

INTERNSHIP REPORT– KALGUDI DIGITAL PVT LIMITED

**A Project Report submitted in partial fulfillment of the requirements for the
award of the degree of**

**BACHELOR OF
TECHNOLOGY IN
COMPUTER SCIENCE AND ENGINEERING**

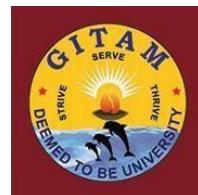
Submitted by

KADIYALA P B ROHIT BHARADWAJ, 121910313006

Under the esteemed guidance of

Mr. Sreekanth Voleti

(Head of the core)



**DEPARTMENT OF COMPUTER SCIENCE &
ENGINEERING**

**GITAM
(Deemed to be University)
VISAKHAPATNAM
May-June 2022**



DECLARATION

I, hereby declare that the Internship report is an original work done in the "**KALGUDI DIGITAL PVT LIMITED**" submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date:07-11-2022

121910313006 KADIYALA P B ROHIT BHARADWAJ Signature:

A handwritten signature in blue ink, which appears to read "Rohit", placed over a light blue rectangular background.

CERTIFICATE



24-Jun-2022

To whom so ever it may concern

This is to certify that Mr. Kadiyala P B Rohit Bharadwaj of Gandhi Institute of Technology and Management has done his project in Building Knowledge Graphs and Graph Databases for FPOs (Using Neo4j and Python) Yield and Profit Prediction Tool for FPOs (Tkinter and Python Data Science) from 16 May 2022 to 24 June 2022 as a part of fulfillment of his curriculum in B Tech-CSE under the guidance of Mr Sreekanth Voleti,(Head of Core).

We take this opportunity of wishing him the very best in his future endeavors.

Thanking you,

For Kalgudi Digital Pvt. Ltd.



A handwritten signature of "Venkatesh" is placed above a circular company stamp. The stamp is blue and contains the text "KALGUDI DIGITAL PRIVATE LIMITED" around the perimeter, with a small star symbol in the center.

Human Resources

TABLE OF CONTENTS

1	Abstract	V
2.	Introduction	1
3.	Literature Review	1-2
4.	Problem Identification & Objectives	3
5.	System Methodology	3
6.	Overview of Technologies	4
7.	Implementation	5-18
7.1	Coding	
7.2	Part-1 And Part-2	
8.	Results & Discussions	19-21
8.1	Part-1	
8.2	Part-2	
10.	Conclusion & Future Scope	22
11.	References	22

1. ABSTRACT

The contribution of agriculture in GDP has declining recent past from 50 per cent in 1950 to 16.5 per cent in 2019-20. In current Covid-19 pandemic, this is the only sector which is contributed positively with 3.4 per cent growth in both the first quarter of the financial year 2020-21. Farmers face different challenges such as scarcity of land and water sources, impassable roads, unavailability of better financial services and new technologies. Government wants to double the farmers' income in 2022-23 by addressing the challenges of farm sector. The government affirmed that farmer producer organizations (FPO) are the most appropriate institutional form around which farmers can mobilize and build their capacity to collectively leverage their production and marketing strength. The members of FPOs are smallholder farmers who organize themselves with the objective of improving farm income through improved production, marketing, and local processing activities. unfortunately, majority of the members in FPOs lack of knowledge on farm inputs, cost of cultivation and profitability. Hence we developed an optimization tool which helps the farmers in planning their farming by allocating most profitable crops and develop an farm by allocating different crops which can result in more returns To make work easy for the FPOs to analyze their farm and farm inputs, the current tools gathers information on climate forecast from India Meteorological Department, yield from crop simulation models and prices from price forecasts developed by ANGRAU. Using all this information the tools provide optimum profitable cropping pattern to the FPO farmers to maximize their profits

2. INTRODUCTION

Collectivization of producers, especially small and marginal farmers, into producer organizations has emerged as one of the most effective pathways to address the many challenges of agriculture, most importantly, improved access to technology, inputs and markets. The Department of Agriculture and Cooperation, Ministry of Agriculture, Government of India has identified farmer producer organizations to be registered under the special provisions of the Companies Act, 2013 as the most appropriate institutional form to mobilize farmers and build their capacity to collectively leverage their production and marketing strength. Though several studies have so far reviewed performances of FPOs in India, most of them evaluated the status of farmer based institutions like cooperatives, mutually aided cooperative credit societies, associations under NGOs etc. Indeed, there is a paucity of literature and studies on FPOs. Also, different implementation models are being adopted by different FPOs operational in various states. There is a lot of problem for FPOs in identifying correct fertilizer composition. In this process we planned developed a tool to calculate fertilizer requirements, yield and profit maximization options for the FPO farmers. This tool gathers values from dataset of University fertilizer recommendations and calculates the correct composition of each component of fertilizer so that there will be no excess or lesser usage of fertilizer doses. Furthermore, it also predicts the Yield and profit for that particular crop in particular area based on IMD (India Meteorological Department) seasonal climate forecast using crop simulation models. This tools helps the FPOs and the registered farmers under such FPO to maximize their profit. Also, A Knowledge graphs are built in the Neo4j software in order to represent entities like Farmer, FPOs, Crops, Stores, Fertilizers, etc.

3. LITERATURE REVIEW

Agricultural extension systems are stretched beyond physical limits while assisting farmers during crop production in the developing world. Information and Communication Technologies are booming worldwide with the introduction of mobile applications in smart phones. Hence several people have come up with various tools and apps to assist the farmers in gathering latest research results and timely contextual information. In this literature review we tried to gather various latest tools developed different startup companies, organizations to help FPOs and farming community.

Indev consultancy (2022) developed an automation software for the Farmer Producer Organization (FPO) which helps small and poor farmers of India to get organized. The application framework will help farmers to participate in high-value markets to sell their products. Further to train the farmers and track their growth, they have developed a web-based dashboard and a Mobile Application with monitoring and evaluation systems. It will provide training, advisory, and support to the farmers and also gather the relevant information to help them compete in the markets.

Kehti Buddy (2022) has developed a platform to make the entire farming process easier for the FPO members. The members will get starting from sowing advice to following climate-resilient farming practices, farmers can get all necessary information via a farm app. FPOs can also view weather forecasts and other information about farming practices, request for input materials or even update data on the app if needed

FarmERP (2022) is an intelligent and next-generation farm management platform. FarmERP's agriculture management system can help FPOs and FPCs manage and optimize various business operations across all plots. FarmERP can empower them with next-generation farm management technology for ensuring high productivity across all their farmer partners' farms.

ICRISAT and its partners have developed an advisory system for FPO registered farmers called "Intelligent agricultural Systems Advisory Tool – ISAT" capable of generating and disseminating data driven location specific advisories that assist farmers in anticipating and responding to the emerging conditions through the season. Using a decision tree approach, a structured and systematic approach to decision making was devised that considers the insights obtained from the analysis of historical climatic conditions, climate and weather forecasts and prevailing environmental conditions (Rao et al., 2019).

Progressive Environment Agriculture Technologies (PEAT), a German startup company developed a mobile application 'Plantix' that assists in detecting damage on plants with the help of a smart phone image for the benefit farmers and FPO members. Mobile internet connectivity enables uploading of image (time and geo-tagged) to a cloud based server where machine learning based algorithms are used to process the optical patterns and detect the diseases. This information is conveyed back to the user within few seconds along with suggested knowledge on disease management including biological and chemical control methods (Srikanth and Alexaander 2018)

4. PROBLEM IDENTIFICATION & OBJECTIVES

Majority of farmers doesn't have the basic knowledge about the proper recommended fertilizer quantities and other inputs they should use for each crop they are growing as majority of them are illiterates. So they will elect a FPO president among them who have more knowledge on agriculture. We designed a Fertilizer tool and Yield and profit prediction tool which helps the FPOs to train the farmers on cropping pattern they have to follow based on seasonal climate forecast and also how to optimize their resources such as land, fertilizers etc., in order to maximize their profits by careful allocation of land with various crops.

5. SYSTEM METHODOLOGY

DATA PREPROCESSING

We performed the data preprocessing on the required data to reduce the feature [i.e columns]. checked for null values, duplicate rows, in-consistence of data and removed those using Pandas

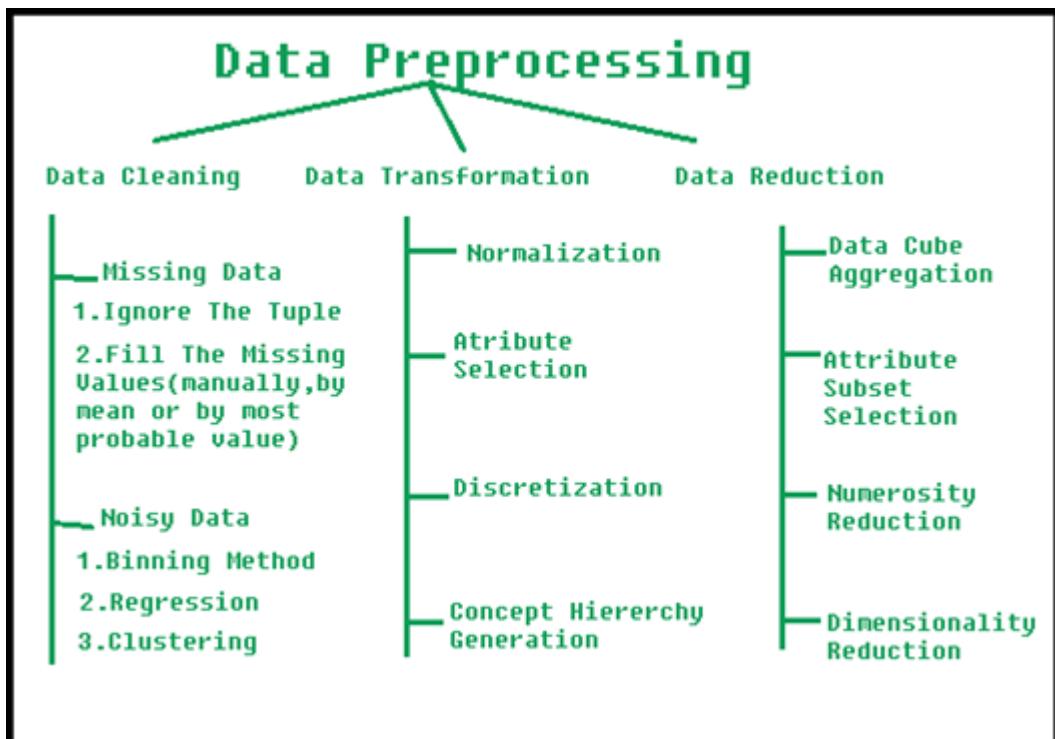


Fig 1. Diagram showing details of data preprocessing

6. OVERVIEW OF TECHNOLOGIES

6.1. TKINTER

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Creating a GUI application using Tkinter is an easy task. We used Tkinter library to design our front end of our tool. We used this package to make our tool more interactive, user friendly which provides the user great satisfaction.

6.2. MATPLOTLIB

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible. it helps in creating and publication of quality plots. It also helps in making interactive figures that can zoom, pan, and update. We used this library to visualize bar graphs for comparing cost of cultivation and returns for a specific crop sown in a particular area.

6.3. PANDAS

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” and “labeled” data both easy and intuitive. We used this package to read the CSV files and use them as data frames for further calculations. We used this packages mainly for data preprocessing

6.4. NEO4J

Neo4j is a graph database. A graph database, instead of having rows and columns has nodes edges and properties. It is more suitable for certain big data analytics applications than row and column databases or free-form JSON document databases for many use cases. A Graph database is used to represent relationships.

7. IMPLEMENTATION:

Here we took the data of three crops, Cotton, Paddy and Maize and predicted the Profit with the help of the data we had from IMD, crop models, cost of cultivation details and output price forecasts .

7.1. CODING

Items	Andhra Pradesh	Gujarat	Karnataka	Madhya Pradesh	Maharashtra	Odisha	Punjab	Tamil Nadu	Telangana
Operational Cost	43234.84	38158.43	30776.43	33784.05	51082.82	37358.20	44640.10	72987.71	51607.21
Human Labour	18759.47	18795.00	11197.30	13336.39	17920.87	21379.35	17460.87	41172.42	19360.04
Animal Labour	987.71	2227.97	3623.98	4087.36	4352.39	1716.81	289.43	0.00	3958.08
Machine Labour	5265.31	5639.64	6149.88	7058.33	11797.25	3805.01	10895.05	9720.42	11428.18
Seed	3812.04	2526.46	2623.52	4002.87	4371.00	3727.80	6862.09	5948.71	5375.91
Fertilizer & Manure	8701.25	6439.09	5448.65	2877.69	9836.68	5536.72	4761.97	9643.38	7457.02
Insecticides	3683.31	213.87	189.73	1269.58	201.59	148.97	2437.09	1071.76	2428.93
Irrigation Charges	488.17	1155.95	571.05	22.45	1154.00	298.36	656.59	3469.63	222.89
Crop Insurance	0.00	0.00	38.68	228.64	0.00	0.00	0.00	0.00	0.00
Miscellaneous	329.16	292.12	139.10	65.97	107.50	43.72	191.02	521.47	90.13
Interest on Working Capital	1208.43	868.33	794.54	834.77	1341.54	701.46	1085.99	1439.92	1286.02
Fixed Costs	33518.92	9913.32	13228.79	11709.39	21217.71	12025.00	26793.18	21536.82	29183.50
Derived Yield (Qtl./Hectare)	42.39	19.33	30.74	30.49	48.69	27.91	36.64	40.20	43.72

DATASET

CODE:

The screenshot shows the PyCharm IDE interface with a Python file named T1.py open. The code reads three CSV files from a local directory, converts them into pandas DataFrames, and then merges them into a single DataFrame. The IDE's status bar at the bottom indicates the current environment settings.

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - T1.py
Test1 Internship > T1.py
Project: T1.py
1 import tkinter.messagebox as messagebox
2 from tkinter import *
3 import pandas
4 import matplotlib.pyplot as plt
5 from tkinter import ttk
6 from PIL import ImageTk, Image
7 import numpy as np
8 df1=pandas.read_excel(r"F:\Downloads\2018-19.xlsx",sheet_name="Maize",engine='openpyxl')
9 df2=pandas.read_excel(r"F:\Downloads\2018-19.xlsx",sheet_name="Paddy",engine='openpyxl')
10 df3=pandas.read_excel(r"F:\Downloads\2018-19.xlsx",sheet_name="Cotton",engine='openpyxl')
11 pd.set_option('display.max_columns', None)
12 pd.set_option('display.max_rows', None)
13 df1.to_csv(r"F:\Downloads\2018-19.csv", index=None)
14 df2.to_csv(r"F:\Downloads\2018-19.csv", index=None)
15 df3.to_csv(r"F:\Downloads\2018-19.csv", index=None)
16 df1.to_csv(r"F:\Downloads\2018-19-1.csv", index=None)
17 df2.to_csv(r"F:\Downloads\2018-19-2.csv", index=None)
18 df3.to_csv(r"F:\Downloads\2018-19-3.csv", index=None)
19 d1 = pd.DataFrame(pd.read_csv(r"F:\Downloads\2018-19-1.csv"))
20 d2 = pd.DataFrame(pd.read_csv(r"F:\Downloads\2018-19-2.csv"))
21 d3 = pd.DataFrame(pd.read_csv(r"F:\Downloads\2018-19-3.csv"))
22
23
24
```

Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console

10:89 CRLF UTF-8 8 spaces* Python 3.10 (user) ENG IN 20:04 10-10-2022

The screenshot shows the PyCharm IDE interface with a Python file named T1.py open. The code defines a function Table_format() which creates a Tkinter window with a Treeview widget. It sets up the geometry, title, and icon for the window. It then creates a Treeview instance and applies styles to it. The columns for the Treeview are defined as SNo, Items, Cotton, Paddy, and Maize. The code uses a style object to apply specific colors and fonts to the treeview items based on their state (disabled or selected). The IDE's status bar at the bottom indicates the current environment settings.

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - T1.py
Test1 Internship > T1.py
Project: T1.py
26 def Table_format():
27     c=Toplevel(root)
28     c.geometry('800x800')
29     c.title("Cultivation Details")
30     c.iconbitmap("download.ico")
31     tv = ttk.Treeview(c)
32     style = ttk.Style()
33     style.theme_use('clam')
34     def fix_map(option):
35         return [elm for elm in style.map('Treeview', query_opt=option)
36                 if elm[:2] != ('!disabled', '!selected')]
37     style.map('Treeview', foreground=fix_map('foreground'),
38               background=fix_map('background'))
39     tv.tag_configure('bold', font=('bold'), foreground='red')
40
41     tv['columns'] = ('SNo', 'Items', 'Cotton','Paddy','Maize')
42     tv.column('#0', width=0, stretch=NO)
43     tv.column('SNo', anchor= CENTER, width=120)
44     tv.column('Items', anchor= CENTER, width=120)
45     tv.column('Cotton', anchor= CENTER, width=120)
46     tv.column('Paddy', anchor= CENTER, width=120)
47     tv.column('Maize', anchor= CENTER, width=120)
48
Table_format()
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
49:31 CRLF UTF-8 8 spaces* Python 3.10 (user) ENG IN 20:05 10-10-2022
```

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** Test1
- File:** Internship/T1.py
- Code:** Python script T1.py containing calculations for Operational Cost, Human Labour, Animal Labour, Machine Labour, Seed, Fertilizer & Manure, Insecticides, Irrigation Charges, Crop Insurance, and Miscellaneous categories.
- Toolbars:** Standard PyCharm toolbars for File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help.
- Status Bar:** Shows file path (T1.py - T1.py), line count (154), character count (374), and line number (11).
- Bottom Bar:** Includes Version Control, Python Packages, TODO, Python Console, Problems, Terminal, Services, and Graph Database Console.

```
tv.heading('#0', text='', anchor= CENTER)
tv.heading('SNo', text='SNo', anchor= CENTER)
tv.heading('Items', text='Items', anchor= CENTER)
tv.heading('Cotton', text='Cotton', anchor= CENTER)
tv.heading('Paddy', text='Paddy', anchor= CENTER)
tv.heading('Maize', text='Maize', anchor= CENTER)
x = c3.get()
Tot_c = round((d3[x][1]) * float(var1.get())) + round((d3[x][2]) * float(var3.get())) + round((d3[x][3]) * float(var1.get())) + round((d3[x][4]) * float(var3.get()))
Tot_p = round((d2[x][1]) * float(var3.get())) + round((d2[x][2]) * float(var3.get())) + round((d2[x][3]) * float(var3.get())) + round((d2[x][4]) * float(var3.get())) + round((d2[x][5]) * float(var3.get())) + round((d2[x][6]) * float(var3.get())) + round((d2[x][7]) * float(var3.get())) + round((d2[x][8]) * float(var3.get())) + round((d2[x][9]) * float(var3.get())) + round((d2[x][10]) * float(var3.get()))
Tot_m = round((d1[x][1]) * float(var2.get())) + round((d1[x][2]) * float(var2.get())) + round((d1[x][3]) * float(var2.get())) + round((d1[x][4]) * float(var2.get())) + round((d1[x][5]) * float(var2.get())) + round((d1[x][6]) * float(var2.get())) + round((d1[x][7]) * float(var2.get())) + round((d1[x][8]) * float(var2.get())) + round((d1[x][9]) * float(var2.get())) + round((d1[x][10]) * float(var2.get()))
tv.insert(parent='', index=0, iid=0, text='values=(1, 'Operational Cost', Tot_c,Tot_p, Tot_m)')
tv.insert(parent='', index=1, iid=1, text='values=(2, 'Human Labour', round((d3[x][1]) * float(var1.get())), round((d2[x][1]) * float(var3.get())))
tv.insert(parent='', index=2, iid=2, text='values=(3, 'Animal Labour', round((d3[x][2]) * float(var1.get())), round((d2[x][2]) * float(var3.get())))
tv.insert(parent='', index=3, iid=3, text='values=(4, 'Machine Labour', round((d3[x][3]) * float(var1.get())), round((d2[x][3]) * float(var3.get())))
tv.insert(parent='', index=4, iid=4, text='values=(5, 'Seed', round((d3[x][4]) * float(var1.get())), round((d2[x][4]) * float(var3.get())))
tv.insert(parent='', index=5, iid=5, text='values=(6, 'Fertilizer & Manure', round(l[0]), round(l1[0]), round(l2[0]))
tv.insert(parent='', index=6, iid=6, text='values=(7, 'Insecticides', round((d3[x][6]) * float(var1.get())), round((d2[x][6]) * float(var3.get())))
tv.insert(parent='', index=7, iid=7, text='values=(8, 'Irrigation Charges', round((d3[x][7]) * float(var1.get())), round((d2[x][7]) * float(var3.get())))
tv.insert(parent='', index=8, iid=8, text='values=(9, 'Crop Insurance', round((d3[x][8]) * float(var1.get())), round((d2[x][8]) * float(var3.get())))
tv.insert(parent='', index=9, iid=9, text='values=(10, 'Miscellaneous', round((d3[x][9]) * float(var1.get())), round((d2[x][9]) * float(var3.get())))

```

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** Test1
- File:** Internship/T1.py
- Code:** Python script T1.py containing code to insert data into a tree view (tv) for various categories.
- Toolbars:** Standard PyCharm toolbars for File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help.
- Status Bar:** Shows file path (T1.py - T1.py), line count (154), character count (374), and line number (11).
- Bottom Bar:** Includes Version Control, Python Packages, TODO, Python Console, Problems, Terminal, Services, and Graph Database Console.

```
tv.insert(parent='', index=1, iid=1, text='values=(2, 'Human Labour', round((d3[x][1]) * float(var1.get())), round((d2[x][1]) * float(var3.get())))
tv.insert(parent='', index=2, iid=2, text='values=(3, 'Animal Labour', round((d3[x][2]) * float(var1.get())), round((d2[x][2]) * float(var3.get())))
tv.insert(parent='', index=3, iid=3, text='values=(4, 'Machine Labour', round((d3[x][3]) * float(var1.get())), round((d2[x][3]) * float(var3.get())))
tv.insert(parent='', index=4, iid=4, text='values=(5, 'Seed', round((d3[x][4]) * float(var1.get())), round((d2[x][4]) * float(var3.get())))
tv.insert(parent='', index=5, iid=5, text='values=(6, 'Fertilizer & Manure', round(l[0]), round(l1[0]), round(l2[0]))
tv.insert(parent='', index=6, iid=6, text='values=(7, 'Insecticides', round((d3[x][6]) * float(var1.get())), round((d2[x][6]) * float(var3.get())))
tv.insert(parent='', index=7, iid=7, text='values=(8, 'Irrigation Charges', round((d3[x][7]) * float(var1.get())), round((d2[x][7]) * float(var3.get())))
tv.insert(parent='', index=8, iid=8, text='values=(9, 'Crop Insurance', round((d3[x][8]) * float(var1.get())), round((d2[x][8]) * float(var3.get())))
tv.insert(parent='', index=9, iid=9, text='values=(10, 'Miscellaneous', round((d3[x][9]) * float(var1.get())), round((d2[x][9]) * float(var3.get())))

```

```
tv.insert(parent='', index=10, iid=10, text='',
          values=('11', 'Interest on Working Capital', round((d3[x][10]) * float(var1.get())),
                  round((d2[x][10]) * float(var3.get())), round((d1[x][10]) * float(var2.get())))
tv.insert(parent='', index=11, iid=11, text='',
          values=('12', 'Fixed Costs', round((d3[x][11]) * float(var1.get())), round((d2[x][11]) * float(var3.get())), round((d1[x][11]) * float(var2.get())))
tv.insert(parent='', index=12, iid=12, text='',
          values=(
              '13', 'Total', round(Tot_c+round((d3[x][11]) * float(var1.get()))), round(Tot_p+round((d2[x][11]) * float(var3.get()))),
              round(Tot_m+round((d1[x][11]) * float(var2.get())))))
for row in tv.get_children()[-1:]:
    tv.item(row, tags='bold')
tv.pack()
if(var4.get() == "Normal"):
    y_m = ((d1[x][12]) * float(var2.get())) * 2100
    y_p = ((d2[x][12]) * float(var3.get())) * 2000
    y_c = ((d3[x][12]) * float(var1.get())) * 7000
    pl_c = y_c - (Tot_c + round((d3[x][11]) * float(var1.get())))
    pl_p = y_p - (Tot_p + round((d2[x][11]) * float(var3.get())))
    pl_m = y_m - (Tot_m + round((d1[x][11]) * float(var2.get())))
    if (pl_c > 0):
        X1 = Label(c,
                    text="Profit for Cotton" + '(' + str(var1.get()) + "ha" + ')' + ":" + "Rs" + str(
                        round(abs(pl_c), 0)), font=("bold", 16), foreground="Green").place(x=80, y=300)
    if (pl_p < 0):
        X2 = Label(c,
                    text="Loss for Paddy" + '(' + str(var3.get()) + "ha" + ')' + ":" + "Rs" + str(
                        round(abs(pl_p), 0)), font=("bold", 16), foreground="Red").place(x=80, y=350)
    if (pl_m > 0):
        X3 = Label(c,
                    text="Profit for Maize:" + '(' + str(var2.get()) + "ha" + ')' + ":" + "Rs" + str(
                        round(abs(pl_m), 0)), font=("bold", 16), foreground="Green").place(x=80, y=400)
    if (pl_m < 0):
        X3 = Label(c,
                    text="Loss for Maize" + '(' + str(var2.get()) + "ha" + ')' + ":" + "Rs" + str(
                        round(abs(pl_m), 0)), font=("bold", 16), foreground="Red").place(x=80, y=400)
```

```
if (pl_c < 0):
    X1 = Label(c,
                text="Loss for Cotton" + '(' + str(var1.get()) + "ha" + ')' + ":" + "Rs" + str(
                    round(abs(pl_c), 0)), font=("bold", 16), foreground="Red").place(x=80, y=300)
if (pl_p > 0):
    X2 = Label(c,
                text="Profit for Paddy" + '(' + str(var3.get()) + "ha" + ')' + ":" + "Rs" + str(
                    round(abs(pl_p), 0)), font=("bold", 16), foreground="Green").place(x=80, y=350)
if (pl_p < 0):
    X2 = Label(c,
                text="Loss for Paddy" + '(' + str(var3.get()) + "ha" + ')' + ":" + "Rs" + str(
                    round(abs(pl_p), 0)), font=("bold", 16), foreground="Red").place(x=80, y=350)
if (pl_m > 0):
    X3 = Label(c,
                text="Profit for Maize:" + '(' + str(var2.get()) + "ha" + ')' + ":" + "Rs" + str(
                    round(abs(pl_m), 0)), font=("bold", 16), foreground="Green").place(x=80, y=400)
if (pl_m < 0):
    X3 = Label(c,
                text="Loss for Maize" + '(' + str(var2.get()) + "ha" + ')' + ":" + "Rs" + str(
                    round(abs(pl_m), 0)), font=("bold", 16), foreground="Red").place(x=80, y=400)
```

The screenshot shows the PyCharm IDE interface with a Python file named T1.py open. The code is for a bar chart visualization. It includes a function `bar()` that calculates total values for Cotton, Paddy, and Maize, and then plots them using matplotlib. The plot is titled with the current name and district. The code also contains a section for handling a button click labeled "Visualize". The PyCharm interface includes a Project tool window, a Registry Explorer, and a Structure tool window. The status bar at the bottom shows system information like temperature (27°C), date (10-10-2022), and time (20:07).

```
141
142     def bar():
143         w = 0.4
144         z1 = Tot_c + round((d3[x][11]) * float(var1.get()))
145         z2 = Tot_p + round((d2[x][11]) * float(var3.get()))
146         z3 = Tot_m + round((d1[x][11]) * float(var2.get()))
147         Total_1 = z1 + z2 + z3
148         Total_2 = y_c + y_m + y_p
149         x1 = ["Cotton", "Rice", "Maize", "Total"]
150         A1 = [z1, z2, z3, Total_1]
151         A2 = [y_c, y_p, y_m, Total_2]
152         n = 4
153         r = np.arange(n)
154         plt.bar(r, A1, w, label="COC")
155         plt.bar(r + w, A2, w, label="Returns")
156         plt.xticks(r + w / 2, x1)
157         plt.legend()
158         plt.title(Name.get() + " " + "FP0" + " " + District.get())
159         plt.show()
160         plt.clf()
161
162         b4 = Button(root, text='Visualize', width=15, bg='brown', fg='white', command=bar).place(x=450, y=550)
163     elif(var4.get()=="Below Normal"):
164         ...
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
```

This screenshot shows the same PyCharm IDE session with the code expanded to show calculations for profit or loss based on the value of `var4.get()`. The code uses conditional statements to calculate `y_m`, `y_p`, and `y_c` with different multipliers. It then creates labels `X1` and `X2` to display the results for Cotton and Paddy respectively, using bold font and green/red colors for positive/negative values. The PyCharm interface and system status bar are identical to the first screenshot.

```
163
164     elif(var4.get()=="Below Normal"):
165         y_m = ((d1[x][12]*0.7* float(var2.get())) * 2100
166         y_p = ((d2[x][12]*0.9* float(var3.get())) * 2000
167         y_c = ((d3[x][12] *0.7*float(var1.get())) * 7000
168
169         pl_c = y_c - (Tot_c + round((d3[x][11]) * float(var1.get())))
170         pl_p = y_p - (Tot_p + round((d2[x][11]) * float(var3.get())))
171         pl_m = y_m - (Tot_m + round((d1[x][11]) * float(var2.get())))
172         if (pl_c > 0):
173             X1 = Label(c,
174                         text="Profit for Cotton" + '(' + str(var1.get()) + "ha" + ')' + ":" + "Rs" + str(
175                                         round(pl_c, 0)), font=("bold", 16), foreground="Green").place(x=80, y=300)
176         if (pl_c < 0):
177             X1 = Label(c,
178                         text="Loss for Cotton" + '(' + str(var1.get()) + "ha" + ')' + ":" + "Rs" + str(
179                                         round(abs(pl_c), 0)), font=("bold", 16), foreground="Red").place(x=80, y=300)
180
181         if (pl_p > 0):
182             X2 = Label(c,
183                         text="Profit for Paddy" + '(' + str(var3.get()) + "ha" + ')' + ":" + "Rs" + str(
184                                         round(pl_p, 0)), font=("bold", 16), foreground="Green").place(x=80, y=350)
185         if (pl_p < 0):
186             X2 = Label(c,
187                         text="Loss for Paddy" + '(' + str(var3.get()) + "ha" + ')' + ":" + "Rs" + str(
188                                         round(abs(pl_p), 0)), font=("bold", 16), foreground="Red").place(x=80, y=350)
```

The screenshot shows the PyCharm IDE interface with a Python file named T1.py open. The code is part of a graphical user interface (GUI) application. It includes several functions: `def profit_label()`, `def bar()`, and `def bar()`. The `profit_label` function creates labels for 'Loss for Paddy' and 'Profit for Maize' with specific text, font, and placement details. The first `bar` function calculates totals for Cotton, Rice, and Maize, then plots them as bars with labels. The second `bar` function visualizes the data. The code uses Tkinter for the GUI and NumPy for calculations. The PyCharm interface includes toolbars, a sidebar with 'Project', 'Registry Explorer', and 'Structure' tabs, and a bottom status bar showing the date and time.

```
185     text="Loss for Paddy" + '(' + str(var3.get()) + "ha" + ')' + ":" + "Rs" + str(
186         round(abs(pl_p), 0)), font=("bold", 16),
187         foreground="Red").place(x=80, y=350)
188     if (pl_m > 0):
189         X3 = Label(c,
190             text="Profit for Maize:" + '(' + str(var2.get()) + "ha" + ')' + ":" + "Rs" + str(
191                 round(pl_m, 0)), font=("bold", 16),
192                 foreground="Green").place(x=80, y=400)
193     if (pl_m < 0):
194         X3 = Label(c,
195             text="Loss for Maize" + '(' + str(var2.get()) + "ha" + ')' + ":" + "Rs" + str(
196                 round(pl_m, 0)), font=("bold", 16),
197                 foreground="Red").place(x=80, y=400)
198
199     def bar():
200         w = 0.4
201         z1 = Tot_c + round((d3[x][11]) * float(var1.get()))
202         z2 = Tot_p + round((d2[x][11]) * float(var3.get()))
203         z3 = Tot_m + round((d1[x][11]) * float(var2.get()))
204         Total_1 = z1 + z2 + z3
205         Total_2 = y_c + y_m + y_p
206         x1 = ["Cotton", "Rice", "Maize", "Total"]
207         A1 = [z1, z2, z3, Total_1]
208
209         A1 = [z1, z2, z3, Total_1]
210         A2 = [y_c, y_p, y_m, Total_2]
211         n = 4
212         r = np.arange(n)
213         plt.bar(r, A1, w, label="COC")
214         plt.bar(r + w, A2, w, label="Returns")
215         plt.xticks(r + w / 2, x1)
216         plt.legend()
217         plt.title(Name.get() + " " + "FPO" + " " + District.get())
218         plt.show()
219         plt.clf()
220
221         b4 = Button(root, text='Visualize', width=15, bg='brown', fg='white', command=bar).place(x=450, y=550)
222     elif (var4.get() == "Above Normal"):
223         y_m = ((d1[x][12] * 1.2 * float(var2.get()))) * 2100
224         y_p = ((d2[x][12] * 1.1 * float(var3.get()))) * 2000
225         y_c = ((d3[x][12] * 1.2 * float(var1.get()))) * 7000
226
227         pl_c = y_c - (Tot_c + round((d3[x][11]) * float(var1.get())))
228         pl_p = y_p - (Tot_p + round((d2[x][11]) * float(var3.get())))
229         pl_m = y_m - (Tot_m + round((d1[x][11]) * float(var2.get())))
230         if (pl_m > 0):
231             X1 = Label(c,
```

This screenshot shows the same PyCharm IDE interface as the previous one, but with different code visible in the editor. The code continues from the previous snippet, specifically the second `bar` function. It calculates adjusted values for Cotton, Rice, and Maize based on a multiplier of 1.2 or 1.1. It then subtracts these from the total to get the profit or loss. The code uses NumPy for rounding. The PyCharm interface remains consistent with the first screenshot, including the toolbar, sidebar, and status bar.

```
207         pl_c = y_c - (Tot_c + round((d3[x][11]) * float(var1.get())))
208         pl_p = y_p - (Tot_p + round((d2[x][11]) * float(var3.get())))
209         pl_m = y_m - (Tot_m + round((d1[x][11]) * float(var2.get())))
210         if (pl_m > 0):
211             X1 = Label(c,
212                 text="Profit For Cotton" + '(' + str(var1.get()) + "ha" + ')' + ":" + "Rs" + str(
213                     round(pl_c, 0)), font=("bold", 16),
214                     foreground="Green").place(x=80, y=350)
215         if (pl_p > 0):
216             X2 = Label(c,
217                 text="Profit For Rice" + '(' + str(var3.get()) + "ha" + ')' + ":" + "Rs" + str(
218                     round(pl_p, 0)), font=("bold", 16),
219                     foreground="Green").place(x=80, y=400)
220         if (pl_m > 0):
221             X3 = Label(c,
222                 text="Profit For Maize" + '(' + str(var2.get()) + "ha" + ')' + ":" + "Rs" + str(
223                     round(pl_m, 0)), font=("bold", 16),
224                     foreground="Green").place(x=80, y=450)
225
226         A1 = [pl_c, pl_p, pl_m, Total_1]
227         A2 = [y_c, y_p, y_m, Total_2]
228         n = 4
229         r = np.arange(n)
230         plt.bar(r, A1, w, label="COC")
231         plt.bar(r + w, A2, w, label="Returns")
232         plt.xticks(r + w / 2, x1)
233         plt.legend()
234         plt.title(Name.get() + " " + "FPO" + " " + District.get())
235         plt.show()
236         plt.clf()
237
238         b4 = Button(root, text='Visualize', width=15, bg='brown', fg='white', command=bar).place(x=450, y=550)
239     elif (var4.get() == "Below Normal"):
240         y_m = ((d1[x][12] * 0.8 * float(var2.get()))) * 2100
241         y_p = ((d2[x][12] * 0.9 * float(var3.get()))) * 2000
242         y_c = ((d3[x][12] * 0.8 * float(var1.get()))) * 7000
243
244         pl_c = y_c - (Tot_c + round((d3[x][11]) * float(var1.get())))
245         pl_p = y_p - (Tot_p + round((d2[x][11]) * float(var3.get())))
246         pl_m = y_m - (Tot_m + round((d1[x][11]) * float(var2.get())))
247         if (pl_m > 0):
248             X1 = Label(c,
```

The screenshot shows the PyCharm IDE interface with the following details:

- File Menu:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help.
- Project Bar:** Test1 > Internship > T1.py
- Code Editor:** The file T1.py contains Python code for a graphical user interface (GUI) using Tkinter. The code defines labels X1, X2, and X3 to display text based on variable values (pl_c, pl_p, pl_m). The text includes calculations for Loss or Profit for Cotton, Paddy, and Maize, along with their respective amounts and unit symbols (Rs).
- Side Panels:**
 - Project:** Shows the project structure with files like T1.py, T1.pyw, and __init__.py.
 - Bookmarks:** Shows a list of bookmarks.
 - Registry-Holder:** Shows a list of registry entries.
 - Structure:** Shows the code structure with navigation icons.
- Bottom Navigation:** Version Control, Python Packages, TODO, Python Console, Problems, Terminal, Services, Graph Database Console.
- System Status:** 79:56 CRLF UTF-8 8 spaces* Python 3.10 (user), 27°C Cloudy, ENG IN, 10-10-2022, 2009.
- Toolbars:** Standard Windows-style toolbars for copy, paste, cut, etc.

The screenshot shows a PyCharm IDE interface with the following details:

- File Menu:** File, Edit, View, Navigate, Code, Befactor, Run, Tools, VCS, Window, Help.
- Project Bar:** Test1 > Internship > T1.py
- Code Editor:** The code is for a function `bar()` which calculates totals for cotton, rice, maize, and total production across four categories. It uses `float(var1.get())`, `float(var2.get())`, and `float(var3.get())` to get values from GUI components. The code then creates a bar chart using `plt.bar()` with labels "COC" and "Returns".

```
def bar():
    w = 0.4
    z1 = Tot_c + round((d3[x][11]) * float(var1.get()))
    z2 = Tot_p + round((d2[x][11]) * float(var2.get()))
    z3 = Tot_m + round((d1[x][11]) * float(var3.get()))
    Total_1 = z1 + z2 + z3
    Total_2 = y_c + y_m + y_p
    x1 = ["Cotton", "Rice", "Maize", "Total"]
    A1 = [z1, z2, z3, Total_1]
    A2 = [y_c, y_p, y_m, Total_2]
    n = 4
    r = np.arange(n)
    plt.bar(r, A1, w, label="COC")
    plt.bar(r + w, A2, w, label="Returns")
    plt.xticks(r + w / 2, x1)
    plt.legend()
    plt.title(Name.get() + " " + "FPO" + " " + District.get())
    plt.show()
    plt.clf()

b4 = Button(root, text='Visualize', width=15, bg='brown', fg='white', command=bar).place(x=450, y=550)
```
- Else Block:** If `var4.get() == "Above Norm..."`, it shows an info message box.

```
else:
    msbox.showinfo(title='FRC', message='Invalid Data')
```
- Table Format:** A conditional block based on `Table.format()` and `elif var4.get() == "Above Norm..."`.
- Bottom Bar:** Version Control, Python Packages, TODO, Python Console, Problems, Terminal, Services, Graph Database Console.
- System Icons:** Weather (27°C Cloudy), Taskbar icons (Windows, Search, File Explorer, Task View, Mail, Google Chrome, Microsoft Edge, File Explorer, Power BI, Word, Excel, OneDrive).
- Right Sidebar:** Includes a graph database section with 154 nodes, 374 edges, 11 clusters, and a notifications section.

File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - T1.py

Test1 Internship > T1.py

```
global l
x = c3.get()
if(var1.get()!=0.0 and var1.get()!=" "):
    l = []
    MOP = (15 / 60) * 100 * float(var1.get())
    DAP = (60 / 46) * 100 * float(var1.get())
    z = ((18 * DAP) / 100) / float(var1.get())
    N1 = 30 - z
    N2 = (N1 / 46) * 100 * float(var1.get())

    N1 = round(N2, 1)
    P = round(DAP, 1)
    K = round(MOP, 1)

    #####
    U = (40 / 46) * 100 * float(var1.get())
    N11 = round(U, 1)
    P1 = round(0, 1)
    K1 = round(0, 1)

    #####
    U1 = (40 / 46) * 100 * float(var1.get())
    MOP1 = (15 / 60) * 100 * float(var1.get())
```

Table.format() > elif (var4.get() == "Above Norm... > bar)

Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console

267:21 CRLF UTF-8 8 spaces* Python 3.10 (user) 20:09 10-10-2022

Cloudy 27°C ENG IN

File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - T1.py

Test1 Internship > T1.py

```
Ni2 = round(U1, 1)
P2 = round(0, 1)
K2 = round(MOP1, 1)

Ur = (Ni + Ni1 + Ni2) * Urea
Da = (P + P1 + P2) * DAP_price
Mo = (K + K1 + K2) * MOP_price
T = Ur + Da + Mo

HL1 = (d3[x][1]) * float(var1.get())
AL1 = (d3[x][2]) * float(var1.get())
ML1 = (d3[x][3]) * float(var1.get())
Seed1 = (d3[x][4]) * float(var1.get())
FM1 = T
z1 = d3[x].replace(d3[x][5], FM1)
l.append(z1[5])
Insec1 = (d3[x][6]) * float(var1.get())
Irr11 = (d3[x][7]) * float(var1.get())
CI1 = (d3[x][8]) * float(var1.get())
Misc1 = (d3[x][9]) * float(var1.get())
WC1 = (d3[x][10]) * float(var1.get())
FC1 = (d3[x][11]) * float(var1.get())
Total1 = HL1 + AL1 + ML1 + Seed1 + FM1 + Insec1 + Irr11 + CI1 + Misc1 + WC1 + FC1
```

Table.format() > elif (var4.get() == "Above Norm... > bar)

Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console

267:21 CRLF UTF-8 8 spaces* Python 3.10 (user) 20:10 10-10-2022

Cloudy 27°C ENG IN

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** Test1
- File:** Internship / T1.py
- Code:**

```
    rura = Urea * float(var1.get())
    Total1 = HL1 + AL1 + ML1 + Seed1 + FM1 + Insec1 + Irr1 + CI1 + Misc1 + WC1 + FC1

    L1 = Label(root,
               text="Cost of Cultivation for Cotton:\t\t" + "Rs\t" + str(round(Total1, 0))).place(x=80, y=400)

    else:
        messagebox.showinfo(title='FRC', message='Cotton Crop Area cannot be Zero or Empty')

def Kharif_Total_maize():
    global l2
    x = c3.get()
    if (var2.get() != 0.0 and var2.get() != ""):
        l2 = []
        MOP = (25 / 60) * 100 * float(var2.get())
        DAP = (60 / 46) * 100 * float(var2.get())
        z = ((18 * DAP) / 100) / float(var2.get())
        N1 = (200 / 3) - z
        N2 = (N1 / 46) * 100 * float(var2.get())
        Ni = round(N2, 1)
        P = round(DAP, 1)
        K = round(MOP, 1)
        U = ((200 / 3) / 46) * 100 * float(var2.get())
        Table_format()
```
- Toolbars and Status Bar:** Version Control, Python Packages, TODO, Python Console, Problems, Terminal, Services, Graph Database Console. Status bar shows: 267:21 CRLF UTF-8 8 spaces* Python 3.10 (user)
- System Taskbar:** Cloudy, 27°C, ENG IN, 20:10, 10-10-2022.

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** Test1
- File:** Internship / T1.py
- Code:**

```
    Ni1 = round(U, 1)
    P1 = round(0, 1)
    K1 = round(0, 1)
    U1 = ((200 / 3) / 46) * 100 * float(var2.get())
    MOP1 = (25 / 60) * 100 * float(var2.get())
    Ni2 = round(U1, 1)
    P2 = round(0, 1)
    K2 = round(MOP1, 1)
    Ur = (Ni + Ni1 + Ni2) * Urea
    Da = (P + P1 + P2) * DAP_price
    Mo = (K + K1 + K2) * MOP_price
    T = Ur + Da + Mo
    x = c3.get()
    HL2 = (d1[x][1]) * float(var2.get())
    AL2 = (d1[x][2]) * float(var2.get())
    ML2 = (d1[x][3]) * float(var2.get())
    Seed2 = (d1[x][4]) * float(var2.get())
    FM2 = T
    z2 = d1[x].replace(d1[x][5], FM2)
    l2.append(z2[5])
    Insec2 = (d1[x][6]) * float(var2.get())
    Irr2 = (d1[x][7]) * float(var2.get())
    CI2 = (d1[x][8]) * float(var2.get())
    Misc2 = (d1[x][9]) * float(var2.get())
    Table_format()
```
- Toolbars and Status Bar:** Version Control, Python Packages, TODO, Python Console, Problems, Terminal, Services, Graph Database Console. Status bar shows: 267:21 CRLF UTF-8 8 spaces* Python 3.10 (user)
- System Taskbar:** Cloudy, 27°C, ENG IN, 20:10, 10-10-2022.

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** Internship
- File:** T1.py
- Code Content:**

```
395
396     U = (60 / 46) * 100 * float(var3.get())
397     Ni1 = round(U, 1)
398     P1 = round(0, 1)
399     K1 = round(0, 1)
400     U1 = (60 / 46) * 100 * float(var3.get())
401     MOP1 = (45 / 60) * 100 * float(var3.get())
402     Ni2 = round(U1, 1)
403     P2 = round(0, 1)
404     K2 = round(MOP1, 1)
405     Ur = (Ni + Ni1 + Ni2) * Urea
406     Da = (P + P1 + P2) * DAP_price
407     Mo = (K + K1 + K2) * MOP_price
408     I = Ur + Da + Mo
409     x = c3.get()
410     HL3 = (d2[x][1]) * float(var3.get())
411     AL3 = (d2[x][2]) * float(var3.get())
412     ML3 = (d2[x][3]) * float(var3.get())
413     Seed3 = (d2[x][4]) * float(var3.get())
414     FM3 = T
415     z3 = d2[x].replace(d2[x][5], FM3)
416     l1.append(z3[5])
417     Insec3 = (d2[x][6]) * float(var3.get())
418     Irr3 = (d2[x][7]) * float(var3.get())
419
Table.format() elif var4.get() == "Above Norm...": bar()
```

- Toolbars and Status Bar:** The status bar at the bottom shows "267:21 CRLF UTF-8 8 spaces* Python 3.10 (user)".
- SIDE BAR:** Includes Project, Bookmarks, Registry Explorer, and Structure.
- RIGHT SIDE:** Includes a vertical color scheme palette, Notifications, and Run.

The screenshot shows the PyCharm IDE interface with a Python file named T1.py open. The code implements a GUI application for calculating cultivation costs. It uses Tkinter for the graphical interface, variables for input fields, and floating-point arithmetic for calculations. The code includes error handling for empty input fields.

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - T1.py
Test1 Internship > T1.py
Project T1.py
418     CI3 = (d2[x][8]) * float(var3.get())
419     Misc3 = (d2[x][9]) * float(var3.get())
420     WC3 = (d2[x][10]) * float(var3.get())
421     FC3 = (d2[x][11]) * float(var3.get())
422     Total3 = HL3 + AL3 + ML3 + Seed3 + FM3 + Insec3 + Irri3 + CI3 + Misc3 + WC3 + FC3
423     L3 = Label(root,
424                 text="Cost of Cultivation for Paddy\n" + str(round(Total3, 0))).place(x=80, y=500)
425     else:
426         messagebox.showinfo(title='FRC', message='Paddy Crop Area cannot be Zero or Empty')
427     # GUI tkinter Part#####
428     root = Tk()
429     c3 = StringVar()
430     root.geometry('700x800')
431     root.title("FRC")
432     Name = StringVar()
433     District = StringVar()
434     var1 = DoubleVar()
435     var2 = DoubleVar()
436     var3 = DoubleVar()
437     var4 = StringVar()
438     Urea=5.52
439     DAP_price=22.5
440     MOP_price=17.77
441     global L1,L2,L3
Table.format() > elif (var4.get() == "Above Normal") > bar0
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
267:21 CRLF UTF-8 8 spaces* Python 3.10 (user) ENG IN 20:11 10-10-2022
```

The screenshot shows the PyCharm IDE interface with the same Python file T1.py open. This version of the code includes two new functions: clear() and COC(). The clear() function resets all input fields to their initial state. The COC() function initializes labels and entry fields for entering district names and FPO details.

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - T1.py
Test1 Internship > T1.py
Project T1.py
441     global L1,L2,L3
442     global Total1,Total2,Total3,Z1
443
444     def clear():
445         Name.set("")
446         District.set("")
447         var1.set(0)
448         var2.set(0)
449         var3.set(0)
450         c3.set("Select")
451
452     def COC():
453         Btotal()
454         Kharif_Total_maize()
455         Kharif_Total_paddy()
456         b5 = Button(root, text='Details', width=15, bg='brown', fg='white', command=Table_format).place(x=450, y=450)
457
458         l0 = Label(root, text="FERTIGURU", font=("bold", 20), foreground="Red",
459                     background="#FFF000").place(x=240, y=20)
460         l1 = Label(root, text="Name of the FPO", width=20, font=("bold", 10)).place(x=80, y=80)
461         e1 = Entry(root, textvariable=Name).place(x=300, y=80)
462         l2 = Label(root, text="District Name", width=20, font=("bold", 10)).place(x=80, y=130)
Table.format() > elif (var4.get() == "Above Normal") > bar0
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
267:21 CRLF UTF-8 8 spaces* Python 3.10 (user) ENG IN 20:11 10-10-2022
```

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - T1.py
Test1 Internship > T1.py
Project T1.py x
464 e2 = Entry(root, textvariable=District).place(x=300, y=130)
465
466 l6_1 = Label(root, text="Crop Area(ha) of Cotton:", width=20, font=("bold", 10)).place(x=80, y=180)
467 e6_1 = Entry(root, textvariable=var1).place(x=300, y=180)
468
469 l6_2 = Label(root, text="Crop Area(ha) of Maize:", width=20, font=("bold", 10)).place(x=80, y=230)
470 e6_2 = Entry(root, textvariable=var2).place(x=300, y=230)
471
472 l6_3 = Label(root, text="Crop Area(ha) of Paddy:", width=20, font=("bold", 10)).place(x=80, y=280)
473 e6_3 = Entry(root, textvariable=var3).place(x=300, y=280)
474
475 l7 = Label(root, text="State Name:", width=20, font=("bold", 10)).place(x=80, y=320)
476 list3 = []
477 for col in di.columns:
478     list3.append(col)
479 list3.sort()
480 list3.remove("Items")
481 droplist2 = OptionMenu(root, c3, *list3)
482 droplist2.config(width=15)
483 c3.set('Select')
484 droplist2.place(x=300, y=320)
485 b3 = Button(root, text='Clear', width=15, bg='brown', fg='white', command=clear).place(x=250, y=550)
486 root.configure(background='#C1E1D2')
487
Table.format() > elif (var4.get() == "Above Normal" > bar())
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
267:21 CRLF UTF-8 8 spaces* Python 3.10 (user)
Cloudy ENG IN 20:11 10-10-2022
```

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - T1.py
Test1 Internship > T1.py
Project T1.py x
488 b2 = Button(root, text='Total Cost', width=15, bg='brown', fg='white', command=COC).place(x=80, y=550)
489 l5 = Label(root, text="Climate Forecast:", width=20, font=("bold", 10)).place(x=80, y=360)
490 Radiobutton(root, text="Normal", variable=var4, value="Normal").place(x=280, y=360)
491 Radiobutton(root, text="Above Normal", variable=var4, value="Above Normal").place(x=380, y=360)
492 Radiobutton(root, text="Below Normal", variable=var4, value="Below Normal").place(x=500, y=360)
493 def display_msg():
494     msgbox.showinfo(title='FRC', message='😊 Thank You For Using Our Calculator 😊')
495     root.destroy()
496     root.configure(background='#C1E1D2')
497     root.iconbitmap("download.ico")
498
499     image1 = Image.open("images.png")
500     test1 = ImageTk.PhotoImage(image1)
501     label1 = Label(image=test1)
502     label1.image = test1
503     label1.place(x=500, y=150)
504
505     root.protocol('WM_DELETE_WINDOW', display_msg)
506     root.mainloop()
507 #####
508
Table.format() > elif (var4.get() == "Above Normal" > bar())
Version Control Python Packages TODO Python Console Problems Terminal Services Graph Database Console
267:21 CRLF UTF-8 8 spaces* Python 3.10 (user)
Cloudy ENG IN 20:11 10-10-2022
```

PART-2

The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - neo4j.py
- Project:** Test1 Internship > neo4j.py
- Code Editor:** Contains Python code for creating nodes and relations in a Neo4j database. The code uses the neo4j library and includes imports for GraphDatabase, basic_auth, and driver.
- Toolbars and Menus:** Standard PyCharm toolbars and menus like File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help.
- Status Bar:** PEP 8: E302 expected 2 blank lines, found 0. Installing package neo4jdb... Show all (2) 5:19 CRLF UTF-8 4 spaces Python 3.10 (user) 27°C Mostly cloudy ENG IN 10-10-2022 20:25

The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help T1.py - neo4j.py
- Project:** Test1 Internship > neo4j.py
- Code Editor:** Contains Python code for interacting with a Neo4j database. It includes functions for session management, graph deletion, displaying all nodes, and running custom queries.
- Toolbars and Menus:** Standard PyCharm toolbars and menus like File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help.
- Status Bar:** PEP 8: E302 expected 2 blank lines, found 0. 5:19 CRLF UTF-8 4 spaces Python 3.10 (user) 27°C Mostly cloudy ENG IN 10-10-2022 20:26

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** Test1 / Internship
- File:** neo4j.py
- Code Content:**

```
48         lambda tx: tx.run(cq,
49                         ).data())
50     for res in results:
51         print(res)
52     def exit():
53         driver.close()
54     while True:
55         print('''
56             *****
57             | 😊 Welcome to Graph Database Neo4j! 😊 |
58             | You can do the following operations: |
59             | [1] Create a Node |
60             | [2] Create a Relationship for existing nodes |
61             | [3] Delete Graph |
62             | [4] Display all nodes |
63             | [5] Return some data from graph |
64             | [6] Exit |
65             *****
66         ''')
67         option = int(input("Enter an option: "))
68         if option==1:create_node()
69         elif option==2:create_rel()
70         elif option == 3:delete_graph()
71         elif option == 4:displayall()
72         elif option==5:ret()
73         elif option==6:
74             print("Thank you")
75             exit()
76             break
77         else:
78             print("Enter a valid option!")
```

- Toolbars and Status Bar:** Version Control, Python Packages, TODO, Python Console, Problems, Terminal, Services, Graph Database Console. Status bar shows: PEP 8: E302 expected 2 blank lines, found 0. Date: 10-10-2022. Time: 20:26.

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** Test1 / Internship
- File:** neo4j.py
- Code Content:**

```
62     | [4] Display all nodes |
63     | [5] Return some data from graph |
64     | [6] Exit |
65     *****
66     ''')
67     option = int(input("Enter an option: "))
68     if option==1:create_node()
69     elif option==2:create_rel()
70     elif option == 3:delete_graph()
71     elif option == 4:displayall()
72     elif option==5:ret()
73     elif option==6:
74         print("Thank you")
75         exit()
76         break
77     else:
78         print("Enter a valid option!")
```

- Toolbars and Status Bar:** Version Control, Python Packages, TODO, Python Console, Problems, Terminal, Services, Graph Database Console. Status bar shows: PEP 8: E302 expected 2 blank lines, found 0. Date: 10-10-2022. Time: 20:26.

8. RESULTS

PART-1

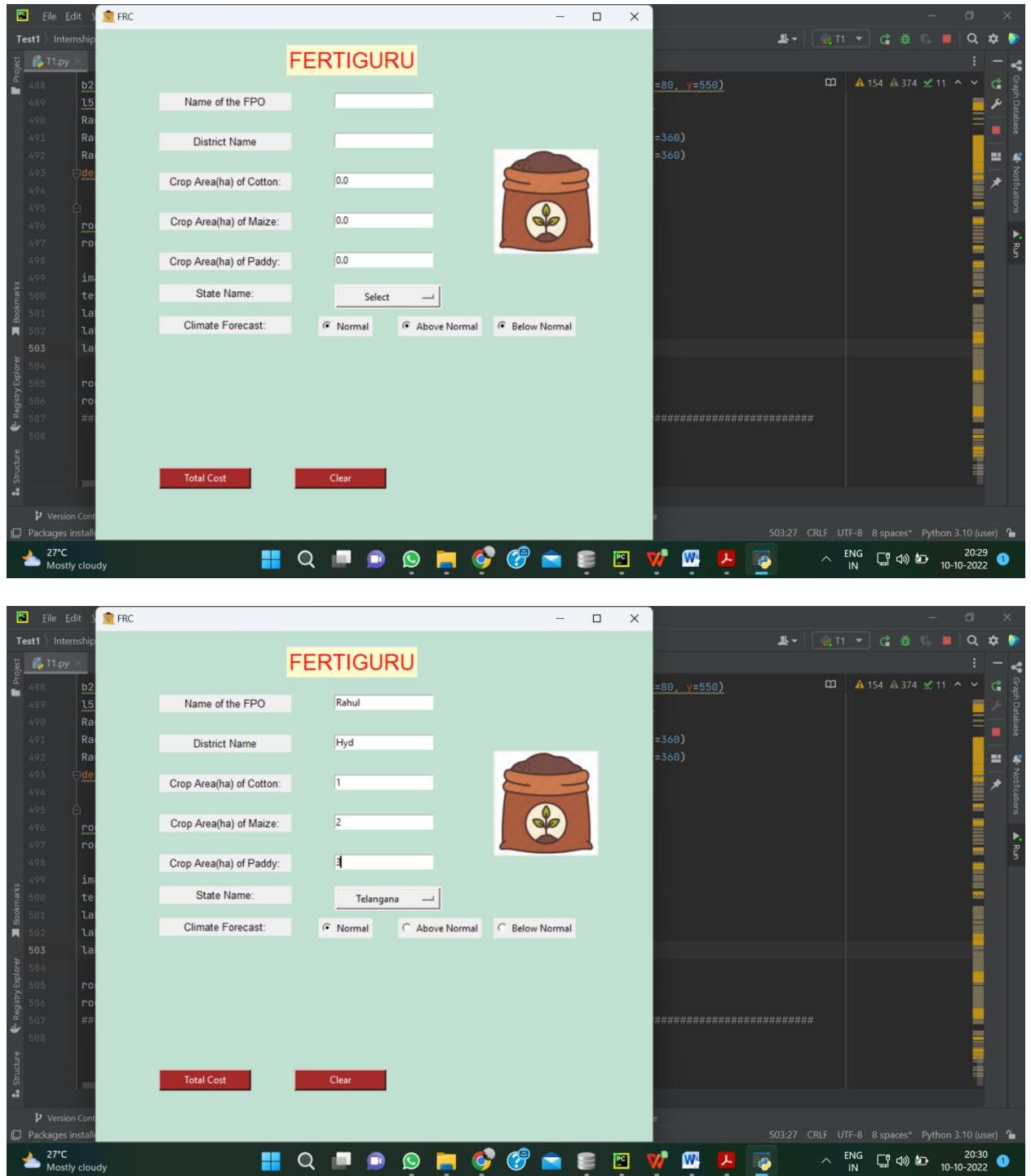


Fig 2. Fertilizer calculator



GREEN SHOWS PROFIT...

SNo	Items	Cotton	Paddy	Maize
1	Operational Cost	54321	168737	101368
2	Human Labour	26734	74487	38720
3	Animal Labour	6458	1999	7916
4	Machine Labour	6558	38740	22856
5	Seed	4720	7830	10752
6	Fertilizer & Manure	4861	26416	13068
7	Insecticides	3219	10757	4858
8	Irrigation Charges	219	3794	446
9	Crop Insurance	0	0	0
10	Miscellaneous	97	395	180

Profit for Cotton(1.0ha):Rs21161.0
 Profit for Paddy(3.0ha):Rs61039.0
 Profit for Maize:(2.0ha):Rs23877.0

Fig 3 Visualization tool showing cost of cultivation and Net Returns

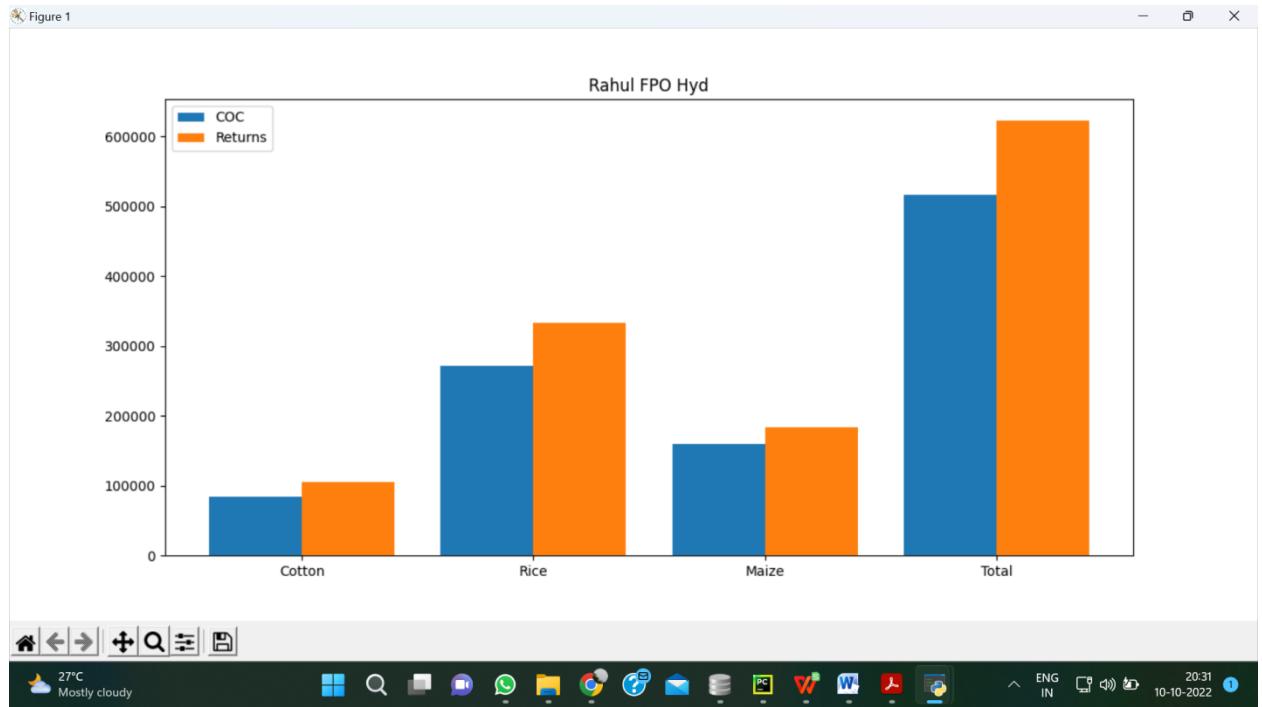
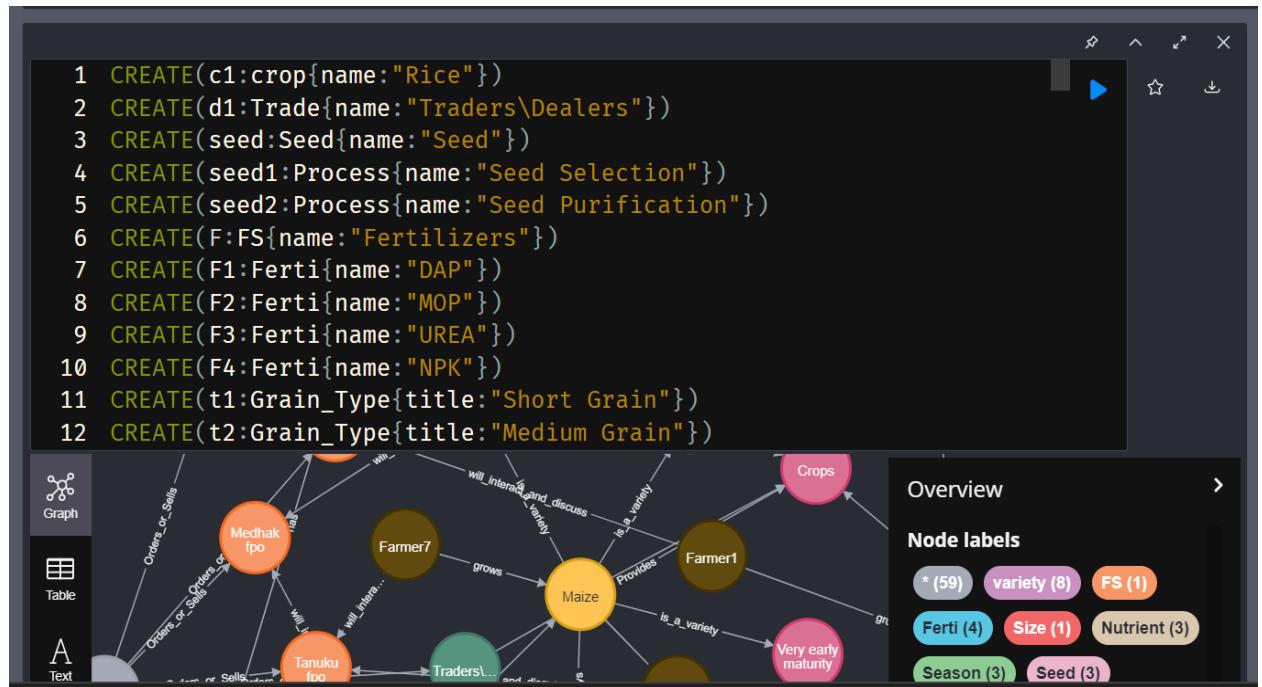


Fig 4. Visualization tool showing cost of cultivation and net returns for overall farm

PART-2



9. CONCLUSION & FUTURE SCOPE

The Yield and profit Prediction can accurately provide the values which helps farmers to grow these three crops and allocate the land accurately to maximize the profit. This tool will also help the FPOS to use the fertilizers calculate which can help in application of correct amounts to prevent the over usage or less usage. Excess application fertilizers will result in environmental pollution and also increases cost of cultivation

The tool also predicts the profit for the crops cultivated. In future we can integrate with other IT tools currently in operation such as sending message to farmers on when to apply fertilizers and other details. The tool can be scalable to several location by linking it with AI and big data tools which automatically gather information from the IMD(New Delhi) and other Agro sites. This will be an out to out interactive tool which will be beneficial with some more future Technology Developments

10. REFERENCES

- [1] Raju KV, Kumar R, Vikraman S, Moses Shyam D, Srikanth R, Kumara Charyulu D and Wani SP. 2017. Farmer Producer Organization in Andhra Pradesh: A Scoping Study. Rythu Kosam Project. Research Report IDC-16. Patancheru 502 324. Telangana, India: International Crops Research Institute for the Semi-Arid Tropics. 160 pp. ISBN 978-92-9066-592-2
- [2] Rao KPC, Dakshina Murthy K, Dhulipala R, Bhagyashree SD, Gupta MD, Sreepada S, Whitbread AM. 2019. Delivering climate risk information to farmers at scale: the Intelligent Agricultural Systems Advisory Tool (ISAT). CCAFS Working Paper no. 243. Wageningen, the Netherlands: CGIAR Research Program on Climate Change, Agriculture and Food Security (CCAFS). Available online at: www.ccafs.cgiar.org
- [3] Srikanth R and Alexaander K 2018. Automated plant disease diagnosis using innovative android App (Plantix) for farmers in Indian state of Andhra Pradesh. International Congress for Plant Pathologists: Boston, USA
- [4] Indev consultancy 2022. Automation for Farmer Producer Organization, accessed 10 October 2022, <https://www.indevconsultancy.com/>
- [5] Kehti Buddy 2022, Software to increase revenue from farm business, accessed 10 October 2022, <https://khetibuddy.com/>
- [6] FarmERP, 2022. Digital Agriculture with the Core of Sustainability. accessed 10 October 2022. <https://www.farmerp.com/>