SWAMI VIVEKANANDA INSTITUTE
OF SCIENCE & TECHNOLOGY
Dakshin Gobindapur Rd, Rajpur Sonarpur, Kolkata

Deptartment of Computer Science Engineering

CROP PREDICTION SYSTEM

Euphoria genx
EXPLORE YOURSELF LIKE NEVER BEFORE

PROJECT REPORT

Machine Learning using Python

PREPARED BY

Abhishek Saha

CSE 3rd Year

24100120025

Table of Contents

Abstract

The Crop Prediction System is an essential tool for farmers to make informed decisions about crop selection and yield prediction. In this project, we develop a Crop Prediction System using Python and Machine Learning techniques. The system incorporates key libraries such as *NumPy, Matplotlib, Pandas, Seaborn,* and *Scikit-learn* for data manipulation, visualization, and modeling. The user interface of the system is interactive using the *"Interact"* library, which allows for the selection of various input parameters to obtain crop yield predictions. Additionally, the *"warnings"* library is used to handle error messages in the system. To improve the accuracy of the model, we apply data normalization techniques using the "Normalizer" function. Finally, the "i-loc" feature is implemented to ensure the security of the system. The results of this project demonstrate the potential of using Python and Machine Learning for developing accurate and user-friendly Crop Prediction Systems.

Keywords: *Crop Prediction System, Python, Machine Learning, NumPy, Matplotlib, Pandas, Seaborn, Scikit-learn, Interact, warnings, Normalizer, i-loc, farmers, crop selection, yield prediction, data manipulation, data visualization, modeling, accuracy, user-friendly, input parameters, security, error handling.*

Acknowledgment

I am deeply grateful to **Mr. Partha Koley**, our mentor and advisor during this training, Machine Learning using Python, for their invaluable advice and guidance. Their industry experience and expertise helped me to better understand and groom myself to the company and the industry needs.

Throughout the training, he has provided me with valuable insights and guidance that helped me to navigate my tasks and responsibilities. He was always available to answer my questions and provide support, and their wisdom and expertise helped me to grow as a professional. I am thankful for their time and support, and for sharing their valuable insights with me.

I would also like to express my note of gratitude to the principal of our college Swami Vivekananda Institute of Science & Technology, **Dr. Sonali Sarkar** and **Euphoria Genx** for arranging such a valuable workshop that might provide us with better opportunities and advancements in our career ahead.

Signature

Software Development Tools (SDT)

Python

Python is a high-level programming language that is widely used in various fields, including data science and machine learning. It offers a vast array of libraries and tools that facilitate data manipulation, analysis, and visualization.

NumPy

NumPy is a fundamental library in Python that offers powerful tools for performing mathematical operations and working with large, multi-dimensional arrays and matrices. It is commonly used in data processing and scientific computing.

Pandas

Pandas is a Python library used for data manipulation and analysis. It provides an efficient way to handle data in the form of tables or data frames and enables users to perform complex operations on data, such as filtering, merging, and aggregation.

Matplotlib

Matplotlib is a Python library that provides a range of data visualization tools, including line plots, scatter plots, bar charts, and histograms. It is a valuable tool for exploring data and presenting results.

Seaborn

Seaborn is a Python library built on top of Matplotlib that offers a more advanced and user-friendly interface for data visualization. It provides a range of tools for visualizing statistical data, including heatmaps, pair plots, and categorical plots.

Warnings

Warnings is a Python library that provides a way to handle and display warning messages. It is useful for debugging code and alerting users of potential issues.

Scikit-learn

Scikit-learn is a Python library for machine learning that provides a range of tools for data mining, analysis, and modeling. It offers a range of algorithms for classification, regression, clustering, and dimensionality reduction.

Interact

Interact is a Python library that allows for the creation of interactive user interfaces. It enables users to manipulate input parameters and immediately visualize the output of their operations.

Normalizer

Normalizer is a tool provided by Scikit-learn that is used to normalize data in a dataset. It transforms the data to have a mean of zero and a variance of one, which can improve the performance of machine learning models.

K-means

Kmeans is a clustering algorithm provided by Scikit-learn that is used to group data into clusters based on their similarity. It is commonly used in unsupervised learning applications.

Random_state

Random_state is a parameter provided by Scikit-learn that enables users to specify a random seed for reproducible results. It is commonly used in machine learning algorithms to ensure that the results are consistent across different runs.

Train_test_split

Train_test_split is a function provided by Scikit-learn that is used to split a dataset into training and testing subsets. It is a valuable tool for evaluating the performance of machine learning models.

LogisticRegression

LogisticRegression is a classification algorithm provided by Scikit-learn that is used for binary classification tasks. It is commonly used in machine learning applications, such as predicting whether an email is spam or not.

Classification_report

Classification_report is a tool provided by Scikit-learn that is used to generate a report on the performance of a classification model. It provides metrics such as precision, recall, and F1-score.

Confusion_matrix

Confusion_matrix is a tool provided by Scikit-learn that is used to evaluate the performance of a classification model. It displays the number of true positives, false positives, true negatives, and false negatives for each class in the dataset.

Model_selection

Model_selection is a tool provided by Scikit-learn that is used to optimize machine learning models by selecting the best set of parameters. It enables users to perform grid search, cross-validation, and other techniques to fine-tune their models.

Software Development Kit (SDK)

Google Colab

Google Colab is a cloud-based platform for data science and machine learning that enables users to write and run Python code in a Jupyter Notebook-like environment. It provides a free, convenient, and user-friendly way to access powerful computing resources and collaborate with others on projects.

Google has been aggressive in the field of AI research. Being a company with enormous resources, it can continually experiment and make breakthroughs in the field of Quantum AI. So, it also has a vested interest in the future of these technologies. Google's AI framework, called TensorFlow, was made open source in 2015. This was followed by making Google's development tool, Colaboratory, free for public use in 2017.

In this Project, Crop Prediction System using Python with Machine Learning, Google Colab was used to build and deploy the system in the following ways:

Accessing Libraries

Google Colab comes with pre-installed libraries such as NumPy, Matplotlib, and Pandas, which are essential for data manipulation and visualization. Additionally, Colab allows users to install and use additional libraries, such as Scikit-learn, Seaborn, and Interact, to perform machine learning tasks and build interactive user interfaces.

Computational Power

Google Colab provides powerful computing resources, including GPU and TPU acceleration, that enable users to perform complex calculations and train machine learning models quickly. This feature is especially useful for the Crop Prediction System, which involves processing and analyzing large datasets.

Software Development Life Cycle

Spiral Model

The spiral model of the software process has been evolving for several years, based on experience with various refinements of the waterfall model as applied to large government software projects. As will be discussed, the spiral model can accommodate most previous models as special cases and further provides guidance as to which combination of previous model's best fits a given software situation. Development of the TRW Software Productivity System (TRW-SPS), described in the next section, is its most complete application to date.

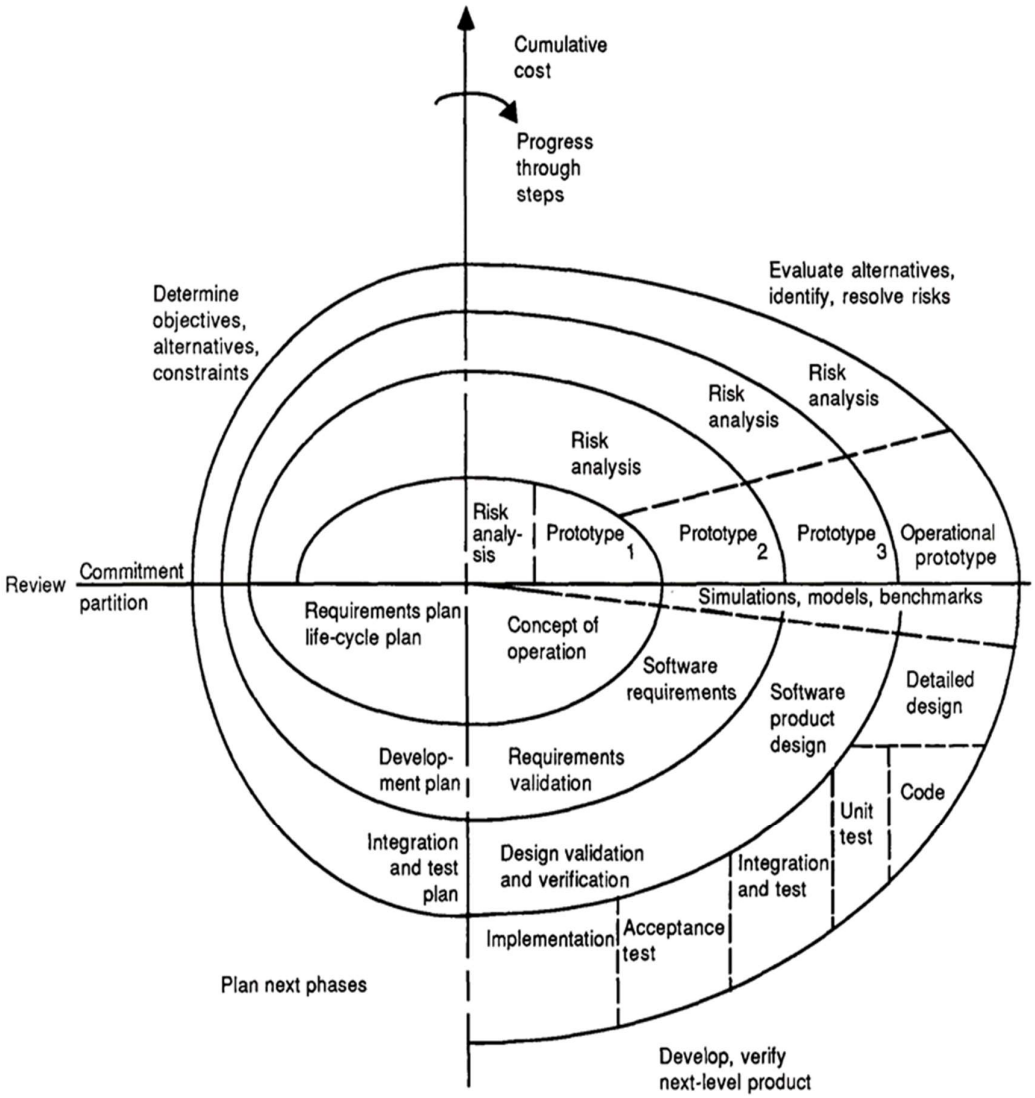


Fig 1: *Spiral model of the software process.*

Machine Learning Life Cycle

The machine learning life cycle is the cyclical process that data science projects follow. It defines each step that an organization should follow to take advantage of machine learning and artificial intelligence (AI) to derive practical business value.

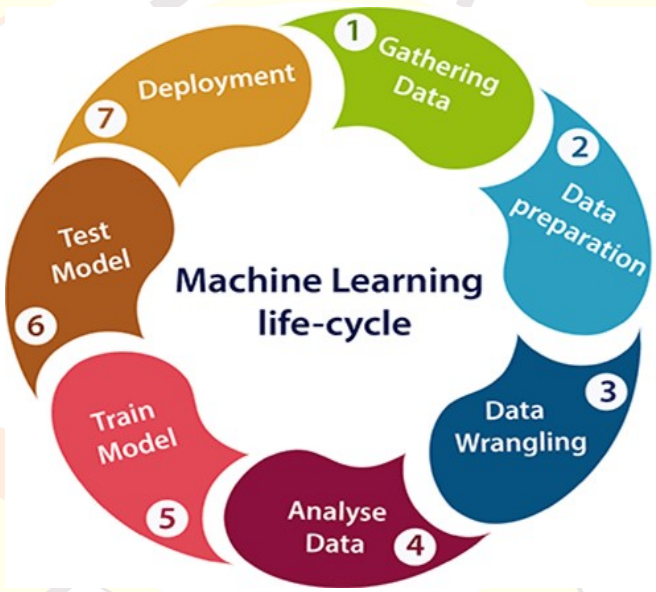


Fig 2: Machine Learning Life Cycle.

Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.

Supervised Learning

Supervised learning is the type of machine learning in which machines are trained using well "labelled" training data, and based on that data, machines predict the output. The labelled data means some input data is already tagged with the correct output. In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly.

Unsupervised Learning

Unsupervised learning is a type of machine learning in which models are trained using an unlabeled dataset and can act on that data without any supervision. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things.

Workflow of Project

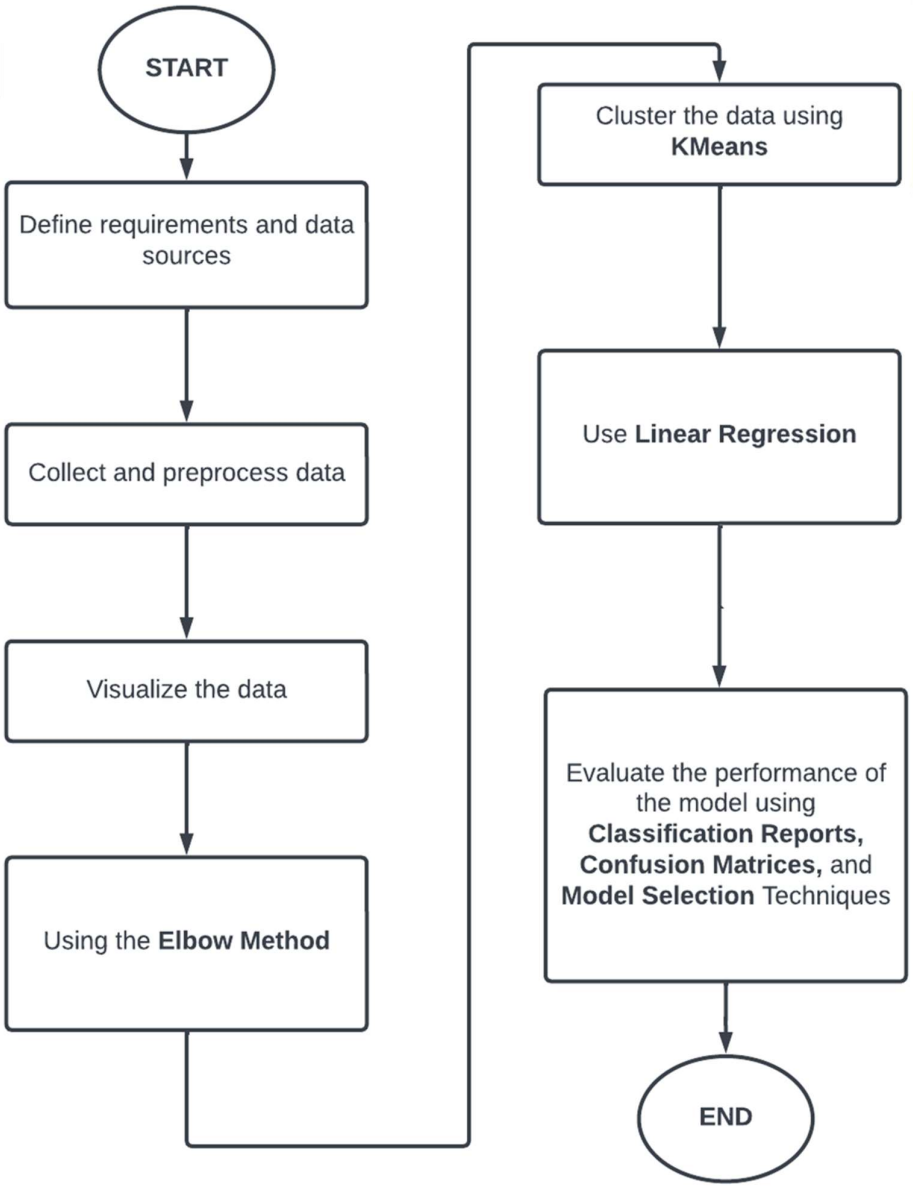


Fig 3: *Flowchart of the Project.*

Logistic Regression

Logistic regression is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict the probability of an instance belonging to a given class. It is used for classification algorithms; its name is logistic regression. It is referred to as regression because it takes the output of the linear regression function as input and uses a sigmoid function to estimate the probability for the given class. The difference between linear regression and logistic regression is that linear regression output is the continuous value that can be anything while logistic regression predicts the probability that an instance belongs to a given class or not.

Elbow Method

The Elbow method is a heuristic used in determining the number of clusters in a data set. The method consists of plotting the explained variation as a function of the number of clusters and picking the elbow of the curve as the number of clusters to use. The same method can be used to choose the number of parameters in other data-driven models, such as the number of principal components to describe a data set.

K-Means

k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. k-means clustering minimizes within-cluster variances (squared Euclidean distances), but not regular Euclidean distances, which would be the more difficult Weber problem: the mean optimizes squared errors, whereas only the geometric median minimizes Euclidean distances. For instance, better Euclidean solutions can be found using k-medians and k-medoids.

Classification Report

The Classification Report is a summary of the key performance metrics of the machine learning model that predicts crop yields based on the clusters. It provides a detailed breakdown of the model's performance, including its precision, recall, F1-score, and support.

Precision measures the proportion of true positives to false positives, indicating the proportion of correctly identified high yield crops out of all crops identified as high yield. Recall measures the proportion of true positives to false negatives, indicating the proportion of correctly identified high yield crops out of all actual high yield crops. The F1-score is the harmonic mean of precision and recall, providing a balanced view of the model's performance. Finally, support is the number of samples for each class.

Confusion Matrix

In this project, the Confusion Matrix is used to evaluate the performance of the machine learning model that predicts crop yields based on the clusters. A Confusion Matrix is a table that shows the number of correct and incorrect predictions made by the model. It is especially useful when dealing with classification problems, such as predicting crop yields. The Confusion Matrix is constructed by comparing the predicted values with the actual values. The table contains four metrics: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

- **True Positive (TP)** represents the number of crops that were correctly predicted to have a high yield.
- **True Negative (TN)** represents the number of crops that were correctly predicted to have a low yield.
- **False Positive (FP)** represents the number of crops that were predicted to have a high yield but actually had a low yield.
- **False Negative (FN)** represents the number of crops that were predicted to have a low yield but actually had a high yield.

By analyzing the values in the Confusion Matrix, it is possible to calculate different performance metrics for the machine learning model, such as accuracy, precision, recall, and F1-score. These metrics help to determine how well the model is performing and can be used to optimize it further.

Source code with Output

Importing Important Packages

Data Manipulation

```
import numpy as np

import pandas as pd
```

Data Visualization

```
import matplotlib.pyplot as plt

import seaborn as sns
```

Interactive Analysis

```
from ipywidgets import interact

import warnings

warnings.filterwarnings("ignore")
```

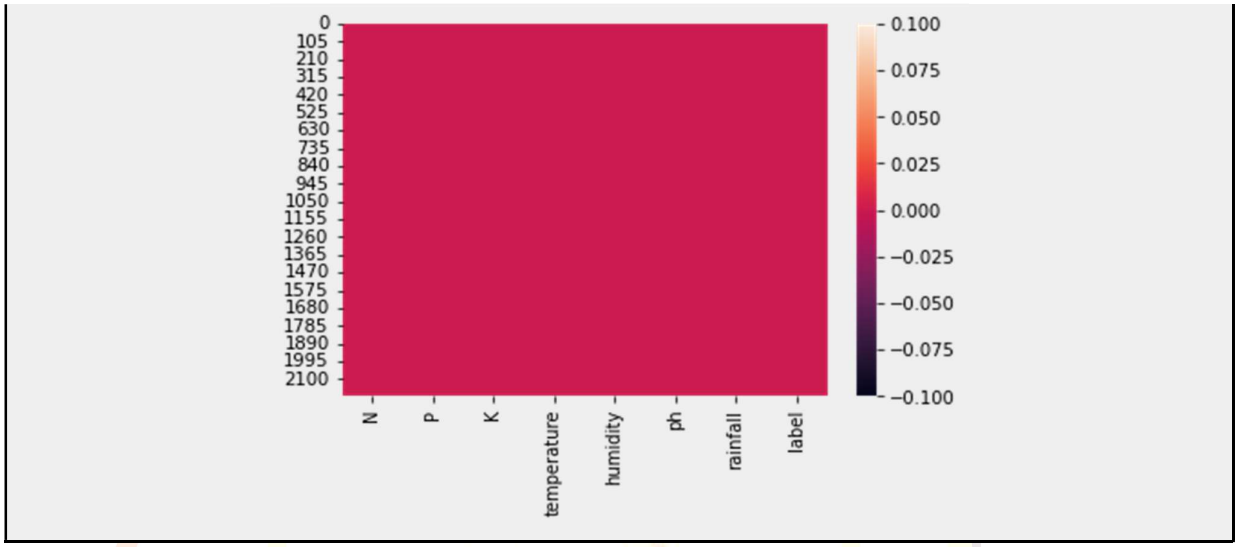
Load Dataset

```
data=pd.read_csv("/content/data.csv")
```

Missing Data Checking

```
sns.heatmap(data.isnull())
```

Output



Seasonal Crops

Summer's Crop

```
print(data[(data['temperature']>30) & (data['humidity']>50)]['label']. unique())
```

```
Output
['pigeonpeas' 'mothbeans' 'blackgram' 'mango' 'grapes' 'orange' 'papaya']
```

Monsoon's Crops

```
print(data[(data[ 'rainfall' ]>200) & (data['humidity']>50)]['label']. unique())
```

```
Output
['rice' 'papaya' 'coconut']
```

Winter Crops

```
print(data[(data['temperature']<20) & (data['humidity']<50)]['label'].unique())
```

```
Output
['chickpea' 'kidneybeans' 'pigeonpeas']
```

Crop Wise Requirement

```
@interact

def summary(crops=list(data['label'].value_counts().index)):

    x = data[data['label']==crops]

    print("Minimum Nitrogen Required",x['N'].min())
```

```
print("Average Nitrogen Required",x['N'].mean())
print("Maximum Nitrogen Required",x['N'].max())
print("----------------------------------------")

print("Minimum Phosphorus Required",x['P'].min())
print("Average Phosphorus Required",x['P'].mean())
print("Maximum Phosphorus Required",x['P'].max())
print("----------------------------------------")

print("Minimum Potassium Required",x['K'].min())
print("Average Potassium Required",x['K'].mean())
print("Maximum Potassium Required",x['K'].max())
print("----------------------------------------")

print("Minimum Temperature Required",x['temperature'].min())
print("Average Temperature Required",x['temperature'].mean())
print("Maximum Temperature Required",x['temperature'].max())
print("----------------------------------------")

print("Min humidity required ",x['humidity'].min())
print("Avg humidity required ",x[ 'humidity'].mean())
print("Max humidity required ",x['humidity'].max())
print("----------------------------------------")

print("Min ph required ",x['ph'].min())
```

```
print("Avg ph required ",x['ph'].mean())

print("Max ph required ",x['ph'].max())

print("----------------------------------------")

print("Min rainfall required ",x['rainfall'].min())

print("Avg rainfall required ",x['rainfall'].mean())

print("Max rainfall required ",x['rainfall'].max())

print("----------------------------------------")
```

Output

```
Minimum Nitrogen Required 60
Average Nitrogen Required 79.89
Maximum Nitrogen Required 99
----------------------------------------
Minimum Phosphorus Required 35
Average Phosphorus Required 47.58
Maximum Phosphorus Required 60
----------------------------------------
Minimum Potassium Required 35
Average Potassium Required 39.87
Maximum Potassium Required 45
----------------------------------------
Minimum Temperature Required 20.0454142
Average Temperature Required 23.6893322105
Maximum Temperature Required 26.92995077
----------------------------------------
Min humidity required  80.12267476
Avg humidity required  82.27282153889999
Max humidity required  84.96907151
----------------------------------------
Min ph required  5.005306977
Avg ph required  6.425470922139999
Max ph required  7.868474653
----------------------------------------
Min rainfall required  182.5616319
```

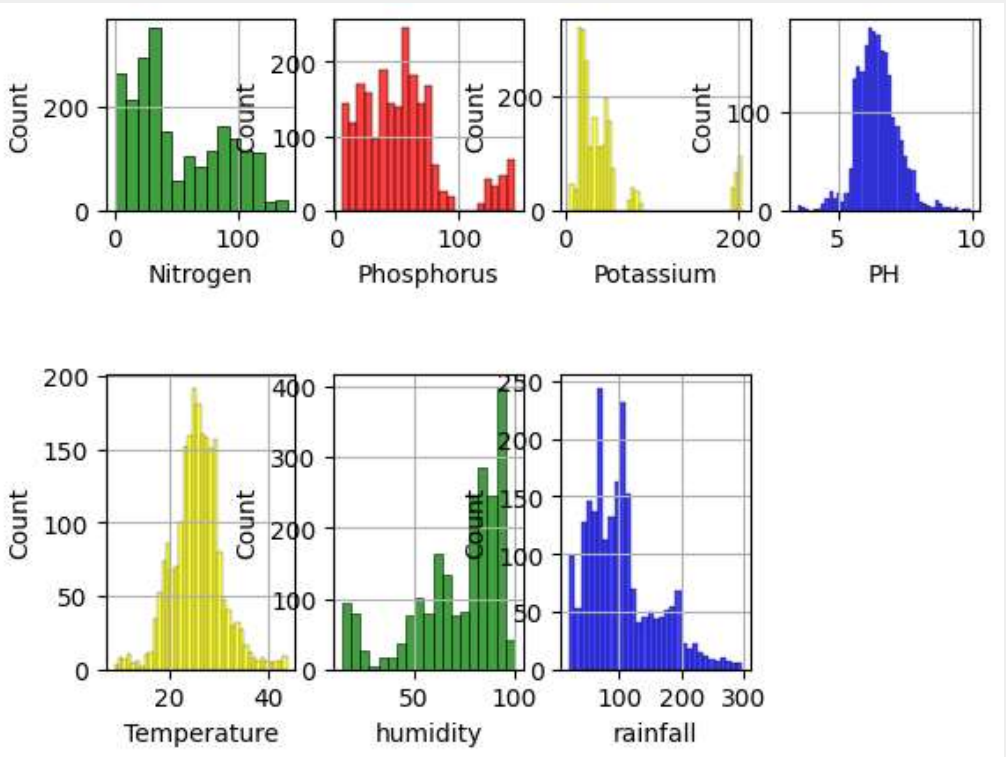
```
Avg rainfall required  236.18111359399998
Max rainfall required  298.5601175
----------------------------------------
```

Graph for Multivariance

```
plt.subplot(3,4,1)
sns.histplot(data['N'], color="green")
plt.xlabel("Nitrogen")
plt.grid()

plt.subplot(3,4,2)
sns.histplot(data['P'], color="red")
plt.xlabel("Phosphorus")
plt.grid()

plt.subplot(3,4,3)
sns.histplot(data[ 'K'], color="yellow")
plt.xlabel("Potassium")
plt.grid()

plt.subplot(3,4,4)
sns.histplot(data['ph'], color="blue")
plt.xlabel("PH")
plt.grid()

plt.subplot(2,4,5)
sns.histplot(data['temperature'], color="yellow")
plt.xlabel("Temperature")
plt.grid()
```

```
plt.subplot(2,4,6)
sns.histplot(data['humidity'], color="green")
plt.xlabel("humidity")
plt.grid()

plt.subplot(2,4,7)
sns.histplot(data['rainfall'], color="blue")
plt.xlabel("rainfall")
plt.grid()
```

Output

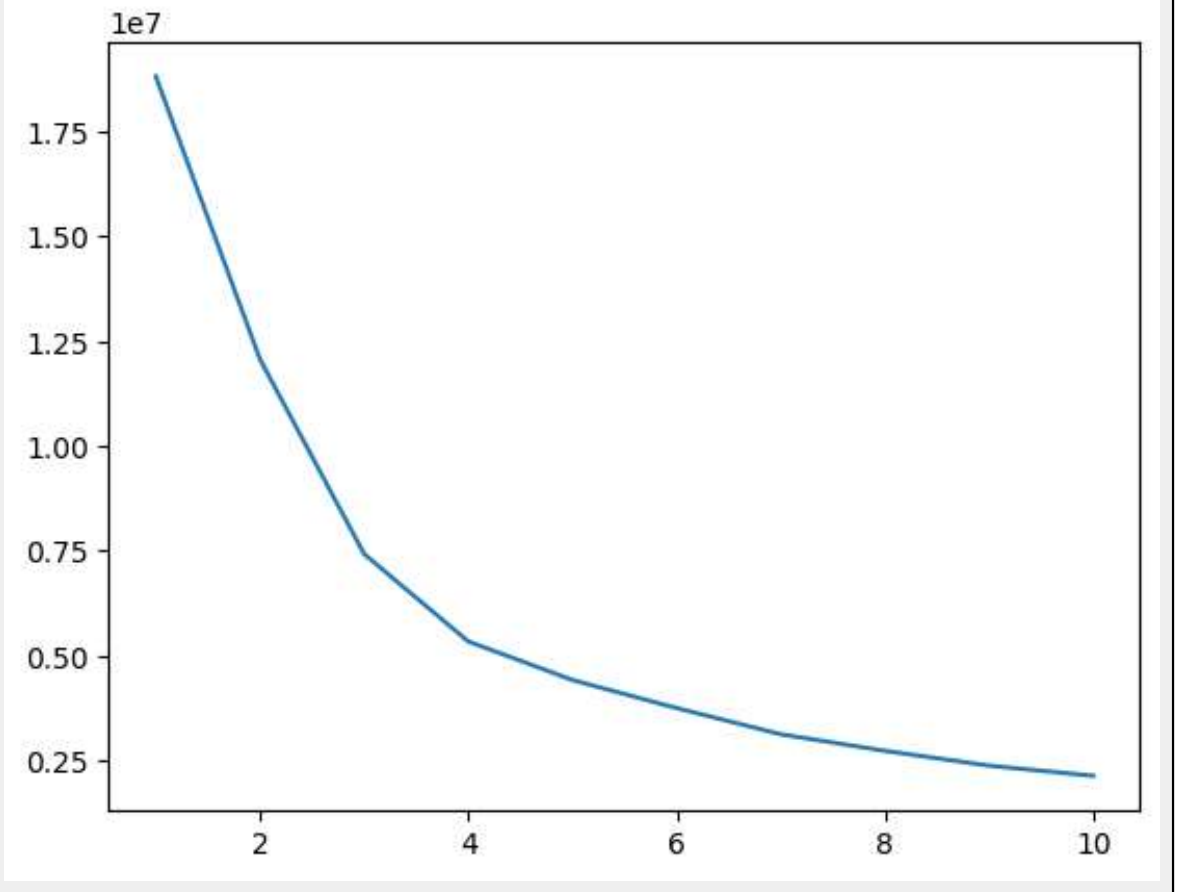


Elbow Method

```
from pandas.core.common import random_state
from sklearn.cluster import KMeans
```

```
x=data.drop(['label'], axis=1)
x=x.values
wcss=[]
for i in range(1,11):
  km=KMeans(n_clusters=i, init="k-means++", max_iter=2000, n_init=10, random_state=0)
  km.fit(x)
  wcss.append(km.inertia_)
plt.plot(range(1,11), wcss)
plt.show()
```

Output



Classifying with Kmeans Clustering

```
km=KMeans(n_clusters=4, init="k-means++", max_iter=2000, n_init=10, random_state=0)

y_means=km.fit_predict(x)

a=data['label']

y_means=pd.DataFrame(y_means)

z=pd.concat([y_means, a], axis=1)

z=z.rename(columns={0: 'cluster'})

print("Cluster 1",z[z['cluster']==0]['label'].unique())

print("Cluster 2",z[z['cluster']==1]['label'].unique())

print("Cluster 3",z[z['cluster']==2]['label'].unique())

print("Cluster 4",z[z['cluster']==3]['label'].unique())
```

Output

```
Cluster 1 ['maize' 'chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans' 'mungbean'
 'blackgram' 'lentil' 'pomegranate' 'mango' 'orange' 'papaya' 'coconut']
Cluster 2 ['maize' 'banana' 'watermelon' 'muskmelon' 'papaya' 'cotton' 'coffee']
Cluster 3 ['grapes' 'apple']
Cluster 4 ['rice' 'pigeonpeas' 'papaya' 'coconut' 'jute' 'coffee']
```

Model Selection

```
y=data['label']

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = .2, random_state=0)
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()

model.fit(x_train,y_train)

y_pred = model.predict(np.array([[20,30,40,20,80,7,100]]))
```

```
print(y_pred)
```

Output

```
['pomegranate']
```

Classification Report

```
y_pred = model.predict(x_test)

from sklearn.metrics import classification_report

cr=classification_report(y_test, y_pred)

print(cr)
```

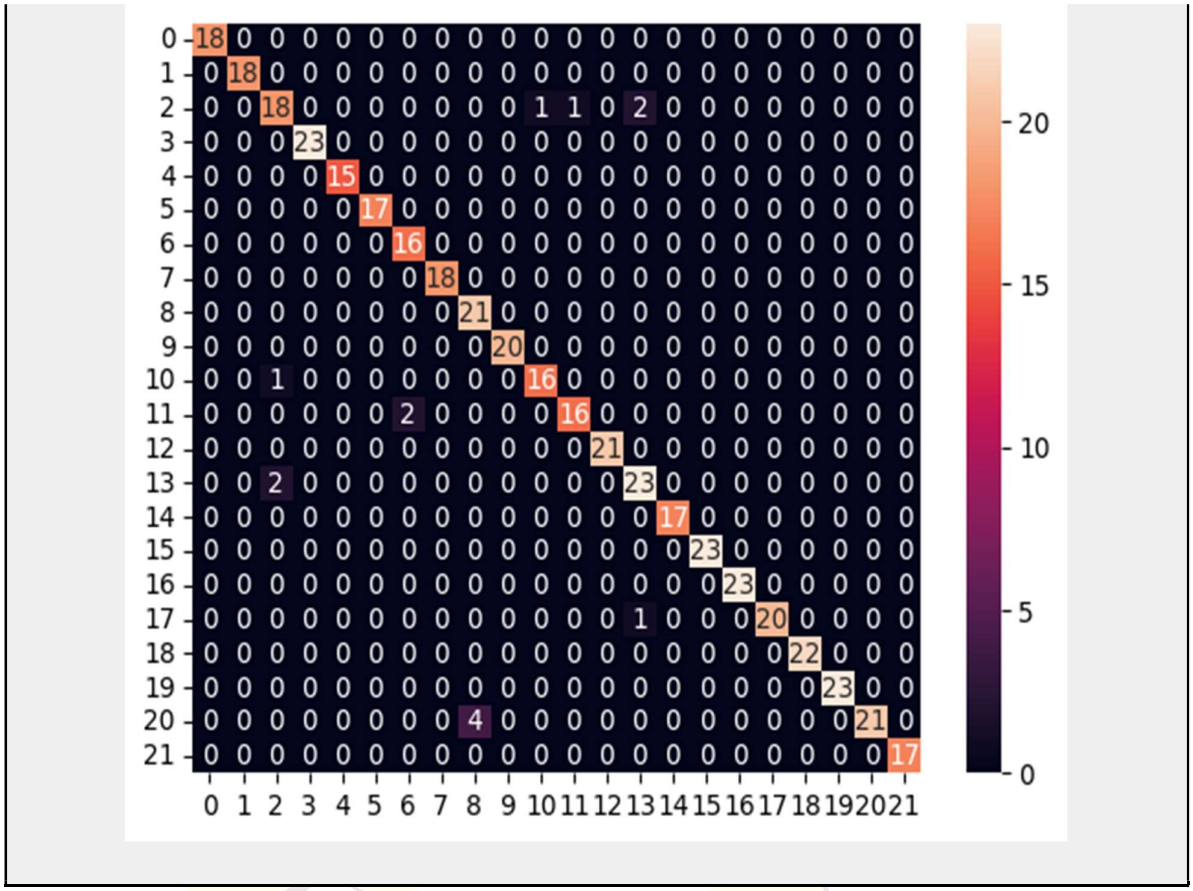
Output

	precision	recall	f1-score	support
apple	1.00	1.00	1.00	18
banana	1.00	1.00	1.00	18
blackgram	0.86	0.82	0.84	22
chickpea	1.00	1.00	1.00	23
coconut	1.00	1.00	1.00	15
coffee	1.00	1.00	1.00	17
cotton	0.89	1.00	0.94	16
grapes	1.00	1.00	1.00	18
jute	0.84	1.00	0.91	21
kidneybeans	1.00	1.00	1.00	20
lentil	0.94	0.94	0.94	17
maize	0.94	0.89	0.91	18
mango	1.00	1.00	1.00	21
mothbeans	0.88	0.92	0.90	25
mungbean	1.00	1.00	1.00	17
muskmelon	1.00	1.00	1.00	23
orange	1.00	1.00	1.00	23
papaya	1.00	0.95	0.98	21
pigeonpeas	1.00	1.00	1.00	22
pomegranate	1.00	1.00	1.00	23
rice	1.00	0.84	0.91	25
watermelon	1.00	1.00	1.00	17
accuracy			0.97	440
macro avg	0.97	0.97	0.97	440
weighted avg	0.97	0.97	0.97	440

Confusion Matrix

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test,y_pred)

sns.heatmap(cm,annot=True)
```

Output



Conclusion

In conclusion, building a Crop Prediction System using Python with Machine Learning can be a powerful tool for farmers to make informed decisions about crop selection and yield prediction. Utilizing libraries such as NumPy, Matplotlib, and Pandas, as well as machine learning tools like Scikit-learn and Logistic Regression, can help analyze and visualize data effectively. Platforms like Google Colab provide a convenient and user-friendly environment to build and deploy the system, with powerful computing resources, collaborative features, and easy deployment options. This can ultimately lead to better crop yields and economic benefits for farmers.

Future Scope

Our model can be trained further with more data points of different states. This system can be extended to different climatic conditions also. The proposed model can

be used not only for our state but also for other states if we provide more accurate data to it using satellite and sensor data.

Bibliography

1. Iniyan, S., Varma, V. A., & Naidu, C. T. (2023). Crop yield prediction using machine learning techniques. *Advances in Engineering Software*, *175*, 103326.
2. M. Gupta, B. V. Santhosh Krishna, B. Kavyashree, H. R. Narapureddy, N. Surapaneni and K. Varma, "Crop Yield Prediction Techniques Using Machine Learning Algorithms," 2022 8th International Conference on Smart Structures and Systems (ICSSS), Chennai, India, 2022, pp. 1-7.
3. Y. J. N. Kumar, V. Spandana, V. S. Vaishnavi, K. Neha and V. G. R. R. Devi, "Supervised Machine learning Approach for Crop Yield Prediction in Agriculture Sector," 2020 5th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2020, pp. 736-741.
4. Chlingaryan, A., Sukkarieh, S., & Whelan, B. (2018). Machine learning approaches for crop yield prediction and nitrogen status estimation in precision agriculture: A review. Computers and Electronics in Agriculture, 151, 61–69.
5. Journal, I. (2020, January 1). IJERT-Prediction and Analysis of Crop Yield using Machine Learning Techniques.
6. Presidency University India (Director). (2021). Report (Agriculture production optimization engine using logistic regression). studocu.com. Retrieved April 4, 2023.