A

Major Project Report on

**ELEVATING AGRICULTURE: ADVANCED HOPS CLASSIFICATION FOR PRECISION CROP MANAGEMENT AND QUALITY HARVESTS**

*Submitted for partial fulfilment of the requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**INFORMATION TECHNOLOGY**

*Submitted by*

|  |  |
| --- | --- |
| **SYED AMMAAR FARHAN** | **20K81A12B7** |
| **MOHAMMED FARHAAN** | **20K81A1299** |
| **BARDAVAL DINESH** | **20K81A1269** |
| **NARRA LOKESH** | **20K81A12A3** |

Under the Guidance of

**Mr. K VENKAT TIRU GOPAL REDDY**

**ASSISTANT PROFESSOR**

****

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**St. MARTIN'S ENGINEERING COLLEGE**

UGC Autonomous

Affiliated to JNTUH, Approved by AICTE,

Accredited by NBA & NAAC A+, ISO 9001-2008 Certified

Dhulapally, Secunderabad - 500 100

[www.smec.ac.in](http://www.smec.ac.in)

**MARCH- 2024**

|  |
| --- |
| **St. MARTIN'S ENGINEERING COLLEGE** |
| UGC Autonomous  Affiliated to JNTUH, Approved by AICTE  NBA & NAAC A+ Accredited Dhulapally, Secunderabad-500100  [ww.smec.ac.in](http://www.smec.ac.in/) |

**Certificate**

This is to certify that the project entitled **“ELEVATING AGRICULTURE: ADVANCED HOPS CLASSIFICATION FOR PRECISION CROP MANAGEMENT AND QUALITY HARVEST”** is being submitted by Syed Ammaar Farhan(20K81A12B7), Mohammed Farhaan(20K81A1299), Dinesh Bardaval(20K81A1269), Narra Lokesh(20K81A12A3) in fulfilment of the requirement for the award of degree of **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** is recorded of bonafide work carried out by them. The result embodied in this report have been verified and found satisfactory.

Signature of Guide Signature of HOD

**Mr. K V T GOPAL REDDY Dr. V K SENTHIL RAGAVAN**

Assistant Professor Professor and Head of Department

Department of Information Technology Department of Information Technology

**Internal Examiner External Examiner**

Place:

Date:

|  |  |  |
| --- | --- | --- |
|  | **St. MARTIN'S ENGINEERING COLLEGE** |  |
| UGC Autonomous  NBA & NAAC A+ Accredited  Dhulapally, Secunderabad - 500 100  www.smec.ac.in |

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**DECLARATION**

We, the students of ‘**Bachelor of Technology in Department of Information Technology’**, session: 2020 - 2024**, St. Martin’s Engineering College, Dhulapally, Kompally, Secunderabad,** hereby declare that the work presented in this Project Work entitled “**ELEVATING AGRICULTURE: ADVANCED HOPS CLASSIFICATION FOR PRECISION CROP MANAGEMENT AND QUALITY HARVEST”** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. This result embodied in this project report has not been submitted in any university for award of any degree.

**SYED AMMAAR FARHAN 20K81A12B7**

**MOHAMMED FARHAAN 20K81A1299**

**BARDAVAL DINESH 20K81A1269**

**NARRA LOKESH** **20K81A12A3**

**ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance have crowded our efforts with success.

First and foremost, we would like to express our deep sense of gratitude and indebtedness to our College Management for their kind support and permission to use the facilities available in the Institute.

We especially would like to express our deep sense of gratitude and indebtedness to **Dr. P. SANTOSH KUMAR PATRA**, Group Director, St. Martin’s Engineering College Dhulapally, for permitting us to undertake this project.

We wish to record our profound gratitude to **Dr. M. SREENIVAS RAO**, Principal, St. Martin’s Engineering College, for his motivation and encouragement.

We are also thankful to **Dr. V K SENTHIL RAGAVAN**, Head of the Department, Information Technology, St. Martin’s Engineering College, Dhulapally, Secunderabad. for his support and guidance throughout our project as well as Project Coordinator **Mrs**. **M SANDHYA RANI**, Assistant Professor, Information Technology department for her valuable support.

We would like to express our sincere gratitude and indebtedness to our project supervisor **Mr. K VENKAT TIRU GOPAL REDDY**, Assistant Professor, Information Technology, St. Martins Engineering College, Dhulapally, for hir support and guidance throughout our project.

Finally, we express thanks to all those who have helped us successfully completing this project. Furthermore, we would like to thank our family and friends for their moral support and encouragement. We express thanks to all those who have helped us in successfully completing the project.

**SYED AMMAAR FARHAN 20K81A12B7**

**MOHAMMED FARHAAN 20K81A1299**

**BARDAVAL DINESH 20K81A1269**

**NARRA LOKESH 20K81A12A3**

i

**ABSTRACT**

Agriculture, the bedrock of human civilization, sustains life and fuels various industries. With burgeoning global populations and shifting climate patterns, there's a growing imperative for inventive methods to optimize crop production. Precision agriculture, leveraging technology, stands as a beacon, streamlining operations, curbing resource wastage, and bolstering yields. Advanced crop classification, notably in crops like hops, presents a potential paradigm shift in this domain. Traditional agriculture leaned heavily on manual labour and basic tools. Crop categorization often relied on visual assessments by seasoned farmers, considering factors like leaf shape, colour, and fruiting attributes. While effective to an extent, this method was time-intensive, prone to human error, and ill-suited for large-scale farming. Technological strides have gradually paved the way for more sophisticated methodologies. Advanced hops classification in precision agriculture is driven by several pivotal factors. The brewing industry, a major hops consumer, demands specific varieties with distinct flavour and aromatic profiles. Accurate classification ensures the cultivation of hops that meet these requirements. Precise crop management is also crucial, optimizing resources like water, nutrients, and pest control, aligning with sustainability goals. Moreover, enhanced crop classification translates to higher yields and favourable market prices, bolstering economic viability for farmers. The challenge at hand revolves around creating a sophisticated hops classification system. This system would harness cutting-edge technology, incorporating machine learning and computer vision to differentiate between various hop varieties accurately. The goal is to enable meticulous crop management practices and guarantee that harvested hops align with the exacting quality standards of the brewing industry. Hops classification signifies a monumental leap in contemporary agriculture.

ii

**LIST OF FIGURES**

iii

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **FIGURE TITLE** | **PAGE NO.** |
|  |  |  |
| 3.1 | Analysis of SVM | 07 |
| 3.2 | Basic classification of SVM | 08 |
| 3.3 | Linear SVM | 09 |
| 3.4 | Test-Vector in SVM | 09 |
| 3.5 | Classification in SVM | 10 |
| 3.6 | Non-Linear SVM | 10 |
| 3.7 | Non-Linear SVM data Separation | 11 |
| 3.8 | Non-Linear SVM best hyperplane | 11 |
| 3.9 | Non-Linear SVM with ROC | 12 |
| 4.1 | Proposed Architecture Diagram of Hops Classification Model | 14 |
| 4.2 | Kernel Discriminant Analysis | 16 |
| 4.3  4.4 | Representation of convolution layer process  Example of convolution layer process (a) an image with size 5×5 is convolving with 3×3 kernel (b) Convolved feature map. | 16  17 |
| 5.1 | Class Diagram | 20 |
| 5.2 | Sequence Diagram | 21 |
| 5.3 | Use Case Diagram | 21 |
| 10.1 | User interface application of proposed Deep learning Approach for Hops Classification. | 54 |
| 10.2 | Illustrates the image after uploading the dataset. | 54 |
| 10.3 | Illustration and Description of classes after pre-processing. | 55 |
| 10.4 | Model accuracy of Existing MLP Classifier. | 56 |
| 10.5 | confusion matrix of existing MLP Classifier | 56 |
| 10.6 | Model Accuracy of proposed Deep learning CNN Algorithm | 57 |
| 10.7 | Model summary of Proposed CNN Architecture | 58 |
| 10.8 | Prediction Results of Proposed CNN Algorithm | 59 |
| 10.9 | Accuracy and loss graph for Proposed Model | 60 |

iv

**LIST OF TABLES**

v

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TABLE NAME** | **PAGE NO.** |
|  |  |  |
| 10.1 | Dataset Description | 53 |
| 10.2 | Performance comparison of MLP classifier and CNN Model | 60 |

**LIST OF ACRONYMS WITH FULL FORMS**



|  |  |  |
| --- | --- | --- |
| **S. NO.** | **ACRONYM** | **FULL FORM** |
| 01. | FAO | Food and Agriculture Organisation |
| 02. | IoT | Internet of Things |
| 03. | WSN | Wireless Sensor Networks |
| 04. | mMTC | Massive Machine-Type Communication |
| 05. | SVM | Support Vector Machine |
| 06. | MLP | Multilayer Perceptron |
| 07. | CNN | Convolutional Neural Network |
| 08. | ReLU | Rectified Linear Unit |
| 09. | IDLE | Integrated Development and Learning Environment |
| 10. | KNN | K-Nearest Neighbours |
| 11. | PCA | Principal Components Analysis |
| 12. | LDA | Linear Discriminant Analysis |
| 13. | UML | Unified Modelling Language |
| 14. | XML | Extensible Markup Language |
| 15. | JSON | JavaScript Object Notation |
| 16. | JIT | Just-In-Time (compiler) |
| 17. | JDBC | Java Database Connectivity |
| 18. | ODBC | Open Database Connectivity |
| 19. | CSV | Comma-Separated Values |
| 20.  21. | EDA  GUI | Exploratory Data Analysis  Graphical User Interface |

vi

**CONTENTS**

**i**

**ii**

**iii**

**v**

**vi**

**1**

1

2

3

4

**6**

**7**

**7**

8

**13**

13

14

14

15

16

**19**

19

20

21

**22**

22

22

25

25

**34**

34

34

**ACKNOWLEDGEMENT**

**ABSTRACT** 

**LIST OF FIGURES**

**LIST OF TABLES v**

**LIST OF ACRONYMS WITH FULL FORMS**

**CHAPTER 1: INTRODUCTION 1**

* 1. HISTORY 1
  2. RESEARCH MOTIVATION
  3. PROBLEM STATEMENT
  4. APPLICATIONS

**CHAPTER 2: LITERATURE SURVEY 5**

**CHAPTER 3: EXISTING SYSTEM 8**

3.1 SUPPORT VECTOR MACHINE ALGORITHM

3.2 SVM WORKING

**CHAPTER 4: PROPOSED SYSTEM 11**

4.1 OVERVIEW 11

4.2 DATA PREPROCESSING

4.2.1 SPLITTING THE DATASET INTO THE TRAINING SET AND

TEST SET

4.3 KLDA

4.4 CNN

**CHAPTER 5: UML DIAGRAMS**

5.1 CLASS DIAGRAM

5.2 SEQUENCE DIAGRAM

5.3 USE CASE DIAGRAM

**CHAPTER 6: SOFTWARE ENVIRONMENT**

6.1 WHAT IS PYTHON

6.2 ADVANTAGES OF PYTHON

6.3 DISADVANTAGES OF PYTHON

6.4 MODULES USED IN PROJECT

**CHAPTER 7: SYSTEM REQUIREMENT**

7.1 SOFTWARE REQUIREMENTS

vii

7.2 HARDWARE REQUIREMENTS

**CHAPTER 8: FUNCTIONAL REQUIREMENT**

**35**

35

35

35

37

37

38

38

**52**

52

53

53

**61**

61

61

62

8.1 OUTPUT DESIGN

8.2 OUTPUT DEFINITION

8.3 KEY ASPECTS OF INPUT DESIGN

8.4 ERROR MANAGEMENT AND DATA VALIDATION

8.5 USER INTERFACE DESIGN

8.6 ERROR MESSAGE DESIGN

8.7 PERFORMANCE REQUIREMENTS

**CHAPTER 10: RESULTS AND DISCUSSION**

10.1 IMPLEMENTATION DISCRIPTION

10.2 DATASET DISCRIPTION

10.3 RESULT AND DISCRIPTION

**CHAPTER 11: CONCLUSION AND FUTURE SCOPE**

11.1 CONCLUSION

11.2 FUTURE SCOPE

**REFERENCES**

viii

**CHAPTER 1**

**INTRODUCTION**

* 1. **HISTORY**

In the realm of precision crop management and the pursuit of quality harvests, the evolution of hops classification has played a pivotal role in enhancing agricultural practices and ensuring optimal yields. The history of advanced hops classification is a narrative of continuous innovation, driven by the need for precision in cultivation and the desire to achieve superior crop quality.

The origins of hops cultivation can be traced back to ancient times, with the earliest documented use dating to the Roman Empire. However, it was not until the Middle Ages that hops gained prominence in brewing, particularly in the production of beer. Over the centuries, as the demand for hops grew, so did the necessity for a more refined classification system to cater to the diverse requirements of brewers and farmers alike.

The initial classifications were rudimentary, often based on observable traits such as plant height, cone size, and colour. As the scientific understanding of plant genetics advanced, so did the sophistication of hops classification. The integration of molecular biology and genetic research allowed for a more nuanced categorization, taking into account the genetic makeup of different hop varieties.

In recent decades, the advent of advanced technologies, including high-throughput sequencing and genetic mapping, has revolutionized the classification of hops. This has led to the identification of specific genes responsible for desirable traits such as aroma, bitterness, and disease resistance. As a result, breeders have been able to develop new hop varieties with tailored characteristics to meet the demands of both brewers and agricultural conditions.

Precision crop management, a contemporary approach to farming, has seamlessly integrated with advanced hops classification. This synergy allows farmers to optimize resources such as water, fertilizers, and pesticides based on the specific needs of each hop variety. Data-driven decision-making, enabled by sensors, drones, and other precision agriculture technologies, has further elevated the efficiency and sustainability of hop cultivation.

The implications of advanced hops classification extend beyond the farm gate. Breweries now have access to a diverse array of hops, each with its own unique flavour and aromatic profile. This has sparked a renaissance in craft brewing, as brewers experiment with different hop varieties to create distinctive and high-quality beers.

* 1. **RESEARCH MOTIVATION**

The impetus behind delving into advanced hops classification for precision crop management and quality harvests stems from a confluence of agricultural challenges and the ever-growing demand for excellence in the brewing industry. This research is propelled by a profound need to optimize agricultural practices, enhance crop quality, and meet the discerning requirements of a burgeoning market.

Historically, the cultivation of hops has been deeply intertwined with brewing traditions, dating back to ancient civilizations and gaining prominence in the Middle Ages. However, as agriculture progressed and brewing evolved into a sophisticated industry, the classification of hops remained a crucial yet evolving aspect. Recognizing the intrinsic link between the genetic makeup of hop varieties and the qualities they impart to brewed products, there has been a persistent drive to refine and advance the classification methods.

The motivation for this research is underscored by the limitations of earlier, more rudimentary classification systems, which often relied on observable traits such as physical characteristics. The insufficiency of these methods became apparent as the demand for diverse hop profiles increased in the brewing sector. To address this gap, scientists and agricultural experts turned to advanced technologies and genetic research to unravel the complex genetic code governing the traits of interest.

In the contemporary context, the motivation for advanced hops classification is intrinsically tied to the principles of precision crop management. The agricultural landscape is evolving with a heightened emphasis on sustainability, resource efficiency, and environmental stewardship. Precision agriculture, enabled by cutting-edge technologies like sensors, drones, and genetic mapping, provides a platform for optimizing resource allocation, minimizing waste, and enhancing overall crop health.

Moreover, the link between precision crop management and the quality of harvested hops is of paramount importance. By tailoring cultivation practices based on the specific needs of each hop variety, farmers can maximize yields while minimizing inputs. This not only contributes to economic viability but also aligns with the broader global movement towards sustainable and eco-friendly farming practices.

* 1. **PROBLEM STATEMENT**

The contemporary landscape of hops cultivation and brewing faces multifaceted challenges that necessitate a comprehensive exploration of advanced hops classification. The existing methodologies for categorizing hop varieties, rooted in historical practices and rudimentary observations, are proving inadequate in meeting the evolving demands of both agriculture and the brewing industry.

One of the central predicaments is the discrepancy between the increasing intricacy of brewing requirements and the outdated classification systems for hops. As brewers strive for uniqueness and precision in crafting diverse beer profiles, the limitations of traditional classification, based on physical attributes alone, hinder their ability to select hops tailored to specific flavour and aromatic nuances. This misalignment poses a substantial impediment to the brewing industry's aspirations for innovation and product differentiation.

Moreover, the broader agricultural context is characterized by a pressing need for sustainability and resource optimization. Conventional farming practices, uninformed by the nuanced requirements of different hop varieties, result in suboptimal resource utilization. The lack of a sophisticated classification system hinders the implementation of precision crop management techniques, preventing farmers from maximizing yields while minimizing inputs such as water, fertilizers, and pesticides.

The gap between the aspirations of precision agriculture and the current state of hops classification exacerbates the environmental impact of cultivation practices. Inefficiencies in resource allocation contribute to environmental degradation, challenging the imperative for sustainable and eco-friendly agricultural practices. This misalignment between agricultural practices and environmental stewardship underscores the urgency of addressing the classification shortcomings in the hops cultivation domain.

Furthermore, the economic dynamics of the brewing industry amplify the significance of advanced hops classification. Craft breweries, in particular, seek to differentiate themselves in a crowded market by offering distinctive and high-quality products. The absence of a robust classification system impedes brewers' ability to make informed decisions about hop selection, limiting their capacity to create unique flavour profiles that resonate with the discerning consumer base.

* 1. **APPLICATIONS**

Precision Crop Management Optimization: Resource Allocation: Advanced hops classification facilitates precision crop management by providing a nuanced understanding of the specific needs of each hop variety. Farmers can optimize resource allocation, including water, fertilizers, and pesticides, leading to improved resource efficiency and reduced environmental impact.

Disease Resistance: The identification of genetic markers for disease resistance in specific hop varieties allows for targeted breeding efforts. This, in turn, enhances the resilience of crops against prevalent diseases, reducing the reliance on chemical interventions and promoting sustainable farming practices.

Brewing Industry Advancements: Flavour and Aroma Tailoring: Brewers can leverage the detailed classification of hop varieties to precisely select those with desired flavour and aromatic profiles. This empowers brewers to create unique and distinctive beer products, meeting the evolving preferences of consumers in the craft brewing market.

Innovation and Product Differentiation: The availability of a diverse array of precisely classified hops opens avenues for innovation in brewing recipes. Craft breweries, in particular, can differentiate themselves by experimenting with novel combinations of hop varieties, fostering creativity and a competitive edge in the marketplace.

Environmental Sustainability: Reduced Environmental Footprint: The implementation of precision crop management, guided by advanced hops classification, contributes to sustainable farming practices. By minimizing the use of inputs and optimizing cultivation practices, the environmental footprint of hops cultivation is diminished, aligning with global sustainability goals.

Biodiversity Conservation: Genetic research associated with advanced hops classification may also contribute to the preservation of genetic diversity within hop varieties. This not only ensures the resilience of crops against changing environmental conditions but also aids in the conservation of biodiversity within agricultural ecosystems.

Economic Viability: Yield Optimization: Farmers stand to benefit economically from optimized yields resulting from precision crop management. By tailoring cultivation practices based on the specific characteristics of each hop variety, they can maximize the quantity and quality of their harvest, enhancing overall economic viability.

Market Demand Response: The brewing industry's response to consumer preferences for unique and high-quality beer products creates economic opportunities for hop farmers. The ability to supply hops with specific flavour profiles that align with market demands positions farmers to meet the needs of a growing and discerning consumer base.

**CHAPTER 2**

**LITERATURE SURVEY**

The increase in demand for crops and food production is associated with the growth of the world population, which according to data from the Food and Agriculture Organization (FAO) of the United Nations, is currently 7.7 billion humans, projected to be 9.4 billion in 2030 and 10.1 billion in 2050, when the world population will need 70% more food, 42% more arable land and 120% more water for food-related purposes [[1](https://www.mdpi.com/2073-4395/13/1/244/html#B1-agronomy-13-00244),[2](https://www.mdpi.com/2073-4395/13/1/244/html#B2-agronomy-13-00244),[3](https://www.mdpi.com/2073-4395/13/1/244/html#B3-agronomy-13-00244),[4](https://www.mdpi.com/2073-4395/13/1/244/html#B4-agronomy-13-00244)].

Since traditional outdoor agriculture does not satisfy food production, coupled with the reduction of limited agricultural land for civil works construction, an optimal solution is protected crops called greenhouses that increase the number of harvests. Better yet, when transformed to smart greenhouses using information technology and sensors, can contribute to the increase of agricultural production [[5](https://www.mdpi.com/2073-4395/13/1/244/html#B5-agronomy-13-00244)].In relation to the technological advances of Industry 4.0, cloud computing and the IoT (Internet of Things) contribute to making traditional systems smart [[6](https://www.mdpi.com/2073-4395/13/1/244/html#B6-agronomy-13-00244),[7](https://www.mdpi.com/2073-4395/13/1/244/html#B7-agronomy-13-00244),[8](https://www.mdpi.com/2073-4395/13/1/244/html#B8-agronomy-13-00244)].

An example of this process is smart farming that improves productivity and reduces surplus elements used in crops [[9](https://www.mdpi.com/2073-4395/13/1/244/html#B9-agronomy-13-00244)]. On the other hand, within the IoT concept, the role of wireless sensor networks (WSN) is paramount [[10](https://www.mdpi.com/2073-4395/13/1/244/html#B10-agronomy-13-00244),[11](https://www.mdpi.com/2073-4395/13/1/244/html#B11-agronomy-13-00244)] because several IoT applications are based on wireless data transmission allowing sensor/actuator nodes to communicate with each other through a wireless network connection, even potentialized within the mMTC (massive machine-type communications) scenario of 5G [[12](https://www.mdpi.com/2073-4395/13/1/244/html#B12-agronomy-13-00244),[13](https://www.mdpi.com/2073-4395/13/1/244/html#B13-agronomy-13-00244),[14](https://www.mdpi.com/2073-4395/13/1/244/html#B14-agronomy-13-00244),[15](https://www.mdpi.com/2073-4395/13/1/244/html#B15-agronomy-13-00244)].

**CHAPTER 3**

**EXISTING SYSTEM**

**3.1 SUPPORT VECTOR MACHINE ALGORITHM**

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Figure 3.1 Analysis of SVM

**Example:** SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



Figure 3.2. Basic classification using SVM

**Types of SVM:** SVM can be of two types:

**Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

**Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier

**3.2 SVM WORKING**

**Linear SVM:** The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair (x1, x2) of coordinates in either green or blue. Consider the below image:



Figure 3.3. Linear SVM

So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



Figure 3.4. Test-Vector in SVM

Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a hyperplane. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as margin. And the goal of SVM is to maximize this margin. The hyperplane with maximum margin is called the optimal hyperplane.



Figure 3.5. Classification in SVM

**Non-Linear SVM:** If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



Figure 3.6. Non-Linear SVM

So, to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third-dimension z. It can be calculated as:

z=x2 +y2

By adding the third dimension, the sample space will become as below image:

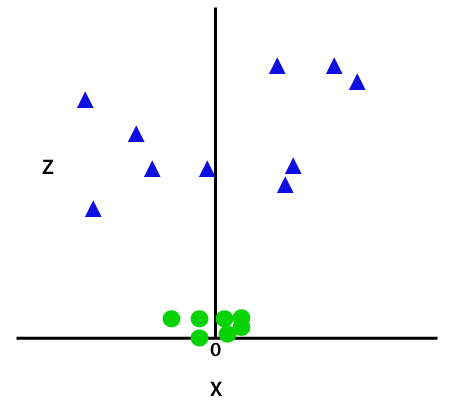


Figure 3.7. Non-Linear SVM data seperation

So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Figure 3.8. Non-Linear SVM best hyperplane

Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:

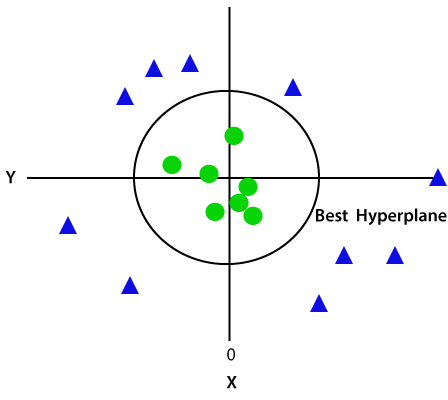


Figure 3.9. Non-Linear SVM with ROC

Hence, we get a circumference of radius 1 in case of non-linear data.

**Disadvantages of support vector machine:**

* Support vector machine algorithm is not acceptable for large data sets.
* It does not execute very well when the data set has more sound i.e. target classes are overlapping.
* In cases where the number of properties for each data point outstrips the number of training data specimens, the support vector machine will underperform.

As the support vector classifier works by placing data points, above and below the classifying hyperplane there is no probabilistic clarification for the classification

**CHAPTER 4**

**PROPOSED SYSTEM**

**4.1 OVERVIEW**

This project implements a graphical user interface (GUI) application for hop plant image classification. The project involves using machine learning techniques, specifically Multilayer Perceptron (MLP) and Convolutional Neural Network (CNN) models, to classify images of hop plants into different categories such as pests, nutrient-related issues, healthy plants, and various diseases (powdery and downy mildew). Here's an overview of the project:

* GUI Setup: The project uses Tkinter, a Python GUI library, to create a graphical interface. The GUI includes buttons for different functionalities and a text area to display information.
* Dataset Handling: The application allows users to upload a dataset of hop plant images. The dataset is expected to have subdirectories corresponding to different categories of images.
* Image Processing: After uploading the dataset, the code provides functionality for image processing and normalization. This likely includes resizing images to a consistent size and converting them to a format suitable for model training.
* MLP Classifier: The project employs a Multilayer Perceptron (MLP) classifier to train a machine learning model. The MLP model is trained on features extracted from the images. The training process involves splitting the dataset into training and testing sets, evaluating accuracy, and displaying a confusion matrix.
* CNN Model: In addition to the MLP classifier, there's a Convolutional Neural Network (CNN) model. CNNs are commonly used for image classification tasks. The code defines a CNN architecture, trains the model, saves its weights and architecture, and displays the accuracy of the trained model.
* Test Image Prediction: The application allows users to upload a test image for classification using the pre-trained models (MLP and CNN). The predicted class is displayed on the image using OpenCV.
* Graphical Representation: The GUI includes a button to plot a graph showing the accuracy and loss over training iterations for the CNN model.

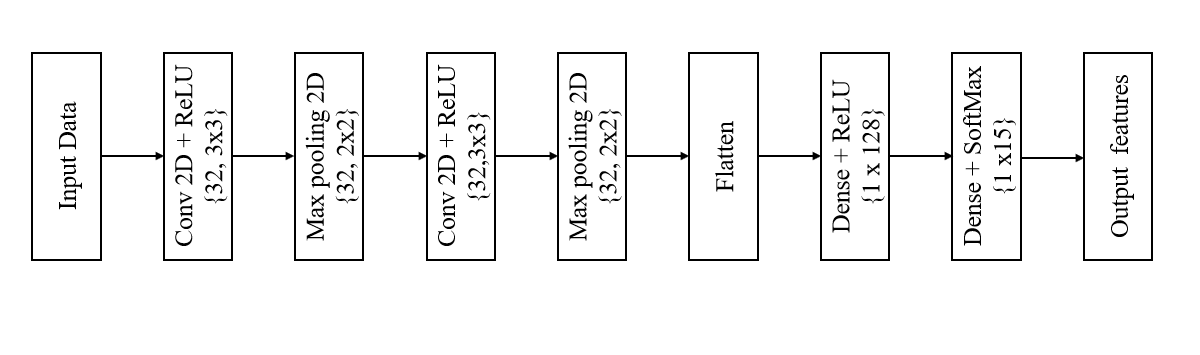
****

Figure 4.1: Proposed Architecture Diagram of Hops Classification Model.

**4.2 DATA PREPROCESSING**

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task.

***Why do we need Data Pre-processing?***

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

* Getting the dataset
* Importing libraries
* Importing datasets
* Finding Missing Data
* Encoding Categorical Data
* Splitting dataset into training and test set
* Feature scaling

**4.2.1 SPLITTING THE DATASET INTO THE TRAINING SET AND TEST SET**

In machine learning data pre-processing, we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model.

Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models.

If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

A picture containing shape

Description automatically generated

**Training** **Set**: A subset of dataset to train the machine learning model, and we already know the output.

**Test** **set**: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

**4.3 KLDA**

The principle of KLDA can be illustrated in below Fig.2. Owing to the severe non-linearity, it is difficult to directly compute the discriminating features between the two classes of patterns in the original input space (left). By defining a non-linear mapping from the input space to a high-dimensional feature space (right), we (expect to) obtain a linearly separable distribution in the feature space. Then LDA, the linear technique, can be performed in the feature space to extract the most significant discriminating features. However, the computation may be problematic or even impossible in the feature space owing to the high dimensionality. By introducing a kernel function which corresponds to the non-linear mapping, all the computation can conveniently be carried out in the input space. The problem can be finally solved as an eigen-decomposition problem like PCA, LDA and KPCA.

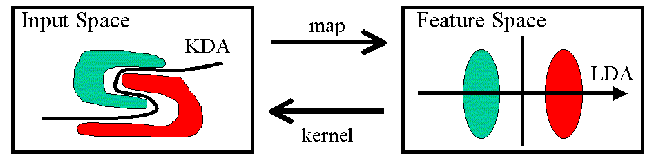


Fig. 4.2: Kernel discriminant analysis.

**4.4 CNN**

According to the facts, training and testing of CNN involves in allowing every source data via a succession of convolution layers by a kernel or filter, rectified linear unit (ReLU), max pooling, fully connected layer and utilize SoftMax layer with classification layer to categorize the objects with probabilistic values ranging from.

Convolution layer is the primary layer to extract the features from a source image and maintains the relationship between pixels by learning the features of image by employing tiny blocks of source data. It’s a mathematical function which considers two inputs like source image where and denotes the spatial coordinates i.e., number of rows and columns. d is denoted as dimension of an image (here d=3 since the source image is RGB) and a filter or kernel with similar size of input image and can be denoted as ..

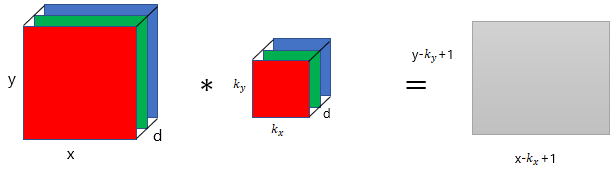
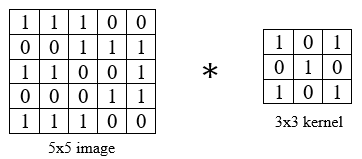


Fig. 4.3: Representation of convolution layer process.

The output obtained from convolution process of input image and filter has a size of , which is referred as feature map. Let us assume an input image with a size of 5×5 and the filter having the size of 3×3. The feature map of input image is obtained by multiplying the input image values with the filter values.



(a)

A picture containing diagram

Description automatically generated

(b)

Fig. 4.4: Example of convolution layer process (a) an image with size 5×5 is convolving with 3×3 kernel (b) Convolved feature map.

**ReLU layer**

Networks those utilizes the rectifier operation for the hidden layers are cited as rectified linear unit (ReLU). This ReLU function is a simple computation that returns the value given as input directly if the value of input is greater than zero else returns zero. This can be represented as mathematically using the function over the set of 0 and the input x as follows:

**Max pooing layer**

This layer mitigates the number of parameters when there are larger size images. This can be called as subsampling or down sampling that mitigates the dimensionality of every feature map by preserving the important information. Max pooling considers the maximum element form the rectified feature map.

**Advantages of proposed system**

* CNNs do not require human supervision for the task of identifying important features.
* They are very accurate at image recognition and classification.
* Weight sharing is another major advantage of CNNs.
* Convolutional neural networks also minimize computation in comparison with a regular neural network.
* CNNs make use of the same knowledge across all image locations.

**CHAPTER 5**

**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process also be added to; or associated with, UML.

The Unified Modeling Language Is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:** The Primary goals in the design of the UML are as follows:

* Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development process.
* Provide a formal basis for understanding the modeling language.
* Encourage the growth of OO tools market.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

**5.1 CLASS DIAGRAM**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an “is-a” or “has-a” relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed “methods” of the class. Apart from this, each class may have certain “attributes” that uniquely identify the class.

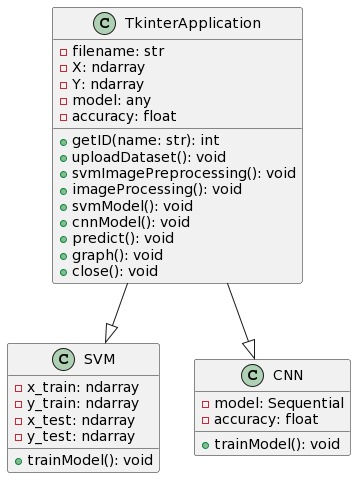


Fig 5.1: Class Diagram

**5.2 SEQUENCE DIAGRAM**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines (“lifelines”), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

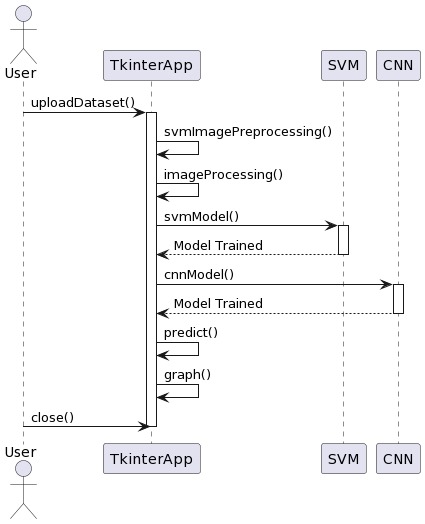


Fig 5.2: Sequence Diagram

**5.3 USE CASE DIAGRAM:**

The purpose of use case diagram is to capture the dynamic aspect of a system.

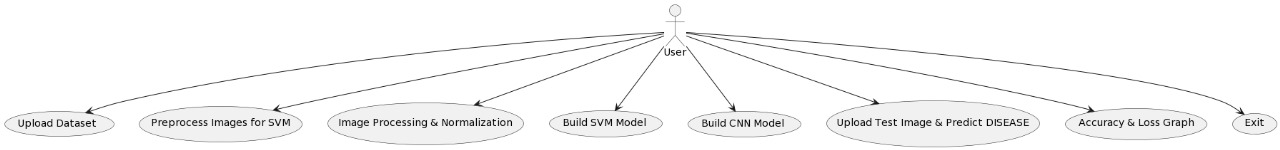


Fig 5.3: Use Case Diagram

**CHAPTER 6**

**SOFTWARE ENVIRONMENT**

**6.1 WHAT IS PYTHON?**

Below are some facts about Python.

* Python is currently the most widely used multi-purpose, high-level programming language.
* Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
* Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
* Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

* Machine Learning
* GUI Applications (like Kivy, Tkinter, PyQt etc. )
* Web frameworks like Django (used by YouTube, Instagram, Dropbox)
* Image processing (like Opencv, Pillow)
* Web scraping (like Scrapy, BeautifulSoup, Selenium)
* Test frameworks
* Multimedia

**6.2 ADVANTAGES OF PYTHON**

Let’s see how Python dominates over other languages.

1. **Extensive Libraries**

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

**2. Extensible**

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

**3. Embeddable**

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

**4. Improved Productivity**

The language’s simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

**5. IOT Opportunities**

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

**6. Simple and Easy**

When working with Java, you may have to create a class to print ‘Hello World’. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

**7. Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

**8. Object-Oriented**

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

**9. Free and Open-Source**

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

**10. Portable**

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

**11. Interpreted**

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

**Advantages of Python Over Other Languages**

**1.Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

**2. Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

**3. Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

**6.3 DISADVANTAGES OF PYTHON**

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

1. **Speed Limitations**

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

**2. Weak in Mobile Computing and Browsers**

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

**3. Design Restrictions**

As you know, Python is dynamically typed. This means that you don’t need to declare the type of variable while writing the code. It uses duck-typing. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

**4. Underdeveloped Database Access Layers**

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

**5. Simple**

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

**6.4 MODULES USED IN PROJECT**

**NumPy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and Ipython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with Ipython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

**Install Python Step-by-Step in Windows and Mac**

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

**How to Install Python on Windows and Mac**

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

**Download the Correct version into the system**

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: <https://www.python.org>

A screenshot of a computer

Description automatically generated with medium confidence

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Graphical user interface, application

Description automatically generated

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Graphical user interface, application

Description automatically generated

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Graphical user interface, text

Description automatically generated

* To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
* To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

**Installation of Python**

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.

Graphical user interface, text, application

Description automatically generated

Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.

Graphical user interface, text, application, chat or text message

Description automatically generated

Step 3: Click on Install NOW After the installation is successful. Click on Close.

Graphical user interface, text, application, chat or text message

Description automatically generated

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

**Verify the Python Installation**

Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.

Graphical user interface, application

Description automatically generated

Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.

A screenshot of a computer

Description automatically generated with medium confidence

Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

**Check how the Python IDLE works**

Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.

Application

Description automatically generated with low confidence

Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save

Graphical user interface, text, application, email

Description automatically generated

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print (“Hey World”) and Press Enter.

Graphical user interface, text, application, email

Description automatically generated

You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won’t work.

**CHAPTER 7**

**SYSTEM REQUIREMENTS**

**7.1 SOFTWARE REQUIREMENTS**

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

* Python IDLE 3.7 version (or)
* Anaconda 3.7 (or)
* Jupiter (or)
* Google colab

**7.2 HARDWARE REQUIREMENTS**

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

* Operating system : Windows, Linux
* Processor : minimum intel i3
* Ram : minimum 4 GB
* Hard disk : minimum 250GB

**CHAPTER 8**

**FUNCTIONAL REQUIREMENTS**

**8.1 OUTPUT DESIGN**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Outputs, whose destination is outside the organization
* Internal Outputs whose destination is within organization and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.

**8.2 OUTPUT DEFINITION**

The outputs should be defined in terms of the following points:

* Type of the output
* Content of the output
* Format of the output
* Location of the output
* Frequency of the output
* Volume of the output
* Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

**8.3 KEY ASPECTS OF INPUT DESIGN**

**INPUT DESIGN**

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**INPUT STAGES**

The main input stages can be listed as below:

* Data recording
* Data transcription
* Data conversion
* Data verification
* Data control
* Data transmission
* Data validation
* Data correction

**INPUT TYPES**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

* External inputs, which are prime inputs for the system.
* Internal inputs, which are user communications with the system.
* Operational, which are computer department’s communications to the system?
* Interactive, which are inputs entered during a dialogue.

**INPUT MEDIA**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

* Type of input
* Flexibility of format
* Speed
* Accuracy
* Verification methods
* Rejection rates
* Ease of correction
* Storage and handling requirements
* Security
* Easy to use
* Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

**8.4 ERROR MANAGEMENT AND DATA VALIDATION**

**ERROR AVOIDANCE**

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

**ERROR DETECTION**

Even though every effort is make to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

**DATA VALIDATION**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

**8.5 USER INTERFACE DESIGN**

It is essential to consult the system users and discuss their needs while designing the user interface:

**User Interface Systems Can Be Broadly Classified As:**

* User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
* Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

**USER INITIATED INTERFACES**

User initiated interfaces fall into two approximate classes:

* Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
* Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

**COMPUTER-INITIATED INTERFACES**

The following computer – initiated interfaces were used:

* The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
* Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

**8.6 ERROR MESSAGE DESIGN**

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

**8.7 PERFORMANCE REQUIREMENTS**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system
* The system should be accurate
* The system should be better than the existing system
* The existing system is completely dependent on the user to perform all the duties.

**CHAPTER 9**

**SOURCE CODE**

from tkinter import \*

from tkinter import simpledialog

import tkinter

import warnings

warnings.filterwarnings('ignore')

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from tkinter import ttk

from tkinter import filedialog

from keras.utils.np\_utils import to\_categorical

from keras.models import Sequential

from keras.layers.core import Dense,Activation,Dropout, Flatten

from sklearn.metrics import accuracy\_score

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

import seaborn as sns

import os

import cv2

import joblib

from sklearn.neural\_network import MLPClassifier

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

from keras.layers import Convolution2D

from keras.layers import MaxPooling2D

import pickle

from keras.models import model\_from\_json

main = Tk()

main.title("Advanced Hops Classification for Precision Crop Management and Quality Harvests")

main.geometry("1300x1200")

global filename

global X, Y

global model

global accuracy

shapes = ['Pests','Nutrient','Healthy','Disease-Powdery','Disease-Downy']

def getID(name):

# Replace this with your implementation to assign labels based on the directory name

# For example:

if name == "Pests":

return 0

elif name == "Nutrient":

return 1

elif name == "Healthy":

return 2

elif name == "Disease-Powdery":

return 3

elif name == "Disease-Downy":

return 4 # Return an appropriate label for other cases

else:

return -1

def uploadDataset():

global X, Y

global filename

text.delete('1.0', END)

filename = filedialog.askdirectory(initialdir=".")

text.insert(END,'dataset loaded\n')

def imageProcessing():

text.delete('1.0', END)

global X, Y

#global speech\_X, speech\_Y

'''

X = []

Y = []

for root, dirs, directory in os.walk(filename):

for j in range(len(directory)):

name = os.path.basename(root)

print(name+" "+root+"/"+directory[j])

if 'Thumbs.db' not in directory[j]:

img = cv2.imread(root+"/"+directory[j])

img = cv2.resize(img, (64,64))

im2arr = np.array(img)

im2arr = im2arr.reshape(64,64,3)

X.append(im2arr)

Y.append(getID(name))

X = np.asarray(X)

Y = np.asarray(Y)

print(Y)

X = X.astype('float32')

X = X/255

test = X[3]

test = cv2.resize(test,(400,400))

cv2.imshow("aa",test)

cv2.waitKey(0)

indices = np.arange(X.shape[0])

np.random.shuffle(indices)

X = X[indices]

Y = Y[indices]

Y = to\_categorical(Y)

np.save('model/X.txt',X)

np.save('model/Y.txt',Y)

'''

X = np.load('model/X.txt.npy')

Y = np.load('model/Y.txt.npy')

text.insert(END,"Total number of images found in dataset is : "+str(len(X))+"\n")

text.insert(END,"Total hops classes found in dataset is : "+str(shapes)+"\n")

def mlp():

global x\_test,x\_train,y\_test,y\_train,df

global classifier

global accuracy

#text.delete('1.0', END)

Categories = ['Pests','Nutrient','Healthy','Disease-Powdery','Disease-Downy']

flat\_data\_arr = [] #input array

target\_arr = [] #output array

datadir = r"Dataset"

flat\_data\_file = os.path.join(datadir, 'flat\_data.npy')

target\_file = os.path.join(datadir, 'target.npy')

if os.path.exists(flat\_data\_file) and os.path.exists(target\_file):

# Load the existing arrays

flat\_data = np.load(flat\_data\_file)

target = np.load(target\_file)

#dataframe

df = pd.DataFrame(flat\_data)

df['Target'] = target #associated the numerical representation of the category (index) with the actual image data

df

#input data

x = df.iloc[:,:-1]

#output data

y = df.iloc[:,-1]

# Splitting the data into training and testing sets

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.20,random\_state=77)

# Load the model from the pkl file

mlp\_classifier = joblib.load('model/MLPClassifier\_model\_weights.pkl')

y\_pred = mlp\_classifier.predict(x\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

acc = accuracy \* 100

text.insert(END,"MLP Classifier Accuracy is = "+str(acc)+"\n")

cm = confusion\_matrix(y\_test, y\_pred)

class\_names = ('Pests','Nutrient','Healthy','Disease-Powdery','Disease-Downy')

sns.heatmap(cm, annot = True, cmap = "Blues", xticklabels = class\_names, yticklabels = class\_names)

plt.xlabel("Predicted")

plt.ylabel("True")

plt.title("Confusion Matrix of MLP Classifier")

plt.show()

else:

# Data Loading & Preprocessing

Categories = ['Pests','Nutrient','Healthy','Disease-Powdery','Disease-Downy']

flat\_data\_arr = [] #input array

target\_arr = [] #output array

datadir = r"Dataset"

#path which contains all the categories of images

for i in Categories:

print(f'loading... category : {i}')

path = os.path.join(datadir,i)

for img in os.listdir(path):

img\_array=imread(os.path.join(path,img))

img\_resized=resize(img\_array,(150,150,3)) #resized to have a width of 150 pixels, a height of 150 pixels, and 3 color channels (r,g,b) helps to ensure that all images in your dataset have the same dimensions,

flat\_data\_arr.append(img\_resized.flatten()) #an image represented as a 2D or 3D array is converted into a 1D array.

target\_arr.append(Categories.index(i))

print(f'loaded category:{i} successfully')

flat\_data=np.array(flat\_data\_arr)

target=np.array(target\_arr)

np.save(os.path.join(datadir, 'flat\_data.npy'), flat\_data)

np.save(os.path.join(datadir, 'target.npy'), target)

def cnnModel():

global model

global accuracy

#text.delete('1.0', END)

if os.path.exists('model/model.json'):

with open('model/model.json', "r") as json\_file:

loaded\_model\_json = json\_file.read()

model = model\_from\_json(loaded\_model\_json)

json\_file.close()

model.load\_weights("model/model\_weights.h5")

model.\_make\_predict\_function()

print(model.summary())

f = open('model/history.pckl', 'rb')

accuracy = pickle.load(f)

f.close()

acc = accuracy['accuracy']

acc = acc[9] \* 100

text.insert(END,"Deep Learning Model Accuracy is = "+str(acc))

else:

model = Sequential() #resnet transfer learning code here

model.add(Convolution2D(32, 3, 3, input\_shape = (64, 64, 3), activation = 'relu'))

model.add(MaxPooling2D(pool\_size = (2, 2)))

model.add(Convolution2D(32, 3, 3, activation = 'relu'))

model.add(MaxPooling2D(pool\_size = (2, 2)))

model.add(Flatten())

model.add(Dense(output\_dim = 256, activation = 'relu'))

model.add(Dense(output\_dim = 5, activation = 'softmax'))

model.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])

print(model.summary())

hist = model.fit(X, Y, batch\_size=16, epochs=10, validation\_split=0.2, shuffle=True, verbose=2)

model.save\_weights('model/model\_weights.h5')

model\_json = model.to\_json()

with open("model/model.json", "w") as json\_file:

json\_file.write(model\_json)

json\_file.close()

f = open('model/history.pckl', 'wb')

pickle.dump(hist.history, f)

f.close()

f = open('model/history.pckl', 'rb')

accuracy = pickle.load(f)

f.close()

acc = accuracy['accuracy']

acc = acc[9] \* 100

text.insert(END,"CNN Model Prediction Accuracy = "+str(acc))

def predict():

global model

filename = filedialog.askopenfilename(initialdir="testImages")

img = cv2.imread(filename)

img = cv2.resize(img, (64,64))

im2arr = np.array(img)

im2arr = im2arr.reshape(1,64,64,3)

test = np.asarray(im2arr)

test = test.astype('float32')

test = test/255

preds = model.predict(test)

predict = np.argmax(preds)

img = cv2.imread(filename)

img = cv2.resize(img, (500,500))

cv2.putText(img, 'Hop Classified as : '+shapes[predict], (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,0.7, (0, 255, 0), 2)

cv2.imshow('Hop Classified as : '+shapes[predict], img)

cv2.waitKey(0)

def graph():

acc = accuracy['accuracy']

loss = accuracy['loss']

plt.figure(figsize=(10,6))

plt.grid(True)

plt.xlabel('Iterations')

plt.ylabel('Accuracy/Loss')

plt.plot(acc, 'ro-', color = 'green')

plt.plot(loss, 'ro-', color = 'blue')

plt.legend(['Accuracy', 'Loss'], loc='upper left')

#plt.xticks(wordloss.index)

plt.title('Performance Evaluation')

plt.show()

def close():

main.destroy()

font = ('times', 15, 'bold')

title = Label(main, text='Advanced Hops Classification for Precision Crop Management and Quality Harvests')

title.config(bg='powder blue', fg='olive drab')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 13, 'bold')

ff = ('times', 12, 'bold')

uploadButton = Button(main, text="Upload Dataset", command=uploadDataset)

uploadButton.place(x=20,y=100)

uploadButton.config(font=ff)

processButton = Button(main, text="Image Processing & Normalization", command=imageProcessing)

processButton.place(x=20,y=150)

processButton.config(font=ff)

mlpButton = Button(main, text="MLP Classifier", command=mlp)

mlpButton.place(x=20,y=200)

mlpButton.config(font=ff)

modelButton = Button(main, text="Build & Train Deep CNN Model", command=cnnModel)

modelButton.place(x=20,y=250)

modelButton.config(font=ff)

predictButton = Button(main, text="Upload Test Image & Classify", command=predict)

predictButton.place(x=20,y=300)

predictButton.config(font=ff)

graphButton = Button(main, text="Accuracy & Loss Graph", command=graph)

graphButton.place(x=20,y=350)

graphButton.config(font=ff)

exitButton = Button(main, text="Exit", command=close)

exitButton.place(x=20,y=400)

exitButton.config(font=ff)

font1 = ('times', 12, 'bold')

text=Text(main,height=30,width=85)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=450,y=100)

text.config(font=font1)

main.config()

main.mainloop()

**CHAPTER 10**

**RESULTS AND DISCUSSION**

**10.1 IMPLEMENTATION DESCRIPTION**

This project uses the Tkinter library to create a GUI for an application related to image classification of hop plants. Here is an overview of the code:

* Importing Libraries: Various libraries are imported, including Tkinter for GUI, Matplotlib for plotting, NumPy for numerical operations, Pandas for data manipulation, OpenCV for image processing, Keras for deep learning, Scikit-learn for machine learning, and Seaborn for statistical data visualization.
* Setting Up GUI: The script creates a Tkinter window named main with a title and specific dimensions.
* Global Variables: Several global variables are declared, including filename, X, Y, model, and accuracy. These variables seem to be used throughout the script to store information about the dataset, image data, trained models, and accuracy metrics.
* Function Definitions: Several functions are defined to perform specific tasks:
* getID(name): Returns a numerical label based on the provided category name.
* uploadDataset(): Prompts the user to select a dataset directory and updates the GUI accordingly.
* imageProcessing(): Processes images in the dataset, potentially resizing and reshaping them, and displays information about the dataset in the GUI.
* mlp(): Utilizes a Multilayer Perceptron (MLP) classifier to train a model, calculate accuracy, and display a confusion matrix using Seaborn.
* cnnModel(): Defines and trains a Convolutional Neural Network (CNN) model, saving the model architecture, weights, and training history. Displays accuracy information in the GUI.
* predict(): Allows the user to upload a test image, classifies it using a pre-trained model, and displays the result in an OpenCV window.
* graph(): Plots a graph showing accuracy and loss over training iterations.
* GUI Components: The GUI includes buttons for various actions, such as uploading the dataset, image processing, using MLP, training a CNN model, predicting with a test image, and displaying graphs. Additionally, there is a text area for displaying information and messages.
* Main Event Loop: The script enters the Tkinter event loop (main.mainloop()) to handle user interactions and GUI updates.

**10.2 DATASET DESCRIPTION**

The dataset contains total of 1100 images with 165 images in Disease-Downy class, 105 images in Disease-Powdery class,528 images in Healthy class, 52 images in Nutrient class and 250 images in Pests class

Table 10.1: Dataset description.

|  |  |  |
| --- | --- | --- |
| S. No. | Number of images | Class type |
| 1 | 165 | Disease-Downy |
| 2 | 105 | Disease-Powdery |
| 3 | 528 | Healthy |
| 4 | 52 | Nutrient |
| 5 | 250 | Pests |

**10.3 RESULTS AND DESCRIPTION**

Figure 1: User interface application of proposed Deep Learning Approach for Hops Classification shows a screenshot or visual representation of the graphical user interface (GUI) of the application created for hops classification. It may include buttons, input fields, and other elements for user interaction.

Figure 2: Illustrates the image after uploading the dataset displays an image or images from the dataset after it has been uploaded. It provides a visual representation of the data that will be used for training and testing the classification models.

Figure 3: Illustration and Description of classes after preprocessing show examples of images from different classes (Pests, Nutrient, Healthy, Disease-Powdery, Disease-Downy) after preprocessing. It could include visualizations or statistics that describe how the images have been prepared for the machine learning models.



Figure 10.1: User interface application of proposed Deep learning Approach for Hops Classification.



Figure 10.2: Illustrates the image after uploading the dataset.

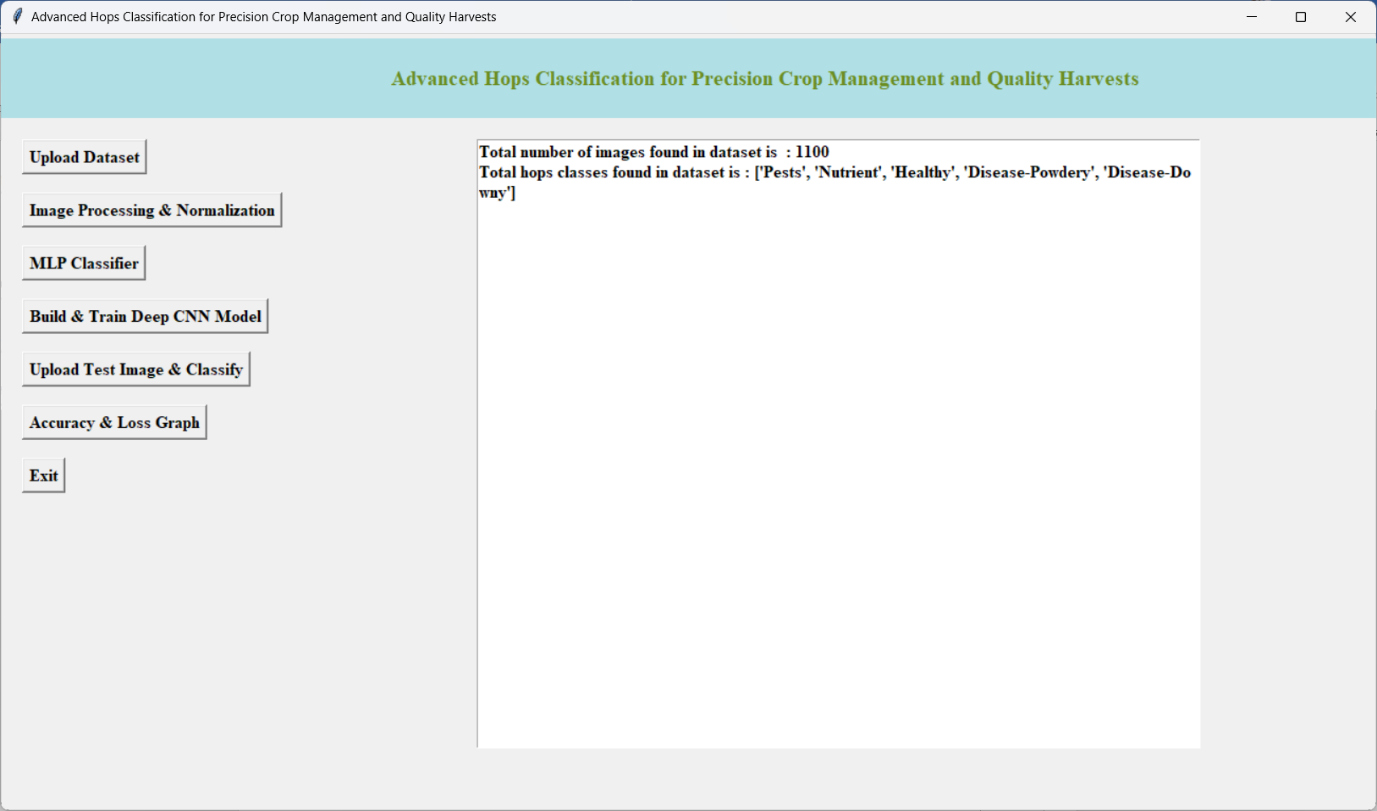


Figure 10.3: Illustration and Description of classes after preprocessing.

Figure 4: Model accuracy of Existing MLP Classifier presents a visualization or numerical representation of the accuracy achieved by the existing Multi-Layer Perceptron (MLP) classifier. It might include metrics indicating how well the model performs on the validation or test set.

Figure 5: Confusion matrix of existing MLP Classifier a graphical representation of the confusion matrix for the existing MLP classifier. The confusion matrix shows how well the classifier performs in terms of true positives, true negatives, false positives, and false negatives for each class.

Figure 6: Model Accuracy of proposed Deep Learning CNN Algorithm shows the accuracy achieved by the proposed Convolutional Neural Network (CNN) algorithm. It could include metrics indicating the performance of the CNN model on the validation or test set.

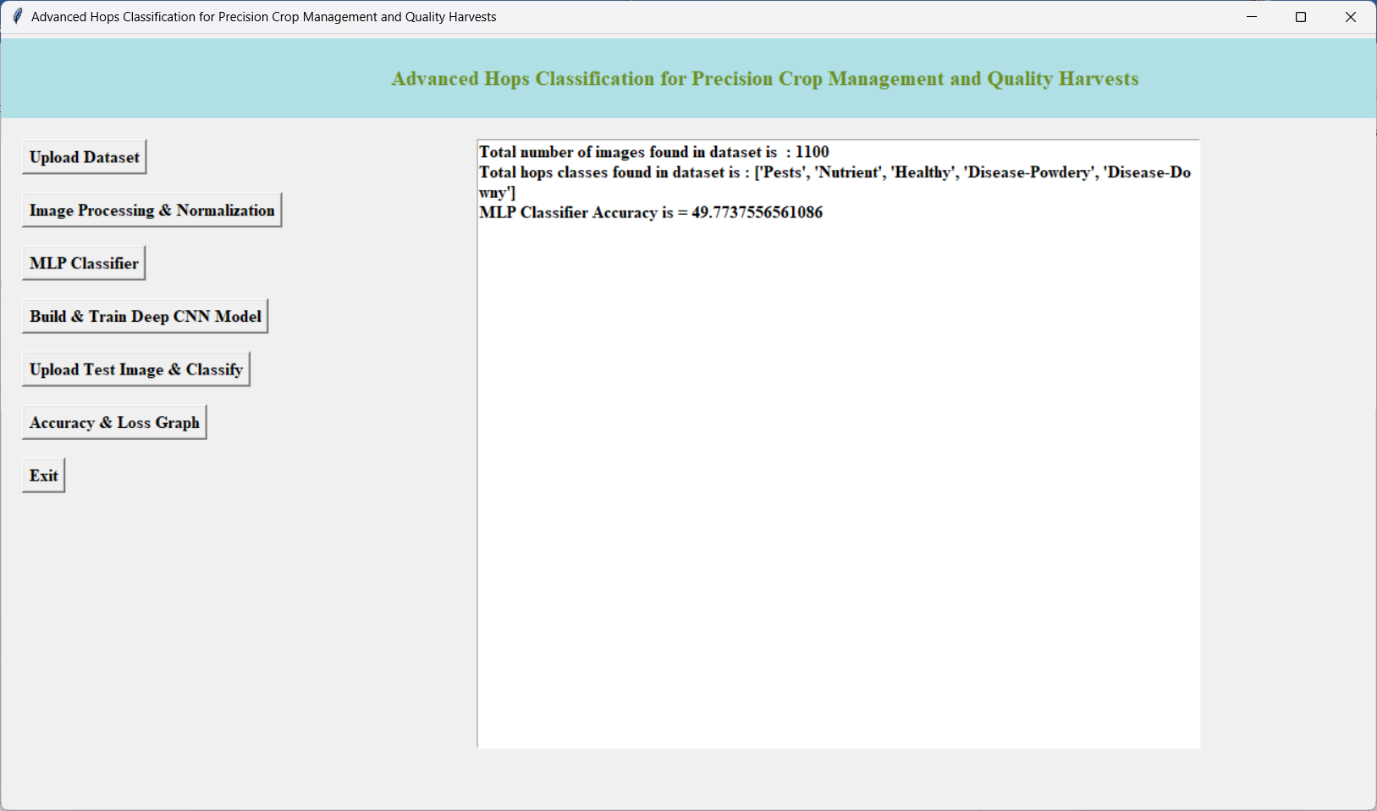


Figure 10.4: Model accuracy of Existing MLP Classifier.

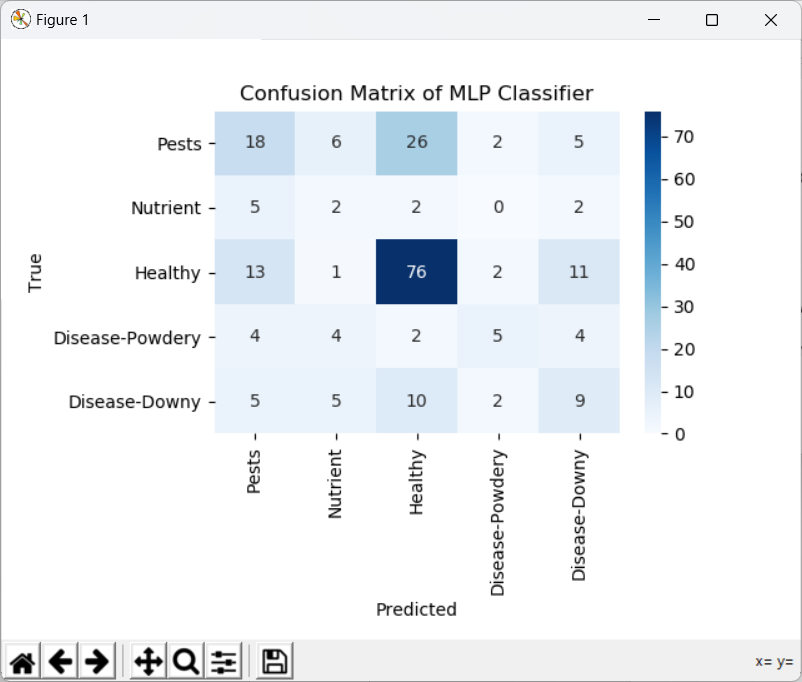


Figure 10.5: confusion matrix of existing MLP Classifier

Figure 7: Model summary of proposed CNN Architecture: provides a summary or visualization of the architecture of the proposed CNN model. It may include details such as the number of layers, type of layers, and the flow of information through the network.

Figure 8: Prediction Results of Proposed CNN Algorithm: displays the results of the predictions made by the proposed CNN algorithm on sample images. It might show the original images alongside the predicted classes.

Figure 9: Accuracy and loss graph for Proposed Model graph depicting the accuracy and loss over training epochs for the proposed CNN model. It provides insights into how well the model is learning from the training data and how its performance changes over time.

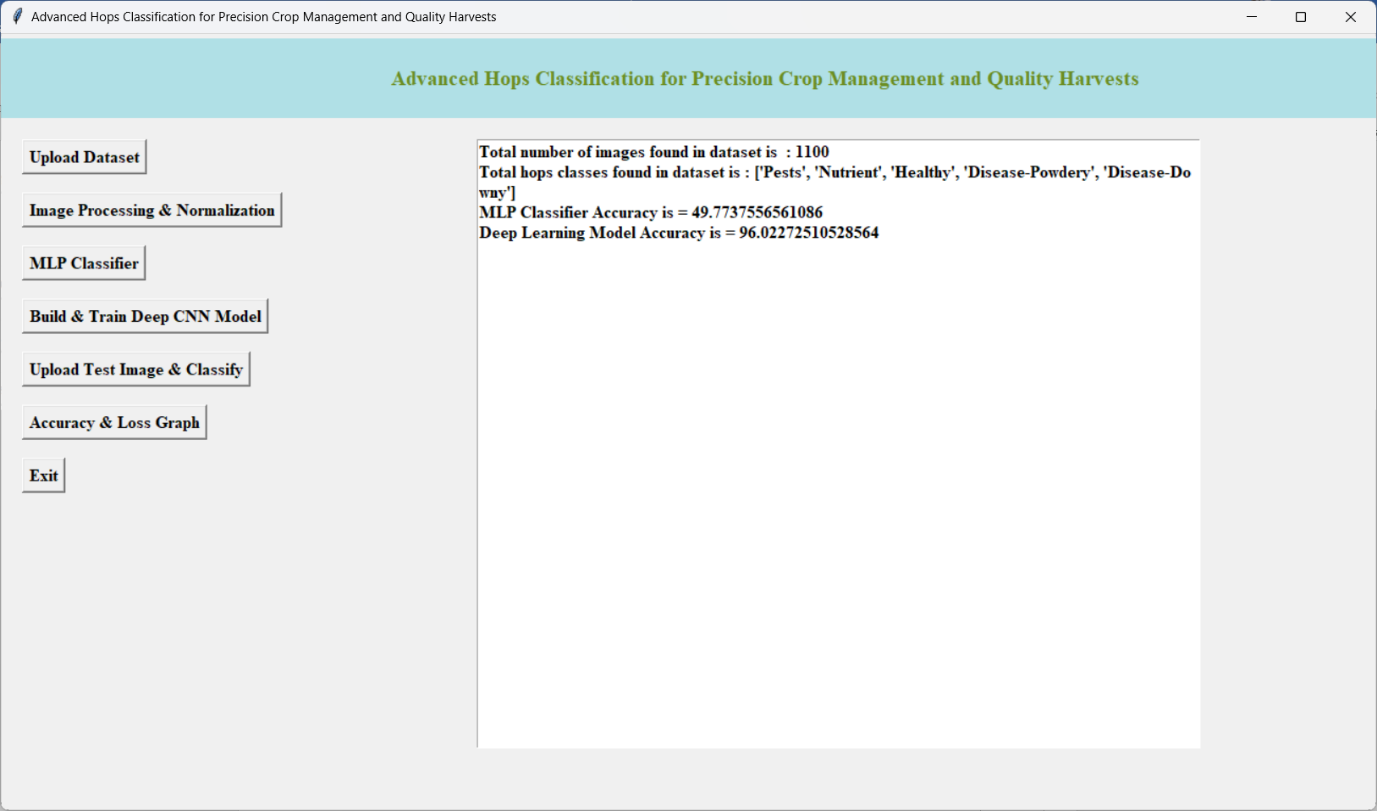


Figure 10.6: Model Accuracy of proposed Deep learning CNN Algorithm

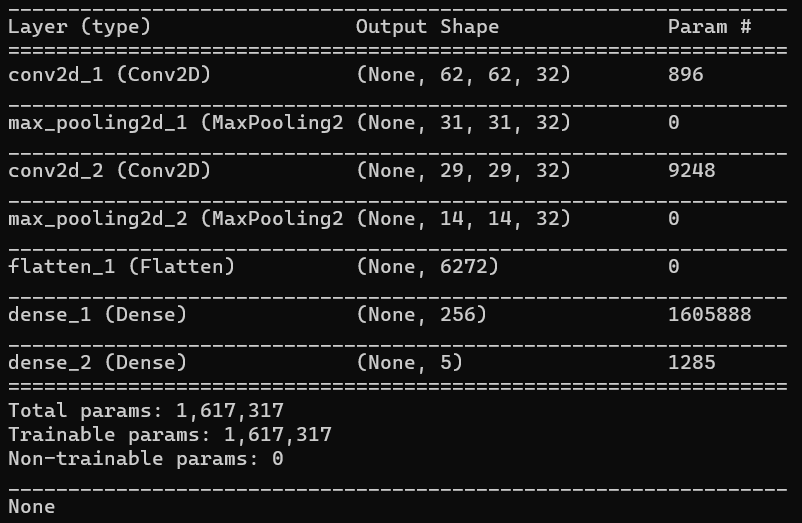
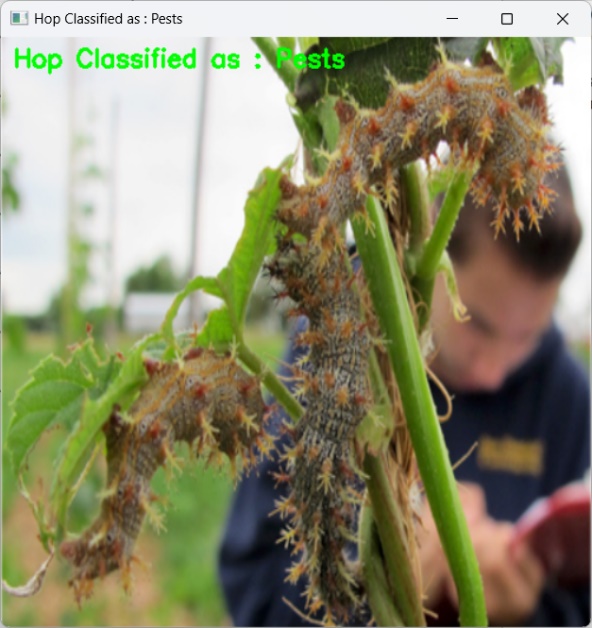
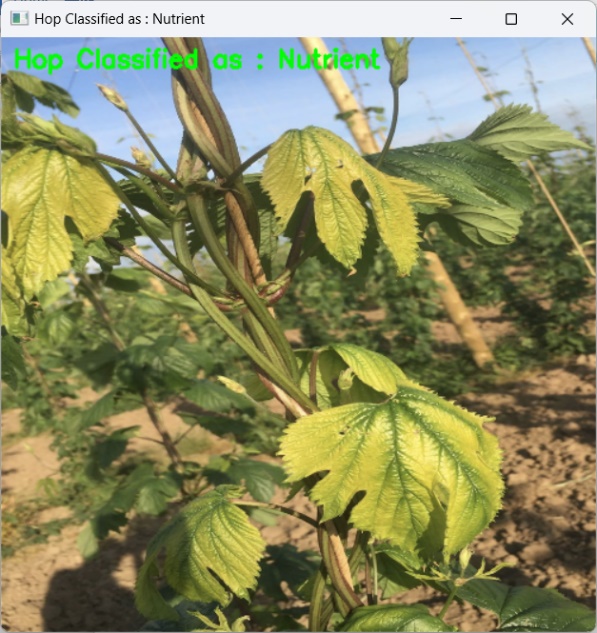


Figure 10.7: Model summary of proposed CNN Architecture

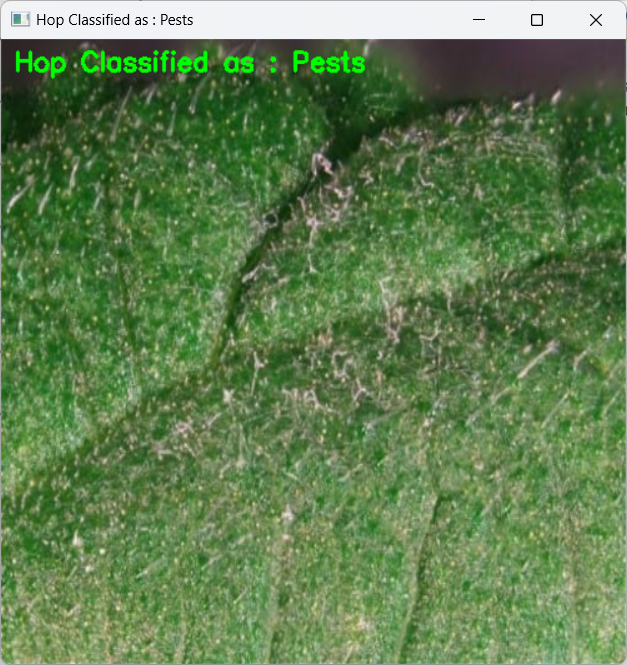
 



Figure 10.8: Prediction Results of Proposed CNN Algorithm

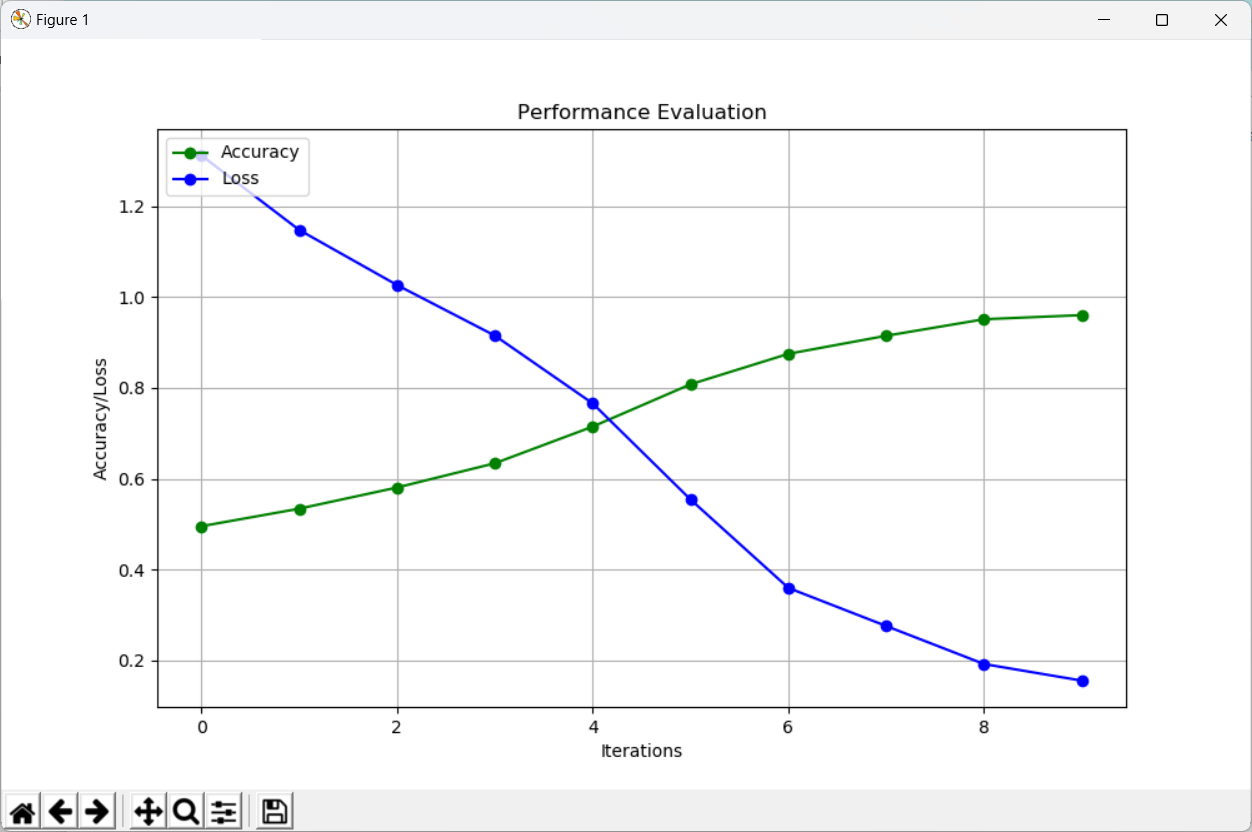


Figure 10.9: Accuracy and loss graph for Proposed Model

Table 10.2: Performance comparison of MLP Classifier and CNN model.

|  |  |  |
| --- | --- | --- |
| Model | MLP Classifier | CNN Model |
| Accuracy (%) | 79.7 | 96.0 |

**For the MLP model**:

* The Accuracy is 79.7, indicating the accuracy between the actual and predicted values

**For the CNN model**:

* The Accuracy is 96.0, indicating the accuracy between the actual and predicted values

**CHAPTER 11**

**CONCLUSION AND FUTURE SCOPE**

**11.1 CONCLUSION**

The significance of this technology lies in its potential to optimize resource allocation, enhance decision-making processes, and ultimately increase the efficiency of hops farming. With accurate classification, farmers can tailor their cultivation practices to the specific needs of each hop variety, thereby maximizing yield and minimizing resource wastage. This targeted approach to crop management is crucial for achieving sustainability in agriculture and meeting the growing demand for high-quality hops. Moreover, the implementation of the advanced hops classification system contributes to the overall improvement of crop quality. By ensuring that each hop plant receives tailored care, farmers can expect more uniform and superior yields. This not only benefits the growers but also positively impacts downstream industries, such as craft breweries, by providing a more consistent and desirable raw material for brewing.

**11.1 FUTURE SCOPE**

As we look toward the future, there are several exciting avenues for further exploration and development in this field. Firstly, continuous refinement and enhancement of the classification algorithms can improve the system's accuracy and versatility, accommodating new hop varieties and evolving agricultural practices. Additionally, integrating real-time data and monitoring capabilities into the system can enable farmers to respond promptly to changing environmental conditions and optimize their interventions accordingly. The future scope also extends to the integration of the Internet of Things (IoT) and other emerging technologies. Smart sensors and devices can be employed to gather additional data on factors like soil moisture, temperature, and nutrient levels, providing a more comprehensive understanding of the growing environment. This holistic approach can further fine-tune cultivation strategies and contribute to the development of a fully automated and adaptive hops farming system.

**REFERENCES**

1. Bartkowiak, A.M. Energy-saving and low-emission livestock buildings in the concept of a smart farming. J. Water Land Dev. 2021, 51, 272–278. [Google Scholar] [CrossRef]
2. Sagheer, A.; Mohammed, M.; Riad, K.; Alhajhoj, M. A Cloud-Based IoT Platform for Precision Control of Soilless Greenhouse Cultivation. Sensors 2021, 21, 223. [Google Scholar] [CrossRef] [PubMed]
3. Akhigbe, B.; Munir, K.; Akinade, O.; Akanbi, L.; Oyedele, L. IoT Technologies for Livestock Management: A Review of Present Status, Opportunities, and Future Trends. Big Data Cogn. Comput. 2021, 5, 10. [Google Scholar] [CrossRef]
4. Mahbub, M. A smart farming concept based on smart embedded electronics, internet of things and wireless sensor network. Internet Things 2020, 9, 100161. [Google Scholar] [CrossRef]
5. Andrianto, H.; Suhardi; Faizal, A. Development of smart greenhouse system for hydroponic agriculture. In Proceedings of the 2020 International Conference on Information Technology Systems and Innovation, Padang, Indonesia, 19–23 October 2020; pp. 335–340. [Google Scholar] [CrossRef]
6. Latino, M.E.; Menegoli, M.; Corallo, A. Agriculture Digitalization: A Global Examination Based on Bibliometric Analysis. IEEE Trans. Eng. Manag. 2022, 1–16. [Google Scholar] [CrossRef]
7. Ullah, M.; Narayanan, A.; Wolff, A.; Nardelli, P.H.J. Unified Framework to Select an IoT Platform for Industrial Energy Management Systems. In Proceedings of the 2021 44th International Convention on Information, Communication and Electronic Technology, Opatija, Croatia, 27 September–1 October 2021; pp. 950–955. [Google Scholar] [CrossRef]
8. Dastjerdi, A.V.; Buyya, R. Fog Computing: Helping the Internet of Things Realize Its Potential. Computer 2016, 49, 112–116. [Google Scholar] [CrossRef]
9. Pflanzner, T.; Hovari, M.; Vass, I.; Kertesz, A. Designing an IoT-Cloud Gateway for the Internet of Living Things. Commun. Comput. Inf. Sci. 2020, 1218, 23–41. [Google Scholar] [CrossRef]
10. Sorri, K.; Mustafee, N.; Seppänen, M. Revisiting IoT definitions: A framework towards comprehensive use. Technol. Forecast. Soc. Chang. 2022, 179, 121623. [Google Scholar] [CrossRef]
11. Vanitha, V.; Raj, C.S.; Kumar, K.M.; Sebastian, J. Design and Development of an Effective Smart Garbage System using the Internet of Things. J. Physics Conf. Ser. 2021, 2040, 012033. [Google Scholar] [CrossRef]
12. Arrubla-Hoyos, W.; Ojeda-Beltrán, A.; Solano-Barliza, A.; Rambauth-Ibarra, G.; Barrios-Ulloa, A.; Cama-Pinto, D.; Arrabal-Campos, F.M.; Martínez-Lao, J.A.; Cama-Pinto, A.; Manzano-Agugliaro, F. Precision Agriculture and Sensor Systems Applications in Colombia through 5G Networks. Sensors 2022, 22, 7295. [Google Scholar] [CrossRef]