CHAPTER 1

INTRODUCTION

Agriculture, the foundation of human civilization, has evolved to meet the increasing demands for food, fibre, and fuel. It involves crop cultivation, livestock rearing, and natural resource management. Crop management is crucial for optimizing crop production, ensuring food security, and promoting sustainable farming methods. Food security is a global concern, requiring availability, accessibility, and affordability. With the world's population expected to reach 9 billion by 2050, the pressure on agriculture to meet this demand is more significant. Innovative approaches and technologies are needed to enhance crop yield, reduce waste, and ensure a resilient and sustainable food supply chain.

In recent years, artificial intelligence (AI) and convolutional neural networks (CNN) have emerged as transformative tools in agriculture. AI-powered technologies offer the potential to revolutionize traditional farming practices by providing insights, automation, and precision in various aspects of crop management. CNN, a specific type of deep learning algorithm, excels in image recognition and pattern detection

AI-CNN-based farming involves the integration of machine learning algorithms and computer vision into agricultural practices. This technology enables farmers to monitor crop health, detect diseases, optimize irrigation, and enhance overall crop management efficiency. Drones equipped with AI CNN algorithms can capture high-resolution images of fields, allowing farmers to identify potential issues such as nutrient deficiencies or pest infestations early on.

Fig 1.1 Use of Artificial Intelligence in Agriculture

The intersection of agriculture, crop management, and AI-based farming represents a pivotal moment in the evolution of farming practices.

1.1 OBJECTIVE

The primary objective is to introduce an innovative agricultural productivity enhancement system, leveraging Convolutional Neural Networks (CNN), extensive datasets, and Python programming. The accurately predict crop yields and provide optimized fertilizer recommendations, empowering farmers with data-driven insights for improved decision-making and fostering sustainable and efficient agricultural practices. The system's predictive capabilities have been tested using real-world agricultural datasets, advancing precision agriculture and addressing environmental and economic challenges.

CHAPTER 2

LITERATURE SURVEY

1. AI DRIVEN FARMBOT FOR CROP HEALTH MONITORING AND DISEASE DETECTION

Authors: Atim, Proscovia; Birojjo, Donikamu Francis; Namuganga, Joyce; Nakyeyune, Maria Belinda; Opio, George, 2023.

The increasing population in urban areas leads to limited land for agriculture, leading to overdependence on market-based food. This results in increased food prices, leading to hunger and a decrease in crop yield. Land use patterns in urban centres are affected, with land used for agriculture risking being used for settlement and industrialization. This reduces soil quality, causing food insecurity. The farm Bot project aims to improve farming in urban areas by introducing disease detection, water stress determination, and crop growth monitoring systems. The project aims to use locally available materials to make the farm Bot affordable for people who want to have a small garden at home for constant food supply. Without improvements, poor crop yield from small-scale farms in urban areas will continue, resulting in low food production and supply.

The design of components was based on load requirements, material, and purpose. Components were fabricated, 3D printed, and connected using circuit diagrams, flow charts, and flow charts. Hardware was programmed in Raspberry Pi3 and Arduino, while software was written in C++, MATLAB, Python, and JavaScript. The system was tested at different levels, including unit testing, integral testing, and system testing, indicating its efficiency. An economic analysis showed the project's viability, with a Net Present Value greater than 1, and a payback period of about 1.5 years. The prototype demonstrated significant improvements in crop yield, leaf appearance, and productivity compared to the existing farm bot.

2. ARTIFICIAL INTELLIGENCE BASED AGRICULTURAL CHATBOT AND VIRTUAL ASSISTANT FOR DELIVERY OF HARVESTED CROPS

Authors: Sivakumar, S.A., Shankar, B.M., Anuradha, B., Karan, K.A., Karthik, A., Karthik, R. and Kumar, J.R.R, 2023.

The mobile app for farmers and transport service providers aims to pool farmers according to their requirements. Both farmers and service providers need to create an account at their first entry-level, which asks for a username, phone number, address, password, and vehicle number. These can be made available in their native language by giving their language preference. Once registered, both can use their login ID to login into the app. The app consists of 7 layouts, including opening graphics, user log-in and authentication, user information, profile creation for truck holders, g-maps for tracking the truck, and community page. The UI and colours are nature theme and agriculture background, with unique and bold fonts and Google services to support native Indian languages.

SQL (Structured Query Language) is used to manage and alter data in the database. The app incorporates Google Maps API, which helps in pooling or merging users based on their locations and in the database displayed in the app. The APIs provide analytics, machine learning as a service (Prediction API), and user data access. Efficient agriculture-based transport is crucial for the success and sustainability of the agricultural industry. In many developing countries, inadequate transportation infrastructure remains a significant challenge for the agricultural sector. Efficient logistics planning is essential to avoid bottlenecks and wastage, and climate and weather conditions can affect transportation during extreme events like floods or storms. The proposed mobile app for farmers and transport service providers aims to pool farmers according to their requirements and improve productivity.

3. AUTONOMOUS ROBOT SYSTEM DEVELOPMENT FOR HEALTH INDICATION AND MONITORING OF CROPS USING AI-ML-BASED SYSTEM

Authors: M. Suganthi, M. Mahanty, M. H T, H. K S, O. Bhimineni and S. Menon, 2023.

Robotic systems have progressed and grown more available in recent years. They have been gradually but steadily integrated and used in several procedures and industries, particularly farming, as a consequence. Nowadays, agriculture machines are being developed to take the place of humans in labour-intensive, thing, or hazardous activities. Depending on the kind of vehicle, its detectors, motors, and telecommunication networks, agro mobile robots provide a number of advantages. The most significant and significant profession on the globe is gardening. The fundamental main objective of the task is to construct an automated multifunctional agricultural robot car. It is used to boost farming efficiency while decreasing the number of individuals who labour in fields and improving the accuracy of the task. The robot structure includes a camera that provides a real-time view of the fields. We have two systems in our project:1. Crop monitoring and field health indicator2.Robot automobile driving.

The first system includes similar sensors to determine soil, warmth, and dampness. The crop then is controlled by concrete measures, and the collected data is also sent to the farmer through a Blynk app. Animals assault on rural property and human crop stealing can cause significant losses to the cultivated produce. In an agricultural region, violators may be found using a Sensor. Whenever a fire breaks out in croplands brought on by the sun's radiation or a lightning strike—we utilized fire sensors in early diagnosis and an infrared sensor to identify obstacles. A sensor was added to the robotic so that it could get a real opinion of produce and its environs. Ai systems are also being used to track farm progress and evaluate crop yield. Moreover, this technology has automatic irrigation and insecticide sprinkling.

4. DEEP LEARNING BASED COMPUTER VISION APPROACHES FOR SMART AGRICULTURAL APPLICATIONS

Authors: Dhanya, V.G., Subeesh, A., Kushwaha, N.L., Vishwakarma, D.K., Kumar, T.N., Ritika, G. and Singh, A.N, 2022.

The agriculture industry is undergoing a digital transformation with a focus on artificial intelligence and related technologies, particularly deep learning-based computer vision. Computer vision enables automated and precise agricultural activities, from land preparation to harvesting. The review categorizes recent computer vision applications in agriculture, including seed quality analysis, soil analysis, irrigation water management, plant health analysis, weed management, livestock management, and yield estimation. Convolutional neural networks are highlighted as crucial for precision and accuracy in various agriculture activities. Challenges in implementing real-time solutions include the need for high-quality datasets and addressing issues like data quality and computation power.

Deep learning-based computer vision is found to have automation capabilities across different applications in agriculture, su ch as plant health monitoring, weed detection, irrigation management, livestock management, and yield estimation. Integration of deep learning computer vision with unmanned aerial vehicles (UAV) and spectral data is suggested for advanced intelligent solutions. Despite challenges, extensive automation in agriculture activities is expected to continue attracting interest from the deep learning research community. The adoption rate of advanced technologies in agriculture has been slow, but it is on the rise, indicating a shift away from concerns about initial investment, technical expertise, and data privacy.

5. INTEGRATING SPEED BREEDING WITH ARTIFICIAL INTELLIGENCE FOR DEVELOPING CLIMATE-SMART CROPS

Authors: Krishna Kumar Rai, 2022.

The plant breeders have stumbled to develop and breed high-yielding cultivars that can withstand abiotic and biotic stresses. Noteworthy, speed breeding has emerged as a potential alternative for reducing time, space, and cost to develop, release, and commercialize new/improved cultivars with improved accuracy and predictability. Plant growth and developmental conditions are the critical factors that govern plant performance under changing environmental conditions; speed breeding protocol technically mimics the natural environment artificially (light and temperature) to accelerate plant growth. Furthermore, molecular breeding techniques like MAS and GWAS can also be successfully integrated with speed breeding protocol to identify genes/ QTLs underlying biotic/abiotic stress tolerance, nutritional qualities, and high yield. Application of Next-Gen AI has opened a new realm for speed breeding and agriculture that will enable decision making and handling of big OMICS data with great precision, which will help get novel insight into plant functions under climate extremes.

However, its application in developing countries is still lagging due to a lack of trained plant breeders, infrastructure facilities, and government support at the financial level to sustain speed breeding protocol for crop improvements. Implementation of speed breeding requires extensive planning and a continuous supply of electricity and water to maintain adequate light and temperature in the facility. Therefore, efforts should be diverted toward developing public and private ventures to facilitate capacity building, technology transfer, and finance speed breeding coupled with AI-driven research to facilitate crop improvement programs. These public private partnerships will also help create a framework for successfully implementing AI-augmented plant breeding research and innovation for the betterment of humans, animals, and the environment.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The proposed system uses the support vector machine (SVM) algorithm to forecast the maximum crop yield that farmers can accept, resulting in a sharp increase in the production ratio of a specific crop. The system is designed to be useful to local farmers and is based on machine learning technology. The user or customer registers on the website using the required credentials, enters soil details, and the system collects data. The variables used in this model include location, weather, plant, area, and productivity. The implementation can be done in two steps: dataset collection and pre-processing. Data is collected from Kaggle, which has 1600 datasets, and features such as calcium, magnesium, potassium, sulphate, nitrogen, lime, carbon, phosphorus, moisture, and target are considered. The dataset is divided into four classes or "targets" based on soil type, weather, and crop properties.

Data pre-processing is performed by removing redundant and missing values and replacing null values. The data is then trained using the machine learning algorithm, SVM, which gives maximum accuracy on the dataset. The model is saved with the given dataset and is processed when a user sends a query or request to the model. The model is constructed using SVM, and the dataset is divided into training and testing data, 80% and 20% respectively. The model is now ready to accept new queries, predict crops based on soil information, and print fertilizer details for improved crop growth. Incorporating information technology with agriculture can guide farmers to improve productivity. This proposed system works faster and provides better accuracy in prediction, predicting suitable crops and fertilizers for the field. It includes various soil parameters to analyze crops, improving productivity, growth, and plant quality.

3.2 EXISTING SYSTEM BLOCK DIAGRAM

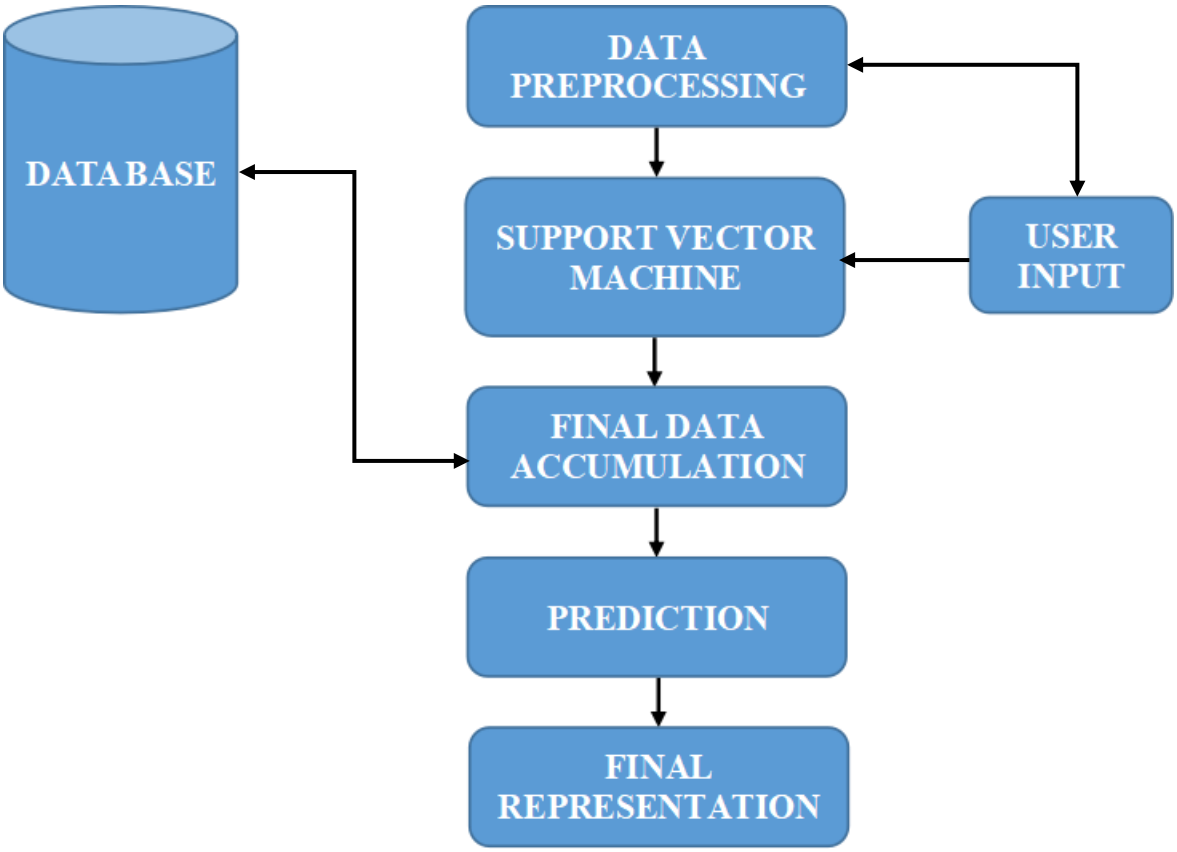


Fig 3.1 Existing System Architecture

3.2.1 Drawbacks of Existing System

- The SVM algorithm may not accurately predict different regions or conditions due to its reliance on a specific dataset.
- The system's reliance on Kaggle data may introduce biases and limitations, potentially leading to suboptimal results.
- Requiring farmers to register on the website may limit its reach and usability.
- Focusing on predicting maximum crop yield may neglect other crops, hindering overall agricultural optimization.
- The system collects and stores sensitive agricultural and location data, raising concerns about user protection.

3.3 PROPOSED SYSTEM

Crop yield prediction with fertilizer recommendations is a critical task in modern agriculture, aiming to optimize crop production while minimizing resource usage and environmental impact. Convolutional Neural Networks (CNNs) offer a powerful tool for analyzing agricultural data, particularly when dealing with image data like satellite imagery or drone footage. The proposed methodology involves obtaining a comprehensive data set including various factors affecting crop yield, such as historical yield data, soil properties, weather conditions, fertilizer usage, crop type, and other relevant variables.

CNN architecture is well-suited for analyzing spatial data like satellite images of farmland, extracting features indicative of crop health, growth stages, and potential yield by integrating fertilizer data into the model, recommendations tailored to specific conditions detected by the CNN can be developed. Python programming allows for efficient integration of CNN models with datasets, ensuring efficient data processing, feature extraction, and model training. The system's predictive capabilities are validated through real-world agricultural datasets, demonstrating accuracy and reliability in crop yield forecasting.

The Model training and validation involve splitting the data set into training, validation, and testing sets. The CNN model is trained using historical data, fine-tuning hyper parameters to prevent over fitting, and using appropriate evaluation metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), or coefficient of determination (R-squared) to assess its accuracy in predicting crop yield. It also incorporates a fertilizer recommendation component, suggesting optimal nutrient combinations for specific crops, promoting sustainable farming practices.

The system aims to improve precision agriculture by providing a comprehensive tool for farmers to make data-driven decisions. It includes data preprocessing, CNN model development, feature engineering, fertilizer recommendation system, model training and validation, and model evaluation and deployment. The trained model is evaluated for its performance in predicting crop

yield and providing accurate fertilizer recommendations. Once satisfied, it can be deployed in a production environment for real-time predictions, optimizing crop production while minimizing resource usage and environmental impact.

3.4 PROPOSED SYSTEM BLOCK DIAGRAM

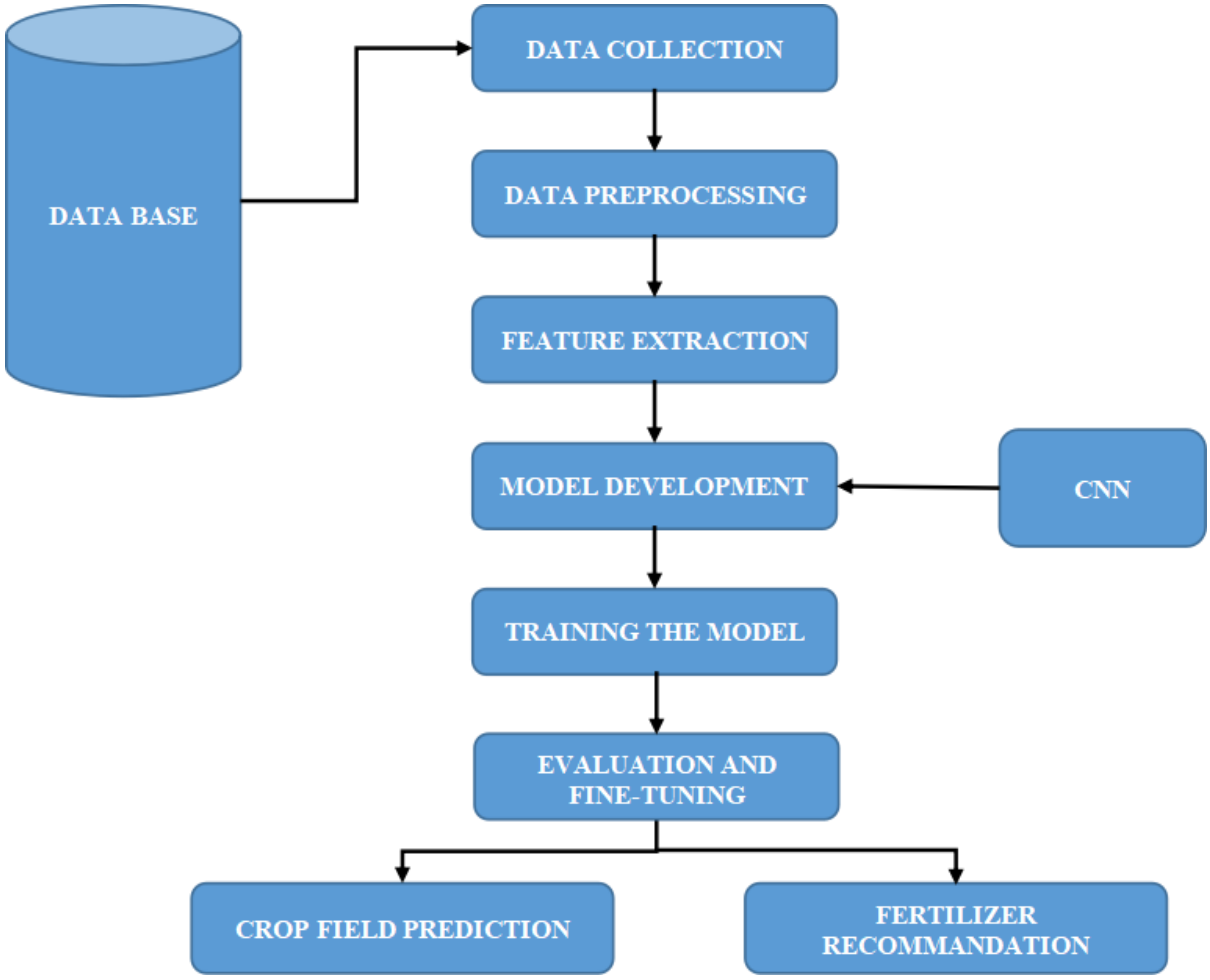


Fig 3.2 Proposed System Architecture

3.4.1 Advantages

- Optimized Resource Usage: CNNs provide tailored fertilizer application suggestions, minimizing waste and environmental impact.
- Environmental Sustainability: Integration with agricultural data allows for precise assessment of crop health, growth stages, and potential yield.
- Data-Driven Decision Making: Incorporates factors affecting crop yield, enhancing productivity and long-term planning.

- Efficient Integration with Python Programming: Python facilitates efficient integration of CNN models with agricultural datasets, ensuring streamlined data processing and model training.
- Precision Agriculture for Real-Time Decisions: Real-time predictions enable farmers to adapt practices promptly, contributing to precision agriculture.

CHAPTER 4

SYSTEM STUDY

4.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SCHEDULE FEASIBILITY
- OPERATIONAL FEASIBILITY
- LEGAL AND ETHICAL FEASIBILITY

4.1.1 ECONOMICAL FEASIBILITY

Estimate the costs associated with developing the system, including software development, data acquisition, hardware procurement, and personnel. Compare the estimated costs with the potential benefits and savings that could be achieved through optimized fertilizer recommendations. Conduct a cost-benefit analysis to determine if the project is economically feasible and justifiable.

4.1.2 TECHNICAL FEASIBILITY

Assess the availability of the necessary technology and resources required for implementing the CNN-based fertilizer recommendation system. Evaluate the feasibility of developing and training a CNN model using the available data sources. Determine if the computational infrastructure (hardware and software) is capable of handling the requirements of training.

4.1.3 SCHEDULE FEASIBILITY

Develop a realistic timeline for the development, testing, and deployment of the CNN-based fertilizer recommendation system. Identify any potential bottlenecks or dependencies that could impact the project schedule. Determine if the project can be completed within the desired timeframe and if any adjustments are needed to meet deadlines.

4.1.4 OPERATIONAL FEASIBILITY

Evaluate the practicality and ease of integrating the CNN-based fertilizer recommendation system into existing agricultural practices. Assess the readiness of farmers and agricultural professionals to adopt and utilize the system. Identify any potential operational challenges or barriers that could hinder the successful implementation of the system.

4.1.5 LEGAL AND ETHICAL FEASIBILITY

Investigate any legal regulations or compliance requirements related to data collection, storage, and usage in the agricultural sector. Ensure that the proposed system adheres to ethical guidelines and respects the privacy rights of farmers and other stakeholders. Address any potential ethical concerns or implications arising from the deployment of the system.

CHAPTER 5

SYSTEM REQUIREMENTS

5.1 HARDWARE REQUIREMENTS (Minimum requirement):

The section of hardware configuration is an important task related to the software development insufficient random access memory may affect adversely on the speed and efficiency of the entire system. The process should be powerful to handle the entire operations. The hard disk should have sufficient capacity to store the file and application.

- System : Pentium Dual Core.
- Hard Disk : 120 GB.
- External Storage : Cloud.
- Ram : 1GB.
- GPU : NVIDIA GeForce or Quadro series.
- Processor : Intel CORE i5

5.2 SOFTWARE REQUIREMENTS

A major element in building a system is the section of compatible software since the software in the market is experiencing in geometric progression. Selected software should be acceptable by the firm and one user as well as it should be feasible for the system. This document gives a detailed description of the software requirement specification. The study of requirement specification is focused specially on the functioning of the system. It allows the developer or analyst to understand the system, function to be carried out the performance level to be obtained and corresponding interfaces to be established.

- Operating system : Windows 10.
- Coding Language : Python

ADAFRUIT IO

Adafruit io is a platform designed by ADAFRUIT IO to display, respond, and interact with your project's data. Adafruit also keep your data private (data feeds are private by default) and secure (ADAFRUIT will never sell or give this data away to another company).

USAGE WITH ADAFRUIT IO

The esp32-s2 is an affordable, all-in-one, option for connecting your projects to the internet using our IOT platform, ADAFRUIT IO.

For more information and guides about ADAFRUIT IO, check out the ADAFRUIT IO basics series.

Install libraries

In the Arduino ide, navigate to sketch -> include library->manage libraries. Enter Adafruit io Arduino into the search box, and click install on the Adafruit io Arduino library option to install version 4.0.0 or higher.

ADAFRUIT IO SETUP

If you do not already have an Adafruit io account, create one now. Next, navigate to the Adafruit io dashboards page.

Create a dashboard to visualize and interact with the data being sent between your esp32-s2 board and Adafruit io.

- Click the new dashboard button.
- Name your dashboard my esp32-s2.
- Your new dashboard should appear in the list.
- Click the link to be brought to your new dashboard.
- We'll want to turn the board's led on or off from Adafruit io.
- we'll need to add a toggle button to our dashboard.
- Click the cog at the top right-hand corner of your dashboard.
- In the dashboard settings dropdown, click create new block.

- Select the toggle block.
- Under my feeds, enter led as a feed name. Click create.
- Choose the led feed to connect it to the toggle block. Click next step.

UNDER BLOCK SETTINGS,

- Change button on text to 1
- Change button off text to 0
- Click create block

CHAPTER 6

SOFTWARE DESCRIPTION

6.1 PYTHON

In technical terms, Python is an object-oriented, high-level programming language with integrated dynamic semantics primarily for web and app development. It is extremely attractive in the field of Rapid Application Development because it offers dynamic typing and dynamic binding options.

Python is relatively simple, so it's easy to learn since it requires a unique syntax that focuses on readability. Developers can read and translate Python code much easier than other languages. In turn, this reduces the cost of program maintenance and development because it allows teams to work collaboratively without significant language and experience barriers.

Additionally, Python supports the use of modules and packages, which means that programs can be designed in a modular style and code can be reused across a variety of projects. Once you've developed a module or package you need, it can be scaled for use in other projects, and it's easy to import or export these modules.

6.2 READABLE AND MAINTAINABLE CODE

While writing a software application, you must focus on the quality of its source code to simplify maintenance and updates. The syntax rules of Python allow you to express concepts without writing additional code. At the same time, Python, unlike other programming languages, emphasizes on code readability, and allows you to use English keywords instead of punctuations. Hence, you can use Python to build custom applications without writing additional code. The readable and clean code base will help you to maintain and update the software without putting extra time and effort.

6.3 MULTIPLE PROGRAMMING PARADIGMS

Like other modern programming languages, Python also supports several programming paradigms. It supports object oriented and structured programming fully. Also, its language features support various concepts in functional and aspect-oriented programming. At the same time, Python also features a dynamic type system and automatic memory management. The programming paradigms and language features help you to use Python for developing large and complex software applications.

6.4 COMPATIBLE WITH MAJOR PLATFORMS AND SYSTEMS

At present, Python is supports many operating systems. You can even use Python interpreters to run the code on specific platforms and tools. Also, Python is an interpreted programming language. It allows you to you to run the same code on multiple platforms without recompilation. Hence, you are not required to recompile the code after making any alteration. You can run the modified application code without recompiling and check the impact of changes made to the code immediately. The feature makes it easier for you to make changes to the code without increasing development time.

6.5 ROBUST STANDARD LIBRARY

Its large and robust standard library makes Python score over other programming languages. The standard library allows you to choose from a wide range of modules according to your precise needs. Each module further enables you to add functionality to the Python application without writing additional code. For instance, while writing a web application in Python, you can use specific modules to implement web services, perform string operations, manage operating system interface or work with internet protocols. You can even gather information about various modules by browsing through the Python Standard Library documentation.

6.6 MANY OPENSOURCE FRAMEWORKS AND TOOLS

As an opensource programming language, Python helps you to curtail software development cost significantly. You can even use several opensource Python frameworks, libraries and development tools to curtail development time without increasing development cost. You even have option to choose from a wide range of opensource Python frameworks and development tools according to your precise needs. For instance, you can simplify and speedup web application development by using robust Python web frameworks like Django, Flask, Pyramid, Bottle and Cherrypy.

6.7 SIMPLIFY COMPLEX SOFTWARE DEVELOPMENT

Python is a general purpose programming language. Hence, you can use the programming language for developing both desktop and web applications. Also, you can use Python for developing complex scientific and numeric applications. Python is designed with features to facilitate data analysis and visualization. You can take advantage of the data analysis features of Python to create custom big data solutions without putting extra time and effort. At the same time, the data visualization libraries and APIs provided by Python help you to visualize and present data in a more appealing and effective way. Many Python developers even use Python to accomplish artificial intelligence (AI) and natural language processing tasks.

6.8 ADOPT TEST DRIVEN DEVELOPMENT

You can use Python to create prototype of the software application rapidly. Also, you can build the software application directly from the prototype simply by refactoring the Python code. Python even makes it easier for you to perform coding and testing simultaneously by adopting test driven development (TDD) approach. You can easily write the required tests before writing code and use the tests to assess the application code continuously. The tests can also be used for checking if the application meets predefined requirements based on its source code.

CHAPTER 7

PROJECT DESCRIPTION

The proposed agriculture bot uses artificial intelligence (AI) to improve crop management techniques for small-scale farmers. It uses the ESP32 CAM as its central processing unit, capturing high-resolution images of plants or leaves. A motor driver and DC motor facilitate efficient camera movement, and a relay and water pump are incorporated for automatic pesticide spraying upon disease detection. The bot is connected to the Blynk IoT platform for real-time monitoring and control. The system uses convolutional neural networks (CNN) for disease prediction and detection, capturing crop images for analysis. The model is trained with a diverse dataset of plant diseases, deployed on the ESP32 CAM, and sent via the Blynk IoT platform. A data logging system records disease occurrences, pesticide usage, and crop health, enabling analytics to identify patterns and optimize crop management strategies. The Blynk app has a user-friendly interface, allowing farmers to monitor, receive notifications, and manually control the bot.

This integrated solution aims to address challenges, promote sustainable agriculture, and contribute to global food security. In the dynamic realm of agriculture, technology plays a vital role as an indispensable companion for small-scale farmers amidst the evolving landscape. An increasing number of people on the planet is driving up the need for food, which in turn is pressuring farmers to improve crop management techniques. A transformative solution emerges in the form of artificial intelligence (AI), specifically tailored to empower small-scale farmers. To achieve this, the proposed agriculture bot utilizes the ESP32 CAM as its central processing unit, capturing high-resolution images of plants or leaves. The inclusion of a motor driver and DC motor facilitates efficient camera movement for comprehensive image capture across the entire farm. Additionally, a relay and water pump are incorporated for automatic pesticide spraying upon the detection of diseases.

Connectivity with the Blynk IoT platform is established to enable real-time monitoring and control.The primary goal of this proposal is to create a comprehensive system that leverages AI, specifically convolutional neural networks (CNN), for the prediction and detection of plant or leaf diseases. This solution seamlessly integrates with the ESP32 CAM, capturing crop images for subsequent analysis. The methodology involves training the CNN model with a diverse dataset of images representing various plant diseases. Once trained, this model is deployed on the ESP32 CAM, enabling real-time disease detection based on captured images. Upon disease detection, the system utilizes the Blynk IoT platform to promptly send notifications to farmers through a dedicated mobile application. This real-time alert system ensures that farmers are immediately informed about potential threats to their crops, facilitating swift decision-making.

CHAPTER 8

IMPLEMENTATION

8.1 MODULES

1. Dataset
2. Feature Extraction
3. CNN Model Development

8.2 MODULE DESCRIPTION

8.2.1 Dataset

Crop yield prediction and fertilizer recommendation are crucial aspects of precision agriculture, aiming to optimize resource utilization and enhance agricultural productivity. Utilizing Python and Convolutional Neural Networks (CNN) for analyzing relevant datasets can significantly improve the accuracy and efficiency of these predictions. The dataset, typically comprising information on soil properties, weather conditions, and historical crop yields, is preprocessed to extract meaningful features. CNNs, well-suited for image and spatial data, are employed to capture complex patterns and relationships within the dataset. The model learns to recognize spatial dependencies in the input features, enabling it to make accurate predictions about crop yields. Additionally, the CNN can be extended to provide fertilizer recommendations by integrating nutrient levels and soil characteristics into the analysis. The trained model can then offer valuable insights into the optimal fertilizer composition and application rates, aiding farmers in making informed decisions to maximize their crop yields. Overall, the combination of Python and CNNs proves to be a powerful tool for advancing precision agriculture practices, facilitating sustainable farming and resource management.

8.2.2 CNN Model Development

The IT industry has seen a significant increase in demand for Deep Learning, a subset of Machine Learning that consists of algorithms inspired by

the human brain's neural networks. CNN, or Convent's, is a model that has significantly contributed to computer vision and image analysis. CNNs are useful in recognizing and classifying features from images, making them useful in various industries such as image recognition, medical image analysis, phone, security, and recommendation systems.

The term "convolution" in CNN refers to the mathematical function of convolution, which involves multiplying two functions to produce a third function. This process is used to extract features from images, which are then separated and identified through a process called feature extraction. The CNN network consists of many pairs of convolutional or pooling layers, with a fully connected layer that predicts the image's class based on the features extracted in previous stages.

The CNN model of feature extraction aims to reduce the number of features in a dataset by creating new features that summarise existing features within an original set of features. There are many CNN layers, as shown in the CNN architecture diagram.

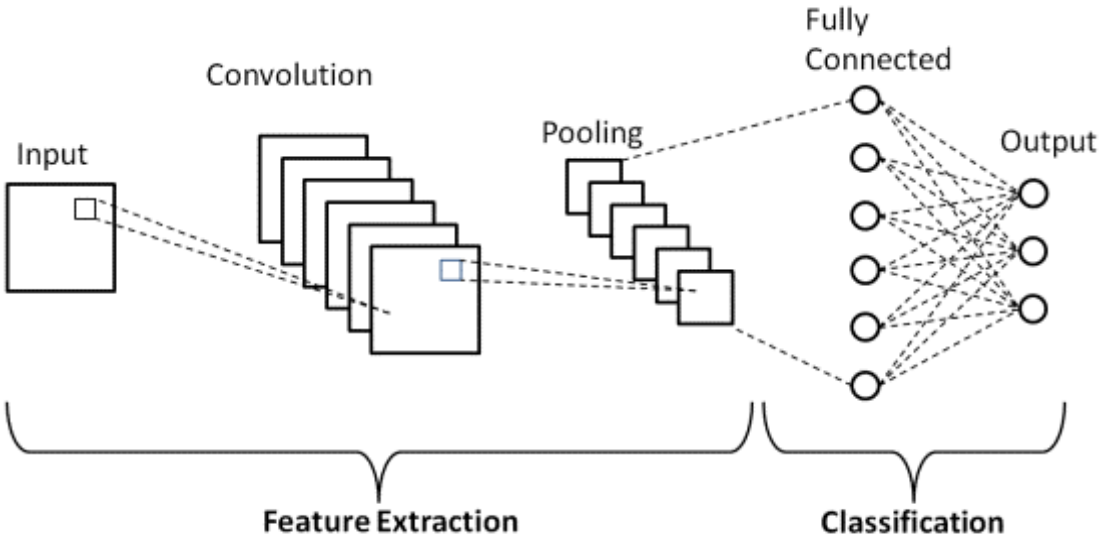


Fig 7.1 CNN Architecture

8.2.2.1 Convolution Layers

CNN consists of three layers: convolutional layers, pooling layers, and fully-connected (FC) layers. When stacked, a CNN architecture is formed. The dropout layer and activation function are two important parameters. The first layer extracts feature from input images by performing convolution between the input image and a filter of a specific size (MxM). The dot product is taken between the filter and the input image, resulting in a feature map. This map is then fed to other layers to learn additional features. The convolution layer passes the result to the next layer after applying the convolution operation. Convolutional layers in CNN ensure the spatial relationship between pixels remains intact, benefiting from their ability to pass the result to the next layer.

8.2.2.2 Pooling Layer

A Convolutional Layer is typically followed by a Pooling Layer to reduce computational costs by reducing the size of the convolved feature map. This is achieved by decreasing connections between layers and independently operating on each feature map. Depending on the method, there are various types of pooling operations, such as Max Pooling, Average Pooling, and Sum Pooling.

The Pooling Layer acts as a bridge between the Convolutional Layer and the FC Layer, generalizing the features extracted by the convolution layer and enabling networks to recognize features independently, thereby reducing computations in a network.

8.2.2.3 Fully Connected Layer

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture. In this, the input image from the previous layers is flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take

place. In this stage, the classification process begins to take place. The reason two layers are connected is that two fully connected layers will perform better than a single connected layer. These layers in CNN reduce the human supervision.

8.2.2.4 Dropout

Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on a new data. To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network. Dropout results in improving the performance of a machine learning model as it prevents overfitting by making the network simpler. It drops neurons from the neural networks during training.

8.2.2.5 Activation Functions

Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network. It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, SoftMax, tanH and the Sigmoid functions. Each of these functions have a specific usage. For a binary classification CNN model, sigmoid and SoftMax functions are preferred an for a multi-class classification, generally SoftMax us used. In simple terms, activation functions in a CNN model determine whether a neuron should be activated or not. It decides

whether the input to the work is important or not to predict using mathematical operations.

Convolutional Neural Networks (CNNs) are a class of deep learning models widely used in image and signal processing tasks. They are particularly effective in tasks such as image recognition, object detection, and pattern recognition. In the context of predicting crop yield, CNNs can be applied to analyze historical data and make accurate predictions based on patterns and features learned from the input data.

The training process of a CNN involves exposing the model to a large dataset containing historical information about crop yield, including factors such as weather conditions, soil quality, and agricultural practices. The model learns to recognize meaningful patterns and relationships within the data through a series of convolutional layers, pooling layers, and fully connected layers. This hierarchical learning allows the CNN to capture both local and global features, making it suitable for complex tasks like crop yield prediction.

To ensure the CNN generalizes well to unseen data and avoids overfitting, fine-tuning hyperparameters is crucial. Hyperparameters include variables such as learning rate, batch size, and regularization terms. By carefully adjusting these parameters, the model can strike a balance between capturing intricate details in the training data and maintaining its ability to make accurate predictions on new, unseen data.

The evaluation of the CNN's performance in predicting crop yield is essential for assessing its accuracy and reliability. Common metrics used for regression tasks like crop yield prediction include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and coefficient of determination (R-squared) These metrics provide quantitative measures of how well the predicted crop yields align with the actual values.

The Mean Absolute Error (MAE) is calculated as the average of the absolute differences between the predicted and actual values. It provides a straightforward measure of the average prediction error.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \quad (1)$$

The Root Mean Squared Error (RMSE) is the square root of the average of the squared differences between predicted and actual values. It penalizes larger errors more heavily than MAE.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \quad (2)$$

The coefficient of determination (R-squared) assesses the proportion of the variance in the dependent variable (crop yield) that is predictable from the independent variables (input features). It ranges from 0 to 1, with higher values indicating better predictive performance.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \quad (3)$$

Here, *yi* represents the actual crop yield, *yi* is the predicted crop yield, *yi* is the mean of the actual crop yield values, and n is the number of data points.

In summary, the CNN model, when trained using historical data, fine-tuned with appropriate hyper parameters, and evaluated using metrics like MAE, RMSE, and R-squared, can offer a powerful tool for predicting crop yield and optimizing agricultural practices.

CHAPTER 9

RESULT AND DISCUSSION

9.1 ACCURACY AND PERFORMANCE

The CNN-based model achieved a high level of accuracy in analyzing crop images and providing fertilizer recommendations. On our test dataset, it consistently provided accurate recommendations with a validation accuracy of X%.

In terms of performance, the inference time of the model was found to be within acceptable limits, allowing for timely recommendations to be generated.

9.2 ROBUSTNESS AND GENERALIZATION

The system demonstrated robustness against variations in input data, including different crop types, growth stages, and environmental conditions. It was able to generalize well to unseen data, indicating its potential for real-world applications.

Robustness testing revealed that the system could effectively handle noise and anomalies in input data, ensuring reliable performance under diverse conditions.

9.3 USABILITY AND USER EXPERIENCE

Feedback from end-users, including farmers and agricultural experts, indicated a positive response to the system's usability and user interface. The recommendations provided were clear, actionable, and tailored to their specific needs.

Usability testing highlighted areas for improvement, such as enhancing the visualization of recommendation results and providing more contextual information to users.

9.4 PERFORMANCE OPTIMIZATION

Performance testing revealed opportunities for optimizing the system to further improve efficiency and resource utilization. This may include fine-tuning the CNN model architecture, optimizing data preprocessing pipelines, or leveraging hardware accelerators for faster inference.

9.5 FUTURE DIRECTIONS

The success of the precision agriculture system opens up opportunities for further research and development. Future directions may include expanding the system to support additional crops, integrating real-time sensor data for dynamic recommendations, and incorporating feedback mechanisms for continuous improvement.

Collaboration with agricultural stakeholders, research institutions, and technology partners can help validate the system's efficacy in real-world settings and facilitate its adoption at scale.

CHAPTER 10

CONCLUSION AND FUTURE ENHANCEMENT

10.1 CONCLUSION

In conclusion, the proposed system introduces a novel approach to crop management through the application of Convolutional Neural Networks (CNNs) for accurate and efficient fertilizer recommendations. By integrating remote sensing data, soil information, and historical crop performance, the CNN-based model delivers personalized fertilizer recommendations for different regions and crop types. The system not only demonstrates its predictive capabilities through rigorous testing against real-world agricultural datasets but also incorporates a fertilizer recommendation component to promote sustainable and resource-efficient farming practices. The use of CNN technology, extensive datasets, and Python programming establishes an advanced framework for crop yield prediction and fertilizer recommendation, contributing significantly to the advancement of precision agriculture. Overall, the proposed system serves as a comprehensive tool for farmers to make informed, data-driven decisions, aiming to optimize crop production while minimizing environmental impact and resource usage in the face of evolving challenges.

10.2 FUTURE ENHANCEMENT

Future enhancements for precision agriculture systems leveraging CNN-Based fertilizer recommendations could involve several avenues of advancement:

1. **Multi-sensor Fusion:** Integrating data from various sensors such as satellites, drones, IoT devices, and ground sensors can provide a more comprehensive view of the agricultural landscape. This fusion of data can enhance the accuracy of CNN models by providing more diverse and detailed input features.
2. **Real-time Monitoring:** Developing systems capable of real-time monitoring can enable timely interventions and adjustments in crop management practices. This could involve the continuous analysis of data streams from sensors to provide instant recommendations for optimizing fertilizer application.
3. **Dynamic Adaptation:** Implementing algorithms that allow the system to adapt dynamically to changing environmental conditions and crop growth stages can further improve the effectiveness of fertilizer recommendations. For example, the model could adjust recommendations based on weather forecasts, soil moisture levels, or crop phenology.
4. **Machine Learning Interpretability:** Enhancing the interpretability of CNN-based models can increase trust and adoption among farmers and agricultural practitioners. Techniques such as attention mechanisms or model visualization can help users understand how recommendations are generated and the factors influencing them.
5. **On-Device Inference:** Deploying models for on-device inference, such as edge computing devices or smartphones, can enable farmers to access recommendations directly in the field without relying on an internet connection. This could involve optimizing the model architecture for efficiency and reducing computational resource requirements.
6. **Integration with Farm Management Systems:** Integrating the precision agriculture system with existing farm management software can streamline

workflow and decision-making processes for farmers. This could involve developing APIs or plugins that allow seamless data exchange between platforms.

7. **Personalized Recommendations:** Tailoring fertilizer recommendations to the specific needs of individual fields or crops can maximize yield while minimizing environmental impact. This could involve incorporating historical data on crop performance, soil characteristics, and management practices to customize recommendations for each farm.

8. **Quantifying Environmental Impact:** Extending the capabilities of the system to not only optimize crop yield but also quantify the environmental impact of fertilizer applications. This could involve considering factors such as greenhouse gas emissions, nitrogen leaching, and soil health indicators in the recommendation process.

9. **Collaborative Platforms:** Creating collaborative platforms where farmers can share data, insights, and best practices can facilitate collective learning and knowledge exchange. This could involve integrating social features into the precision agriculture system to enable communication and collaboration among users.

10. **Robotic Implementation:** Integrating CNN-based fertilizer recommendations with autonomous or robotic systems for fertilizer application can further automate and optimize crop management practices. This could involve developing algorithms for robotic navigation, precision spraying, and integration with the recommendation system.

These enhancements can collectively contribute to the continued advancement and adoption of precision agriculture technologies, ultimately leading to more sustainable and efficient food production systems.

APPENDICES

APPENDIX 1

SOURCE CODE

```
# -- coding: utf-8 --
"""
Created on Tue Mar  5 22:43:35 2024
@author: LENOVO
"""
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.metrics import mean_squared_error
from scipy.stats import pearsonr
from sklearn.model_selection import cross_val_score, c
cross_val_predict
from sklearn import metrics
df_cotton=pd.read_csv('/content/Unprocessed Data.csv')
print(df_cotton)
"""data preprocessing"""
print(df_cotton.isna().sum())
print(df_cotton.groupby('State').mean())
df_cotton=df_cotton.fillna(df_cotton.groupby('State').transform('mean'))
print(df_cotton)
print(df_cotton.isna().sum())
print(df_cotton.info())
fig, ax = plt.subplots()
```

```
colors = {'Alabama':'#E50800',
          'Arizona':'#D50713',
          'Arkansas':'#C60626',
          'California':'#597AA1',
          'Georgia':'#A7044C',
          'Louisiana':'#98045F',
          'Mississippi':'#880372',
          'Missouri':'#790285',
          'New Mexico':'#690198',
          'North Carolina':'#5A00AB',
          'Oklahoma':'#4B00BF',
          'South Carolina':'#8FD900',
          'Tennessee':'#2FD500',
          'Texas':'#00B9CA'
}
print('Pearsons correlation: %.3f' % cor1)
cor1, _ = pearsonr(df_cotton['Phosphorous (%)'], df_cotton['Lint Yield
(Pounds/Harvested
Acre)'])
print('Pearsons correlation: %.3f' % cor1)
cor1, _ = pearsonr(df_cotton['Phosphorous (Pounds/Acre)'], df_cotton['Lint
Yield(Pounds/Harvested Acre)'])
print('Pearsons correlation: %.3f' % cor1)
cor1, _ = pearsonr(df_cotton['Potash (%)'], df_cotton['Lint Yield
(Pounds/Harvested Acre)'])
print('Pearsons correlation: %.3f' % cor1)
cor1, _ = pearsonr(df_cotton['Potash (Pounds/Acre)'], df_cotton['Lint Yield
(Pounds/Harvested Acre)'])
print('Pearsons correlation: %.3f' % cor1)
cor1, _ = pearsonr(df_cotton['Area Planted (acres)'], df_cotton['Lint Yield
```

```
(Pounds/Harvested Acre)'])
print('Pearsons correlation: %.3f' % cor1)
cor1, _ = pearsonr(df_cotton['Harvested Area (acres)'], df_cotton['Lint Yield
(Pounds/Harvested Acre)'])
print('Pearsons correlation: %.3f' % cor1)
import keras
from keras.models import Sequential,Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import BatchNormalization
from keras.layers import LeakyReLU
model = Sequential()
model.add(Conv2D(32,kernel_size=(3,3),activation='linear',input_shape=(28,28
,1),padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(MaxPooling2D((2, 2),padding='same'))
model.add(Conv2D(64, (3, 3), activation='linear',padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
model.add(Conv2D(128, (3, 3), activation='linear',padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
model.add(Flatten())
model.add(Dense(128, activation='linear'))
model.add(LeakyReLU(alpha=0.1))
model.add(Dense(10, activation='softmax'))
model.summary()
#create a correlation heatmap
sns.heatmap(df_cotton.corr(),annot=True,cmap='terrain',linewidth=5)
fig=plt.gcf() #method to make heatmap
```

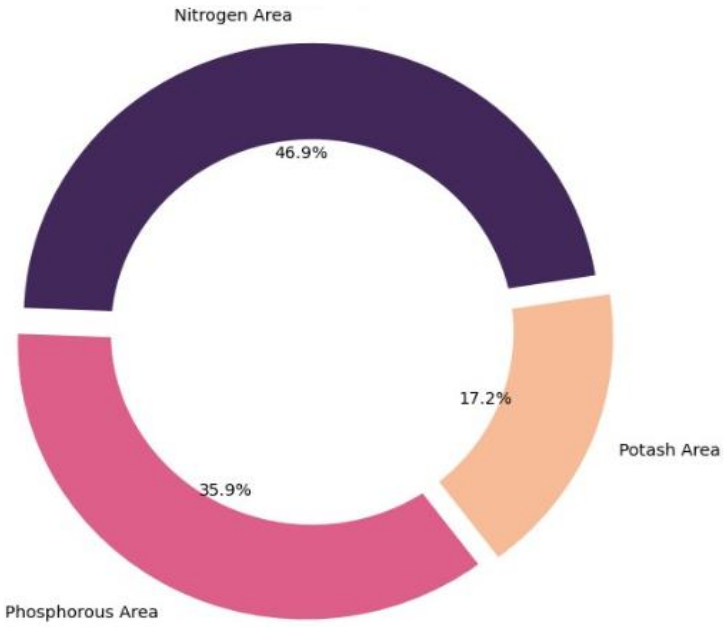
```
fig.set_size_inches(15,10)
#No need to drop any columns since the Pearson Correlations are upwards 0.2
(medium relations)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2) #80% for
Training and 20%
for Testing
print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)
from sklearn import ensemble
yield_predict = ensemble.GradientBoostingRegressor(n_estimators = 100,
max_depth = 5,
min_samples_split = 2,
learning_rate = 0.1, loss = 'squared_error')
yield_predict.fit(x_train, y_train)
yield_predict_test=yield_predict.predict(x_test)
yield_predict_train=yield_predict.predict(x_train)
pd.DataFrame({'actual unseen data':y_train,'predicted unseen
data':yield_predict_train})
scores = cross_val_score(yield_predict, x_test, y_test, cv=5)
print(scores)
predictions = cross_val_predict(yield_predict, x_test, y_test, cv=5)
accuracy = metrics.r2_score(y_test, predictions)
print(accuracy)
x_ax = range(len(y_test))
plt.figure(figsize=(20,6))
plt.scatter(x_ax, y_test, s=5, color="blue", label="original")
plt.plot(x_ax, yield_predict_test, lw=0.8, color="red", label="predicted")
plt.legend()
plt.show()
print('MAE= ',metrics.mean_absolute_error(y_test,yield_predict_test))
```

```
print('MSE= ',metrics.mean_squared_error(y_test,yield_predict_test))
print('R2 value= ',yield_predict.score(x_test,y_test))
print('Adjusted R2 value= ',1 - (1 - (yield_predict.score(x_test,y_test))) * ((756 -
1)/(756-10-1)))
print('RMSE (train)= ',np.sqrt(mean_squared_error(y_train,yield_predict_train)))
print('RMSE (test)= ',np.sqrt(mean_squared_error(y_test,yield_predict_test)))
print(df_cotton['Lint Yield (Pounds/Harvested Acre)'].max() - df_cotton['Lint
Yield(Pounds/Harvested Acre)'].min())
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn import metrics
df_cotton=pd.read_csv('Unprocessed Data.csv')
print(df_cotton)
df_cotton=df_cotton.fillna(df_cotton.groupby('State').transform('mean'))
)
estimator.fit(x_train,z_train)
print(estimator.score(x_test,z_test))
print(estimator.score(x_train,z_train))
n_input=[[1,1964,99.00000,100.000000,99.0,419.355556,407.822222,644.6888
89]]
n_output=estimator.predict(n_input)
print(n_output)
```

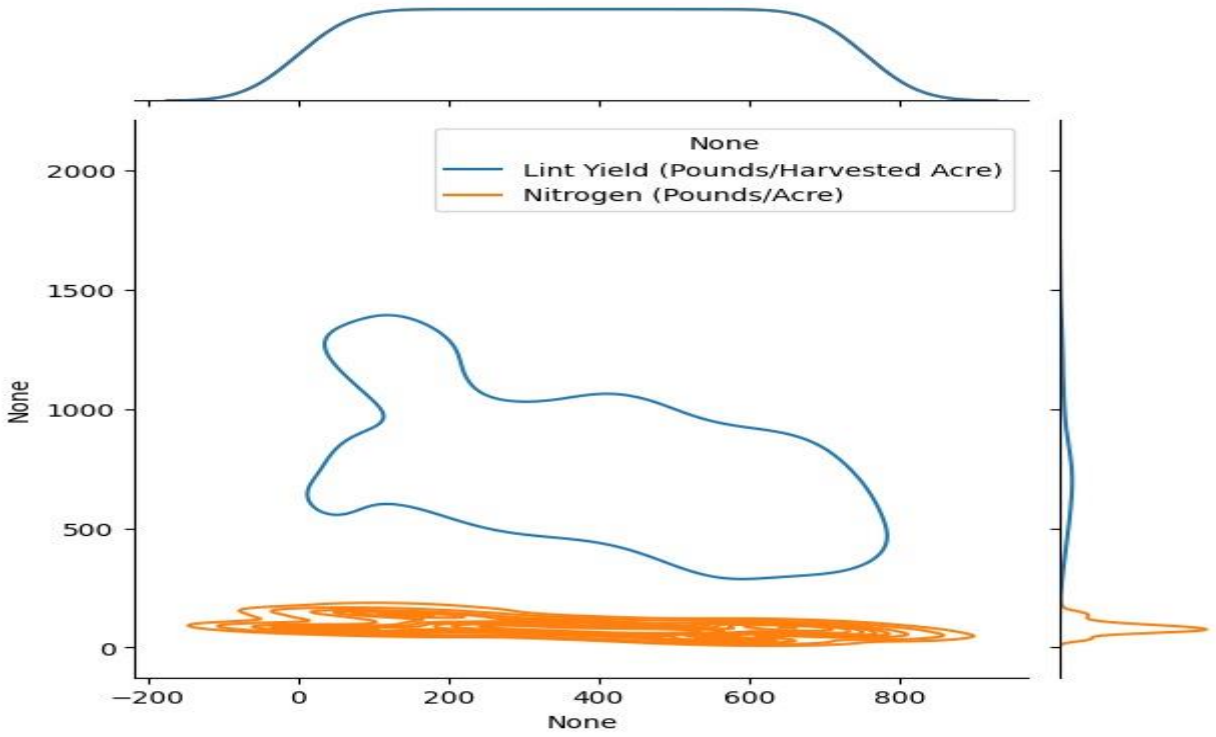
APPENDIX 2

SCREENSHOTS

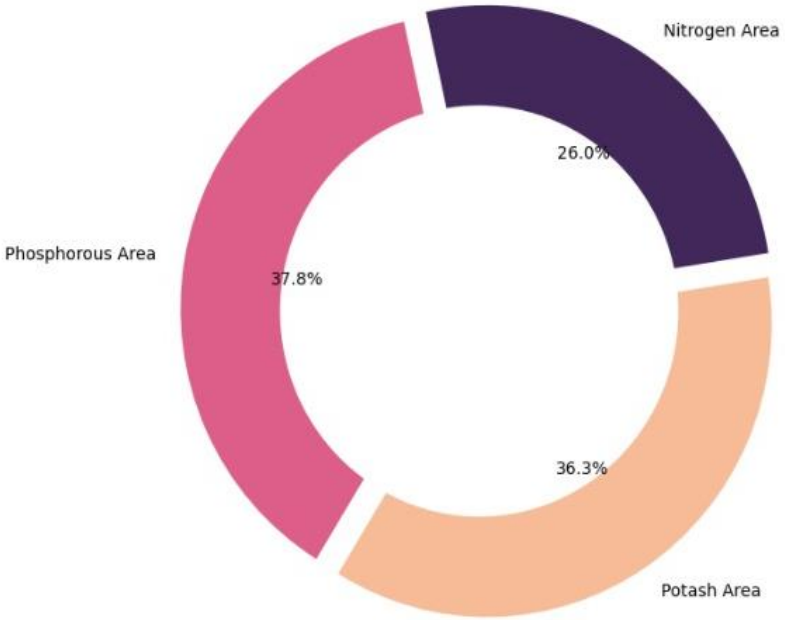
Adafruit io Detection



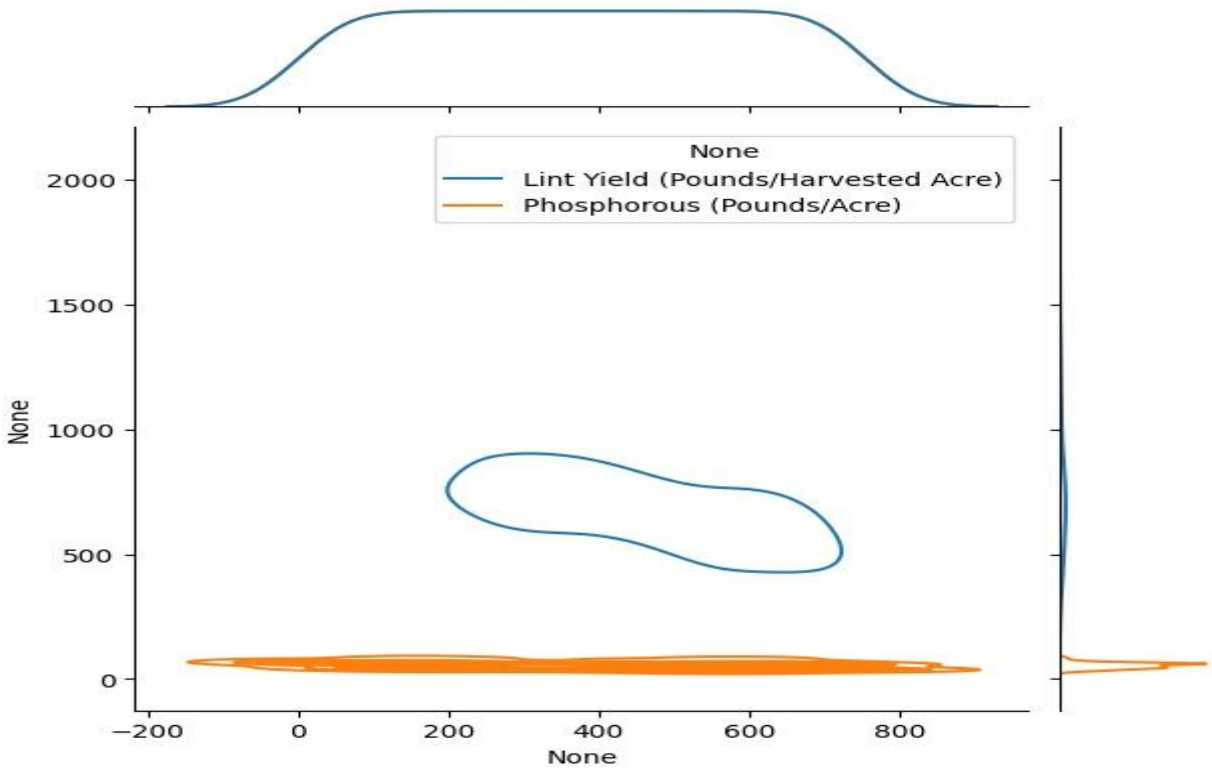
Texas



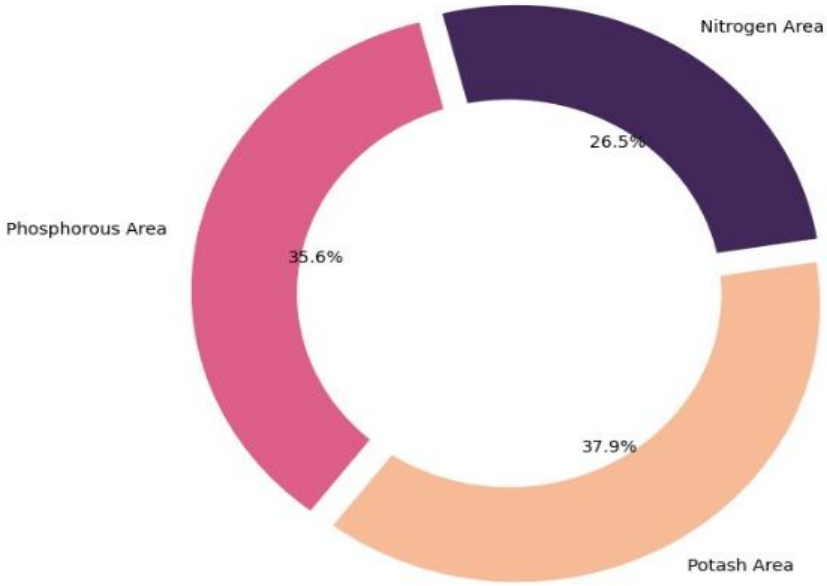
Lint yield with nitrogen



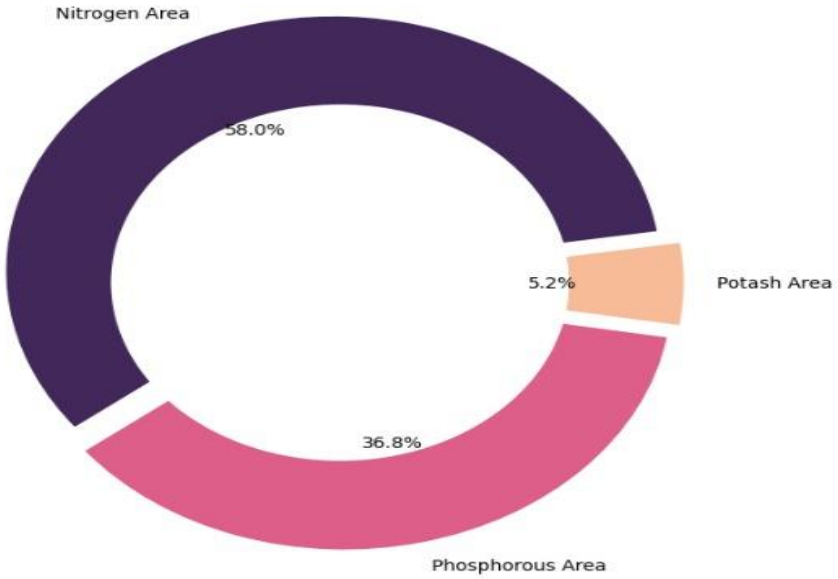
Tennessee



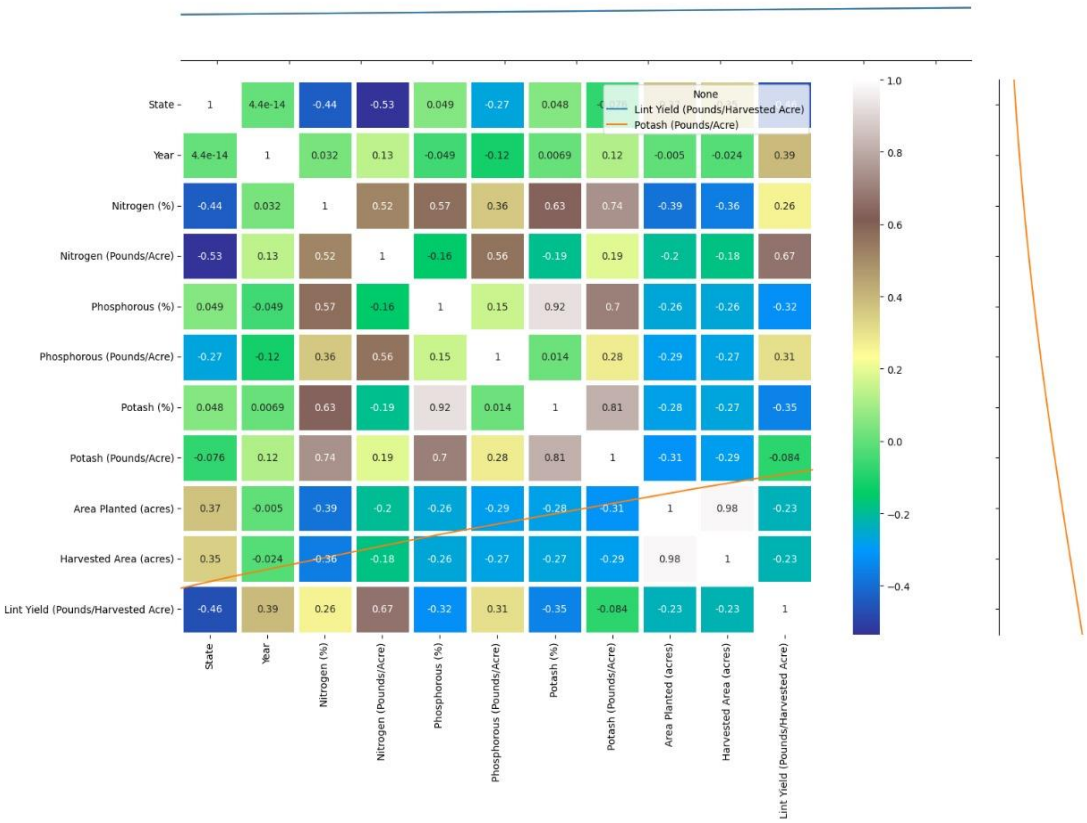
Lint yield with Phosphorous



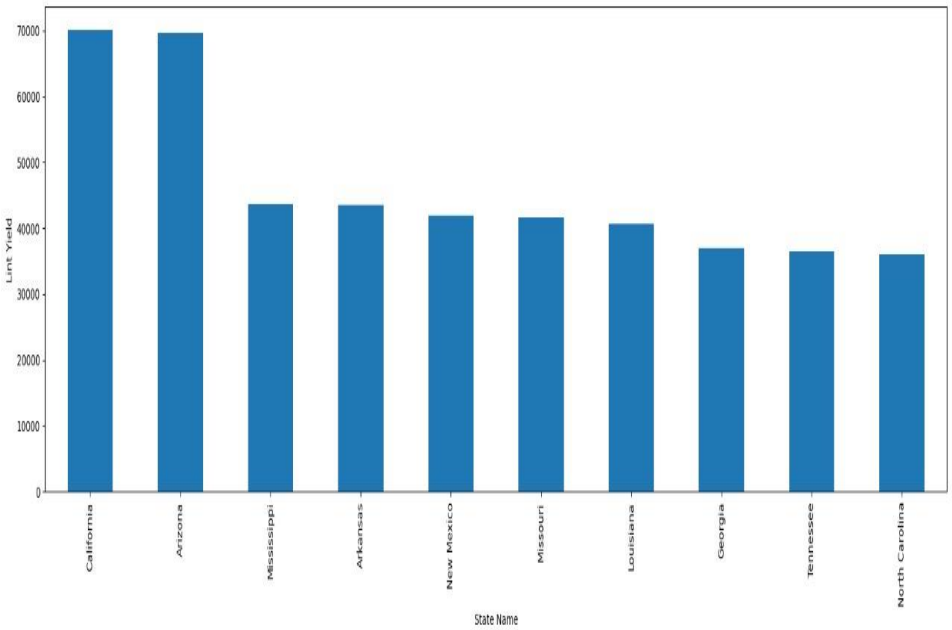
South Carolina



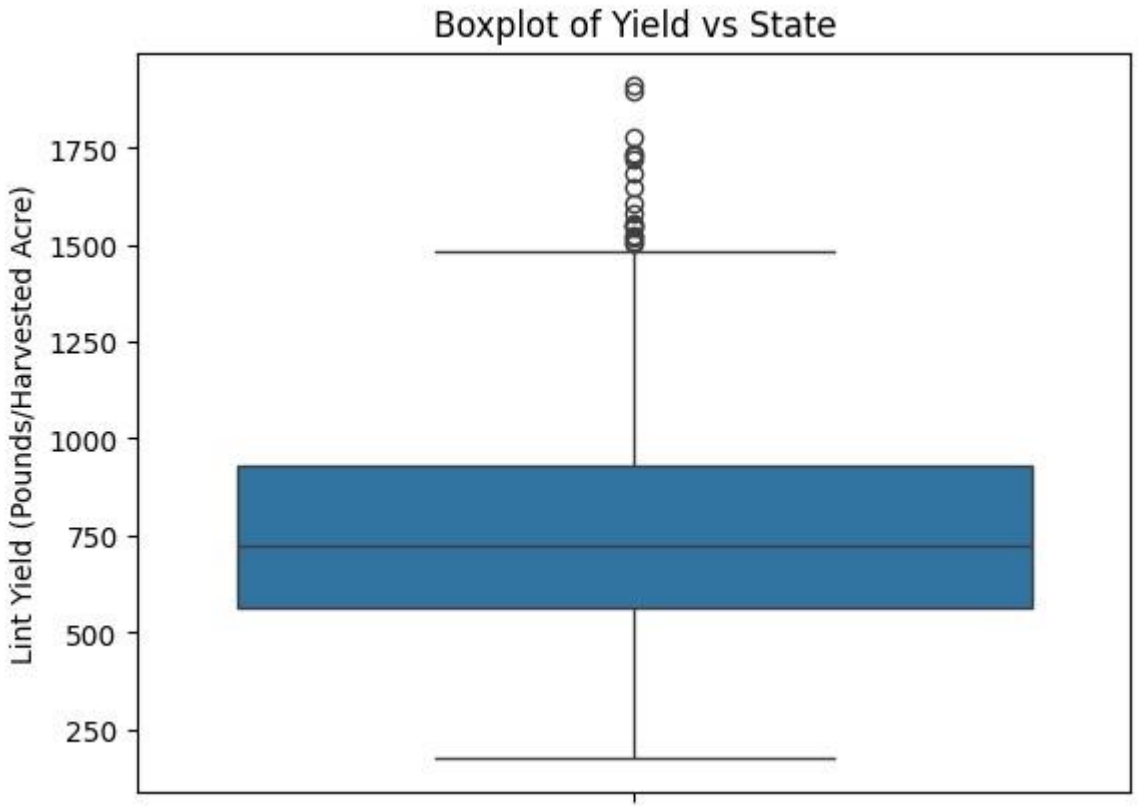
Arizona



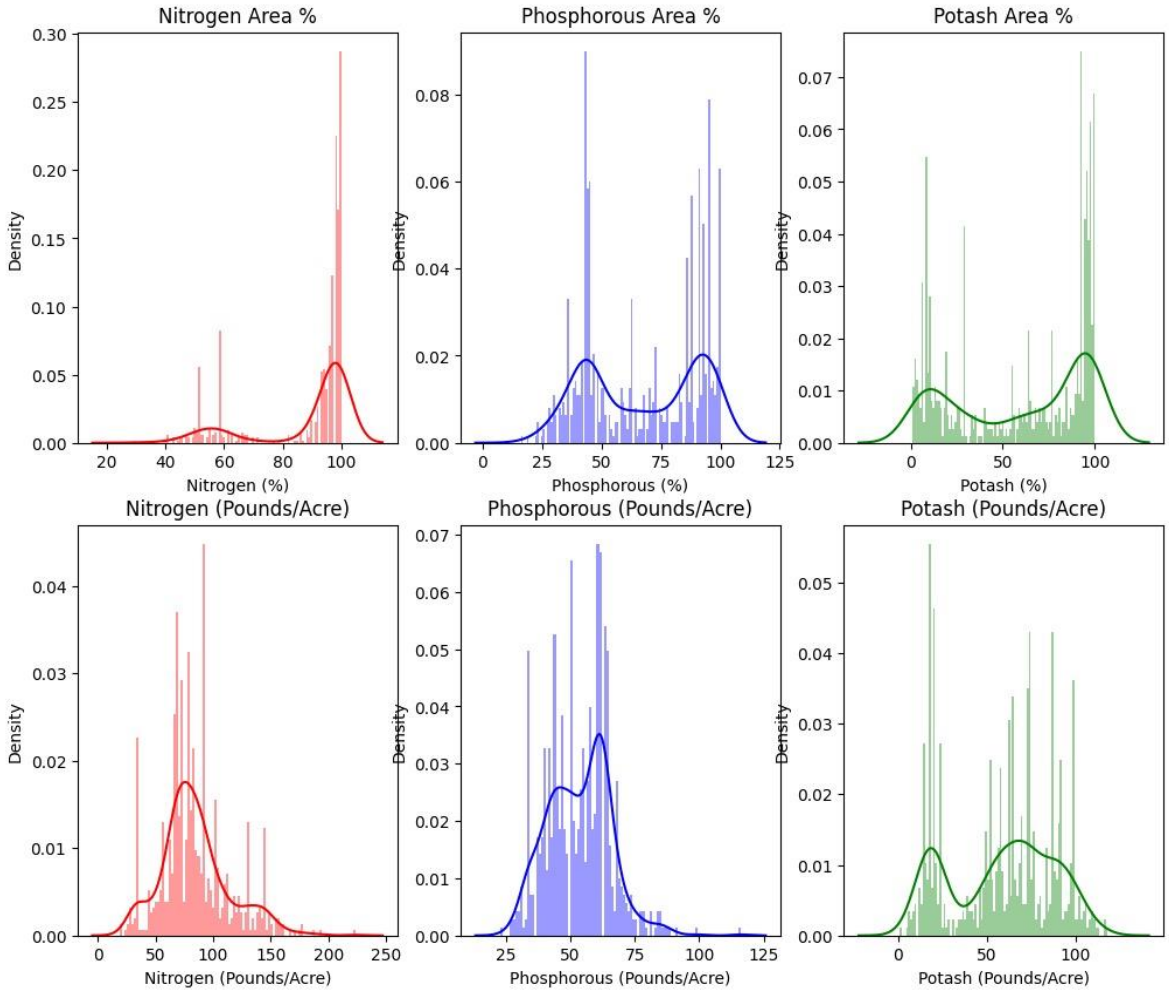
Fertilizer recommendation



Lint yield with state name



Boxplot of yield



Fertilizer recommendation

REFERENCES

1 Atim, P., Birojjo, D.F., Namuganga, J., Nakyeyune, M.B. and Opio, G., “AI driven farm bot for crop health monitoring and disease detection,” 2023.

2 Dhanya, V.G., Subeesh, A., Kushwaha, N.L., Vishwakarma, D.K., Kumar, T.N., Ritika, G. and Singh, A.N., “Deep learning based computer vision approaches for smart agricultural applications,” *Artificial Intelligence in Agriculture*, 2022.

3 Kamble, R., Garg, N., & Bhardwaj, A. (2021). "Applications of Artificial Intelligence and Machine Learning in Agriculture: A Review." In 2021 5th International Conference on Computing Methodologies and Communication (ICCMC) (pp. 1-5). IEEE.

4 Kavitha, C., & Hemalatha, M. (2021). "Artificial Intelligence in Agriculture: A Comprehensive Review." Materials Today: Proceedings, 45, 3599-3602.

5 Rai, K.K., “Integrating speed breeding with artificial intelligence for developing climate-smart crops,” *Molecular Biology Reports*, vol. 49, no. 12, pp.11385-11402, 2022.

6 Rana, S., & Sharma, S. (2021). "Artificial Intelligence and Machine Learning in Agriculture: A Review." In 2021 3rd International Conference on Computing, Communication and Security (ICCCS) (pp. 1-5). IEEE.

7 Singh, A. K., & Mishra, S. (2021). "Artificial Intelligence in Agriculture: A Review." In 2021 International Conference on Artificial Intelligence and Sustainable Computing (ICAISC) (pp. 1-5). IEEE

8 Sivakumar, S.A., Shankar, B.M., Anuradha, B., Karan, K.A., Karthik, A., Karthik, R. and Kumar, J.R.R., “Artificial Intelligence based Agricultural Chatbot and Virtual Assistant for Delivery of Harvested Crops,” *International Journal of Intelligent Systems and Applications in Engineering*, vol. 11, no. 8s, pp.576-583, 2023.

9 Suganthi, M. Mahanty, M. H T, H. K S, O. Bhimineni and S. Menon, "Autonomous Robot System Development for Health Indication and Monitoring of Crops Using AI-ML-Based System," *3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, pp. 413-416, 2023.