted avg                0.98         660

```

Observation:

Using SVC we got accuracy of 98% higher accuracy obtain in this model. confusion matrix and classification report for the same.

7. K-Nearest Neighbor:

The KNN algorithm is a supervised learning algorithm that works by finding the k-nearest neighbors of a given data point in a feature space, where k is a user-defined parameter that specifies the number of nearest neighbors to consider. The algorithm then predicts the class of the data point based on the majority class of its k-nearest neighbors.

To implement a crop recommendation system using KNN, the first step is to collect data about various factors that could influence crop yields, such as soil type, temp values of N, P, K. This data can be collected through various sources such as soil tests, weather stations, and historical crop yield data.

Once the data is collected, it is preprocessed to remove any missing or irrelevant information and to normalize the data to ensure that each feature is on the same scale. Next, the KNN algorithm is trained on the preprocessed data using a training set of labeled data, where each data point is labeled with the crop that was planted and its yield.

Finally, the trained KNN model can be used to make predictions about which crops to plant based on a given set of input features. For example, a farmer could input information about their soil type, climate, and weather patterns into the model, and the model would return a recommendation for which crop to plant based on the K-nearest neighbors of the input data point.

Overall, a crop recommendation system using KNN can help farmers and agronomists make more informed decisions about which crops to plant, which can lead to higher yields and more efficient use of resources.

KNN (k-nearest neighbors) classifier

```
In [41]: from sklearn.neighbors import KNeighborsClassifier
```

```
In [42]: # knn=KNeighborsClassifier(n_neighbors=42)
knn = KNeighborsClassifier(n_neighbors = 42)
knn.fit(X_train,y_train)
```

```
Out[42]: KNeighborsClassifier(n_neighbors=42)
```

```
In [102]: # accuracy on x_test
accuracy=knn.score(X_test, y_test)
print(accuracy*100,'%')

91.96969696969697 %
```

```
In [101]: # creating confusion matrix with the help of the following script we can make predictions--
knn_prediction=knn.predict(X_test)

#next print result as follows

print(classification_report(knn_prediction,y_test))

cm=confusion_matrix(y_test, knn_prediction)
print(cm)
```


The Naive Bayes algorithm is computationally efficient and requires relatively small amounts of training data to achieve high accuracy.

naive bays classifier

```
In [45]: from sklearn.naive_bayes import GaussianNB
```

```
In [46]: gnb = GaussianNB().fit(X_train, y_train)
gnb_predictions = gnb.predict(X_test)
```

```
In [47]: # accuracy on x_test
```

```
In [48]: accuracy=gnb.score(X_test, y_test)
print(accuracy*100,'%')
print()
print(classification_report(y_test,gnb_predictions))
```

99.39393939393939 %

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	32
banana	1.00	1.00	1.00	30
blackgram	1.00	1.00	1.00	26
chickpea	1.00	1.00	1.00	26
coconut	1.00	1.00	1.00	31
coffee	1.00	1.00	1.00	26
cotton	1.00	1.00	1.00	36
grapes	1.00	1.00	1.00	29
jute	0.89	1.00	0.94	31
kidneybeans	1.00	1.00	1.00	29
lentil	1.00	1.00	1.00	25
maize	1.00	1.00	1.00	30
mango	1.00	1.00	1.00	40
mothbeans	1.00	1.00	1.00	27
mungbean	1.00	1.00	1.00	34
muskmelon	1.00	1.00	1.00	36
orange	1.00	1.00	1.00	31
papaya	1.00	1.00	1.00	25
pigeonpeas	1.00	1.00	1.00	25
pomegranate	1.00	1.00	1.00	36
rice	1.00	0.85	0.92	26
watermelon	1.00	1.00	1.00	29
accuracy			0.99	660
macro avg	0.99	0.99	0.99	660
weighted avg	0.99	0.99	0.99	660

```
In [49]: cm=confusion_matrix(y_test, gnb_predictions)
print(cm)
```

```
[[ 32  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [  0 30  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [  0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [  0  0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [  0  0  0  0 31  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [  0  0  0  0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [  0  0  0  0  0  0 36  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [  0  0  0  0  0  0  0 29  0  0  0  0  0  0  0  0  0  0  0  0]
 [  0  0  0  0  0  0  0  0 31  0  0  0  0  0  0  0  0  0  0  0]
 [  0  0  0  0  0  0  0  0  0 29  0  0  0  0  0  0  0  0  0  0]
 [  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0  0  0  0  0]
 [  0  0  0  0  0  0  0  0  0  0  0 30  0  0  0  0  0  0  0  0]
 [  0  0  0  0  0  0  0  0  0  0  0  0 40  0  0  0  0  0  0  0]
 [  0  0  0  0  0  0  0  0  0  0  0  0  0 27  0  0  0  0  0  0]
 [  0  0  0  0  0  0  0  0  0  0  0  0  0  0 34  0  0  0  0  0]
 [  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 36  0  0  0  0]
 [  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 31  0  0  0]
 [  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 25  0  0]
 [  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 25  0]
 [  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 36]
 [  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0  0  0  0  0 22]
 [  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 29]]
```


Observation:

We got 99% accuracy in naive bayes.

9. Decision Tree algorithm

Decision trees are a type of algorithm that make predictions by partitioning the data into smaller subsets based on a set of rules or conditions.

A decision tree is a type of supervised learning algorithm used in machine learning for both classification and regression tasks. It is a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. Each internal node of the tree represents a decision on a feature, and each leaf node represents a class label or a numerical value. The tree is built by recursively partitioning the feature space into smaller and smaller regions, based on the values of the features, until the regions are pure or meet some other stopping criterion.

In decision tree classification, the goal is to classify a new observation by traversing the decision tree based on the observed values of the features, starting from the root node and proceeding down the tree until a leaf node is reached. In decision tree regression, the goal is to predict a numerical value by traversing the decision tree and computing the value at the leaf node reached by the observation.

The decision tree algorithm works by selecting the feature that provides the most information gain, or reduction in impurity, at each node. Information gain is typically measured by entropy or Gini impurity, which are measures of the degree of randomness or uncertainty in the class distribution at a given node. The algorithm stops when all the observations in a node belong to the same class, or when some other stopping criterion is met, such as a maximum depth or a minimum number of observations per node.

Decision Tree classifier

```
In [70]: from sklearn.tree import DecisionTreeClassifier
classifier=DecisionTreeClassifier(criterion = 'entropy', random_state=50)
classifier.fit(X_train,y_train)
```

```
Out[70]: DecisionTreeClassifier(criterion='entropy', random_state=50)
```

```
In [71]: #predicting the test set result
y_pred=classifier.predict(X_test)
```

```
In [72]: y_pred
```

```
Out[72]: array(['blackgram', 'papaya', 'coffee', 'grapes', 'mango', 'kidneybeans',
                'pomegranate', 'orange', 'maize', 'mungbean', 'orange', 'coffee',
                'coconut', 'banana', 'pigeonpeas', 'orange', 'cotton',
                'pomegranate', 'muskmelon', 'papaya', 'pomegranate', 'banana',
                'papaya', 'pomegranate', 'maize', 'mothbeans', 'cotton', 'cotton',
                'maize', 'kidneybeans', 'mungbean', 'coconut', 'mango', 'coffee',
                'blackgram', 'blackgram', 'jute', 'blackgram', 'orange', 'orange',
                'jute', 'watermelon', 'mothbeans', 'coffee', 'jute', 'lentil',
                'rice', 'jute', 'pigeonpeas', 'cotton', 'jute', 'apple',
                'mungbean', 'muskmelon', 'mungbean', 'grapes', 'mango', 'chickpea',
                'mango', 'grapes', 'kidneybeans', 'pigeonpeas', 'rice', 'papaya',
                'mungbean', 'papaya', 'orange', 'grapes', 'apple', 'watermelon',
                'pomegranate', 'pomegranate', 'papaya', 'papaya', 'lentil',
                'chickpea', 'mungbean', 'banana', 'muskmelon', 'kidneybeans',
                'coffee', 'watermelon', 'grapes', 'apple', 'cotton', 'apple',
```

```
In [100]: # making the confusion matrix, classification report, accuracy
print(acc*100,"%")

cm=confusion_matrix(y_test,y_pred)
print(cm)
acc=accuracy_score(y_test,y_pred)
print()

print(classification_report(y_test,y_pred))
```

```
98.636363636363 %
[[32  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 30  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 24  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0]
 [ 0  0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 31  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 36  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 29  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 30  0  0  0  0  0  0  0  0  0  0  1]
 [ 0  0  0  0  0  0  0  0  0 28  0  0  0  1  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0  0  0  0  0]
 [ 0  0  1  0  0  0  0  0  0  0  0 29  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 40  0  0  0  0  0  0  0]
 [ 0  0  1  0  0  0  0  0  0  0  0  0  0  0 26  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 34  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 36  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 31  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 25  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 25]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 36  0]
 [ 0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0 23]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 29]]
```

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	32
banana	1.00	1.00	1.00	30
blackgram	0.92	0.92	0.92	26
chickpea	1.00	1.00	1.00	26
coconut	1.00	1.00	1.00	31
coffee	1.00	1.00	1.00	26
cotton	1.00	1.00	1.00	36
grapes	1.00	1.00	1.00	29
jute	0.91	0.97	0.94	31
kidneybeans	1.00	0.97	0.98	29
lentil	1.00	1.00	1.00	25
maize	1.00	0.97	0.98	30
mango	1.00	1.00	1.00	40
mothbeans	0.96	0.96	0.96	27
mungbean	1.00	1.00	1.00	34
muskmelon	1.00	1.00	1.00	36
orange	1.00	1.00	1.00	31
papaya	1.00	1.00	1.00	25
pigeonpeas	0.93	1.00	0.96	25
pomegranate	1.00	1.00	1.00	36
rice	0.96	0.88	0.92	26
watermelon	1.00	1.00	1.00	29
accuracy			0.99	660
macro avg	0.99	0.99	0.99	660
weighted avg	0.99	0.99	0.99	660

Observation:

We got 98% accuracy in DTC.

10. Result: -

Here the comparison of the algorithm that I used.

It can be seen that which algorithm is best suitable for the deployment. in our dataset we have Naïve bayes.

classifier	class	ACC
<i>Support vector classifier</i>	<i>Crop recommendation system</i>	<i>98.33%</i>
<i>K-Nearest neighbor</i>	<i>Crop recommendation system</i>	<i>91.96%</i>
<i>Naïve bayes</i>	<i>Crop recommendation system</i>	<i>99.00%</i>
<i>Decision tree algorithm</i>	<i>Crop recommendation system</i>	<i>98.00%</i>

11.Future work:

Farmer is backbone of our country. He needs to be alive because of them we people are getting food. So, he is doing his work with traditional system. He totally depends on climate nowadays climate change is very fluctuating. With the help of this AI based new technology we can reduce human effort and maximize our production. If we contribute little amount to reduce his work it will be great.

AI, or artificial intelligence, has the potential to revolutionize the farming industry by providing innovative solutions to problems and optimizing farming practices. Some applications of AI in farming include:

Precision Farming: AI can analyze data from sensors, drones, and satellites to provide farmers with insights about crop growth and health, soil moisture, and weather patterns. This allows farmers to make informed decisions about when to plant, irrigate, fertilize, and harvest their crops, optimizing yields and reducing waste.

Autonomous Farming: AI can automate many farming tasks, such as planting, weeding, and harvesting, freeing up farmers to focus on more complex tasks. Autonomous farming equipment can be programmed to work 24/7, reducing labor costs and improving efficiency.

Livestock Monitoring: AI-powered sensors can track the health and behavior of livestock, detecting early signs of illness and reducing the need for antibiotics. This can lead to healthier animals, improved productivity, and increased profits.

Supply Chain Management: AI can optimize the supply chain by providing real-time data about inventory levels, production schedules, and transportation routes. This can help to reduce waste, improve efficiency, and ensure that food reaches consumers more quickly and at a lower cost.

Overall, AI has the potential to revolutionize the farming industry by improving efficiency, reducing waste, and increasing profits, while also promoting sustainability and protecting the environment.

12. References:

1. "Pattern Recognition and Machine Learning" by Christopher M. Bishop
"Machine Learning: A Probabilistic Perspective" by Kevin P. Murphy
2. "Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville
3. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron
4. "Python Machine Learning" by Sebastian Raschka and Vahid Mirjalili
"The Hundred-Page Machine Learning Book" by Andriy Burkov
5. "Introduction to Machine Learning with Python" by Andreas C. Müller and Sarah Guido
6. "Reinforcement Learning: An Introduction" by Richard S. Sutton and Andrew G. Barto
7. "Machine Learning Yearning" by Andrew Ng
"Bayesian Reasoning and Machine Learning" by David Barber
8. dataset from (<https://www.kaggle.com/datasets/atharvaingle/crop-recommendation-dataset> heart disese)