Development of a Regionally Customized Python-Based Fertilizer Recommendation System for Small-Scale Farmers in Kerala

Akhilesh Muralidharan[1], Jais Binoy[1]

[1]Research Associates at International Centre for Technological Innovations, Kerala, India

ABSTRACT

This research paper introduces a specialized Python code tailored to the unique agricultural landscape of Kerala, India, where small-scale and diverse crop farming is prevalent. With most farmers being small and marginal, the code provides personalized fertilizer recommendations for different crop types and farm sizes based on Kerala Agricultural University's Package of practices. By harnessing data-driven insights, it empowers farmers with regionally customized fertilizer recommendation to enhance agricultural productivity, resource efficiency, and sustainable farming practices. The code's current features include efficient nutrient level interpretation, fertilizer calculation and user-friendly interaction. Future enhancements will expand its capabilities, making it a comprehensive decision support system for Kerala's agricultural community, bridging the gap between traditional farming practices and modern technology.

Keywords:Crops,Fertilizer Calculator, Fertilizer Recommendation, Pulses, Python, Small-Scale Farmers.

INTRODUCTION

Mineral fertilizers hold an indispensable function in addressing the increasing food demand while safeguarding the preservation of long-term soil fertility. [1,2] Most of the farmers in Kerala are small and marginal farmers.[3] Many small-scale agricultural operations primarily target specialized markets, cultivating a diverse array of crops and employing various crop management techniques. This diversity necessitates tailored fertilizer applications, in contrast to large-scale monoculture farming, and poses a challenge for small farms in terms of procuring cost-effective bulk fertilizer blends or liquid fertilizers due to their limited crop acreage, often divided among multiple crops. Consequently, small farmers typically incur higher expenses through the acquisition of multiple fertilizer batches tailored to distinct cropping systems, increasing the risk of both over-application and under-application in meeting crop fertility requirements.[4] In developing nations, mineral-fertilizers have proven to enhance crop productivity among small-scale farmers when used in the correct amounts and at the appropriate times and locations, alongside suitable agronomic practices.[5] Throughout history, economically advanced countries have traditionally benefited from enhanced accessibility to a multitude of fertilizer advisory resources, encompassing written documentation and computer-based platforms, among others.[6].While there are few digital extension services such as the IFFCO KISAN app that provides fertilizer recommendation for farmers. [7,8] fewer one's are available, that considers the fertilizer recommendations developed for the state of Kerala.

Objective

- ☞ To create a specialized Python code tailored to the agricultural landscape of the Kerala. This code will serve as a valuable tool, enabling farmers to obtain individualized fertilizer recommendations based on their specific crop types and the size of their farming area.

- ☞ To augment the decision-making capacity of farmers by providing them with data-driven and regionally customized fertilizer suggestions through the Python code. By offering recommendations based on area-specific and crop-tailored parameters, the code seeks to contribute to increased agricultural productivity, resource efficiency, and environmentally responsible farming practices in Kerala.

METHODOLOGY

Python: Created a code to calculate fertilizer needs based on soil testing results provided by user. Soil rating chart used for the state of Kerala was incorporated into the code, based on data from soil fertility handbook [9].

Nutrient	Low	Medium	High
N	**< 280 kg/ha**	**280-560 kg/ha**	**>560 kg/ha**
P	**<12 kg/ha**	**12- 24 kg/ha**	**>24 kg/ha**
K	**<110 kg/ha**	**110-280 kg/ha**	**>280 kg/ha**

Fertilizer recommendation for important pulses in Kerala was incorporated into the code, the recommendations are from package of practices of Kerala agricultural University [10].

Crop	Fertilizer recommendation (kg/ha)
1. Black gram	20:30:30
2. Cow pea	20:30:10
3. French bean	30:40:60
4. Green gram	20:30:30
5. Green pea	30:40:60
6. Horse gram	0:25:0
7. Red gram	40:80:0
8. Soya bean	20:30:10

In soil test-based fertilizer recommendation approach, fertilizers are suggested based on the soil testing value and fertilizer recommendations given for a crop in a region. In India, generally, nitrogen is given full dose as per the recommendation due to low status of C in soils. But in case of P and K, full dose of fertilizers is given in case of medium status and 25% less is given for high status and 25% more is given for low status respectively, because they are highly variable.[11].

Kg of fertilizer to be applied per ha can be calculated by the equation [11].

(Recommended dose/ % of nutrient in fertilizer) *100

LIMITATIONS

- Here only straight fertilizers like Urea, SSP and MOP are considered.
- In this code, the focus is exclusively directed toward the provision of fertilizer recommendations pertaining to pulse crops.

Development of Fertilizer Calculator based on soil test results and crop type

A. Python Code:

```
# START

# Define a dictionary containing information about various crops, including their nutrient requirements.

crops = {

    "blackgram": {"Name": "Black Gram", "N": 20, "P": 30, "K": 30},

    "cowpea": {"Name": "Cow Pea", "N": 20, "P": 30, "K": 10},

    "frenchbean": {"Name": "French Bean", "N": 30, "P": 40, "K": 60},

    "greengram": {"Name": "Green Gram", "N": 20, "P": 30, "K": 30},

    "greenpea": {"Name": "Green Pea", "N": 30, "P": 40, "K": 60},

    "horsegram": {"Name": "Horse Gram", "N": 0, "P": 25, "K": 0},

    "redgram": {"Name": "Red Gram", "N": 40, "P": 80, "K": 0},

    "soyabean": {"Name": "Soya Bean", "N": 20, "P": 30, "K": 10},

# Get user input for soil test values for Nitrogen (N), Phosphorus (P), and Potassium (K).

n = float(input("Enter the Nitrogen value found in soil test: "))
```

```
p = float(input("Enter the Phosphorus value found in soil test: "))
k = float(input("Enter the Potassium value found in soil test: "))
# Determine the Nitrogen (N) level in the soil based on the input values and provide an adjustment factor.
if n <= 280:
    n_level = "Low"
    # n_adjustment = 1.25   # Increase P by 25%
elif n > 280 and n < 560:
    n_level = "Medium"
  # n_adjustment = 1.0   # Keep P as is
else:
    n_level = "High"
    # n_adjustment = 0.75   # Decrease P by 25%
# Determine the Phosphorus (P) level in the soil based on the input values and provide an adjustment factor.
if p <= 12:
    p_level = "Low"
    p_adjustment = 1.25  # Increase P by 25%
elif p > 12 and p < 24:
    p_level = "Medium"
    p_adjustment = 1.0  # Keep P as is
else:
    p_level = "High"
    p_adjustment = 0.75  # Decrease P by 25%
# Determine the Potassium (K) level in the soil based on the input values and provide an adjustment factor.
if k <= 110:
    k_level = "Low"
    k_adjustment = 1.25  # Increase K by 25%
elif k > 110 and k < 280:
    k_level = "Medium"
    k_adjustment = 1.0  # Keep K as is
else:
    k_level = "High"
    k_adjustment = 0.75  # Decrease K by 25%
# Print the determined nutrient levels.
print(f"Your Nitrogen level is {n_level}, Your Phosphorus level is {p_level},
and Potassium level is {k_level}")
```

```
# Ask the user if they want to continue for further recommendations.
response = input("Do you wish to continue for further recommendation? (y/n): ")
if response.lower() == "n":
    print("Thank You.!")
else:
    while True:
        # Get the user's selected crop and convert it to lowercase and remove spaces.
        crop = input("What is your Crop: ").strip().replace(" ", "").lower()
        # Check if the selected crop is in the dictionary.
        if crop in crops:
            selected_crop = crops[crop]
            adjusted_n = selected_crop['N']
            adjusted_p = selected_crop['P'] * p_adjustment
            adjusted_k = selected_crop['K'] * k_adjustment
# Ask the user for the area unit (cent or acre) and calculate hectares.
            area_unit = input("Is your area cultivation in 'cent' or 'acre'? 
").strip().lower()
            if area_unit == 'cent':
                acres = float(input("Enter your total area of cultivation in 
cents: "))
                hectares = acres * 0.004047
            elifarea_unit == 'acre':
                acres = float(input("Enter your total area of cultivation in 
acres: "))
                hectares = acres * 0.404686
            else:
                print("Invalid area unit. Please Enter 'cent' or 'acre'.")
                continue
            # Calculate the adjusted nutrient values per hectare.
            adjusted_n_acres = adjusted_n * hectares
            adjusted_p_acres = adjusted_p * hectares
            adjusted_k_acres = adjusted_k * hectares
      # Print the nutrient recommendations per hectare.
print(f"Nutrient Proportion recommendation for {acres} {area_unit}s of 
{selected_crop['Name']} is Nitrogen: {adjusted_n_acres} Kg, Phosphorus: 
{adjusted_p_acres} Kg, Potassium: {adjusted_k_acres} Kg")
        # Calculate and print fertilizer recommendations per hectare.
            urea = (adjusted_n_acres / 46) * 100
            ssp = (adjusted_p_acres / 16) * 100

            mop = (adjusted_k_acres / 60) * 100

            print(f"Fertilizer recommendation for {acres} {area_unit}s of
{selected_crop['Name']} is Urea: {urea} Kg, SSP: {ssp} Kg, MOP: {mop} Kg")

            # Exit the loop if the crop is valid.

            break

        else:

            print("Invalid crop name. Please Enter a valid crop.")

# END
```

B. Code Explanation:

The Python code is designed to assist users in analyzing the nutrient levels of their soil based on the results of a soil test and provides area-specific fertilizer calculation for different pulses. Let's break down the code into sections and explain each part in detail.

The code begins by defining a dictionary named `crops`, which contains information about various crops, including their names, and nutrient requirements for Nitrogen (N), Phosphorus (P), and Potassium (K). This dictionary serves as a reference for crop-specific recommendations.

Next, the code prompts the user to input the soil test results for Nitrogen (N), Phosphorus (P), and Potassium (K). The entered values are converted to floating-point numbers for further calculations.

The code then categorizes the Nitrogen (N), Phosphorus (P), and Potassium (K) levels in the soil based on the input values. Depending on these levels, it sets categories such as "Low," "Medium," or "High." In the case of Nitrogen, it also mentions an adjustment factor in the comments, which is not used in the code. These adjustment factors suggest whether to increase, keep, or decrease Phosphorus (P) levels.

Similarly, the code categorizes Phosphorus (P) and Potassium (K) levels and provides adjustment factors for each based on the categories. These adjustment factors determine whether to adjust the Phosphorus and Potassium levels for the selected crop.

The determined nutrient levels (Nitrogen, Phosphorus, and Potassium) are printed to the user, giving them an overview of the soil's condition.

The code then asks the user if they wish to continue for further recommendations. If the user declines by entering 'n' or 'N', the program terminates with a "Thank You!" message. If the user wants to continue, the program proceeds to provide crop-specific recommendations.

In the crop-specific recommendation section, the code enters a while loop, prompting the user to input the name of the crop they are interested in. The input is converted to lowercase, stripped of leading and trailing spaces, and stored in the variable `crop`.

The code checks if the entered crop is in the `crops` dictionary. If the crop is found, it retrieves the crop's nutrient requirements for Nitrogen, Phosphorus, and Potassium from the dictionary. It also adjusts the Phosphorus and Potassium values based on the adjustment factors determined earlier.

Next, the code asks the user to specify the area unit for cultivation, either 'cent' or 'acre,' and then input the total area of cultivation. This allows the program to calculate the area in hectares.

The code then proceeds to calculate the required proportion of fertilizers based on the selected crop's nutrient requirements and the adjusted nutrient levels in the soil. It provides the user with specific recommendations on how much Nitrogen (N), Phosphorus (P), and Potassium (K) fertilizers they should apply per hectare for their chosen crop.

Finally, the loop continues until the user decides to exit, and it can repeat the crop-specific recommendation process for different crops as many times as the user desires.

In summary, this code assists users in analyzing their soil's nutrient levels, offers recommendations for adjusting those levels, and provides crop-specific fertilizer recommendations based on user input.

C. Current features

- **Data-Driven Nutrient levels interpretation:** The code enables data-driven analysis of soil nutrient levels, providing insights into Nitrogen (N), Phosphorus (P), and Potassium (K) levels, which serves as a foundation for informed fertilizer recommendations.

- **Crop-Specific Recommendations:** It offers crop-specific fertilizer recommendations by referencing a dictionary of crop nutrient requirements made by incorporating fertilizer recommendations of the Kerala Agricultural University, ensuring tailored advice for optimizing pulse crop yields.

- **User-Friendly Interaction:** The code provides a user-friendly interface, guiding users through the process of inputting soil test results, selecting crops, and calculating area units, making it accessible to a broad range of agricultural practitioners.

- **Efficiency and Automation:** By automating the calculation process, the code streamlines the analysis and recommendation process, saving users time and effort in determining the appropriate fertilizer application rates.

- **Iterative Capabilities:** Users can easily obtain recommendations for multiple crops in a single session, allowing for efficient analysis and planning of nutrient management strategies for diverse pulse varieties, contributing to sustainable agriculture.

D. Future scope of development

- **User Interface Development:** Develop a user-friendly graphical interface to make the code more accessible to a wider range of users, including those with limited programming experience. A GUI can simplify the data input process and provide visually intuitive recommendations.

- **Expanded Crop Database:** Expand the crop database to include a broader range of crops commonly grown in the Kerala region. This would make the code more comprehensive and relevant to a larger group of farmers and agricultural practitioners

.

- **Soil Amendments Recommendations:** Extend the code's capabilities to recommend application rates of specific soil amendments (e.g., lime, gypsum, organic matter) in addition to traditional fertilizers to address soil pH and nutrient balance issues effectively.

- **Carbon Footprint Assessment:** Develop an environmental impact assessment module that calculates the carbon footprint of fertilizer use and provides recommendations for reducing the environmental impact of agriculture.

- **Incorporation of Micro-Nutrient Recommendations:** Expand the code to incorporate recommendations for essential micro-nutrients, such as iron, zinc, copper, and boron. Utilize authoritative sources to determine the specific micro-nutrient requirements for various crops in the Kerala region. This enhancement will provide a more comprehensive and precise nutrient management system, addressing both macro and micro-nutrient needs of the crops, leading to improved soil and crop health.

- **Mixed Fertilizer Recommendations:** Enhance the code to provide recommendations for mixed or blended fertilizers that combine multiple nutrient sources. Allow users to input the composition of the mixed fertilizer they have or plan to use, and then calculate the application rates based on the nutrient requirements of their selected crop. This feature accommodates the common practice of using customized fertilizer blends and offers flexibility to users, ensuring that the code caters to a broader spectrum of fertilization strategies.

CONCLUSION

In this research paper, a Python-based code is presented, which helps in nutrient management for pulse crops in Kerala. The code's current features, includes data-driven nutrient level interpretation, crop-specific recommendations, user-friendly interaction, efficiency, and iterative capabilities.Looking towards the future, the code stands at the threshold of further development and expansion to better cater to the specific needs of Kerala's agricultural community. The planned enhancements, including a user-friendly graphical interface, expanded crop database, recommendations for soil amendments and micro-nutrients, and assessment of the environmental impact of fertilizer use, will solidify its role as a dynamic, comprehensive, and user-centric agricultural decision support system. The incorporation of recommendations for mixed fertilizers will enhance its versatility and accommodate diverse user preferences. This code is a testament to the fusion of modern technology with traditional farming practices and embodies the continuous betterment of agriculture in Kerala and beyond.

REFERENCES

[1]. Soropa G, Nyamangara J, Nyakatawa EZ, “Nutrient status of sandy soils in smallholder areas of Zimbabwe and the need to develop site-specific fertiliser recommendations for sustainable crop intensification.” South African J Plant Soil ,2019.

[2]. Van der Bom F, Magid J, Jensen LS, “Long-term fertilisation strategies and form affect nutrient budgets and soil test values, soil carbon retention and crop yield resilience”. Plant Soil 434:47-64,2019.

[3]. Gayatri Prem V, Jettin Susan Thomas, “Agriculture sector in Kerala: A glimpse on the Economic Review and the Kerala budget”, KERALA ECONOMY 2022, VOL. 3, NO.2. pp 40-44, 2022.

[4]. Aaron Pettit, “The Importance of Fertilizer Management for Small Farmers”, Cornell University's College of Agriculture and Life Sciences (CALS),2021.

[5]. Vanlauwe B “A fourth principle is required to define conservation agriculture in sub-Saharan Africa: the appropriate use of fertilizer to enhance crop productivity”, Field Crops Res,155:10-13, 2014.

[6]. Lobry de Bruyn L, Andrews S “Are Australian and United states farmers using soil information for soil health management?” Sustainability 8:304, 2016.

[7]. Singh AK, Sohane RK, Aditya, “Leveraging mobile-based technologies for sustainable agricultural development in India”, Indian Journal of Fertilizers, 2016.

[8]. Agashe R, Verma S, Singh P “Opinion of farmers regarding effectiveness of information dissemination through Kisan Suvidha mobile application in Surguja district of Chhattisgarh”, Journal of Krishi Vigyan 2019.

[9]. Doa, Hand Book on Soil fertility,Department of Agriculture, 2018.

[10]. Directorate of extension, Package of Practices Recommendations: Crops,Kerala Agricultural University, 2016.

[11]. T. Yellamanda Reddy, G.H.Sankara Reddy, “Soil-Test Based Fertilizer Recommendation”, Principles of Agronomy, Kalyani publishers, 2016.