VELS

INSTITUTE OF SCIENCE, TECHNOLOGY & ADVANCED STUDIES (VISTAS)
(Deemed to be University Estd. u/s 3 of the UGC Act, 1956)
PALLAVARAM - CHENNAI
ACCREDITED BY NAAC WITH 'A' GRADE
Marching Beyond 30 Years Successfully
INSTITUTION WITH UGC 12B STATUS

SCHOOL OF COMPUTING SCIENCES

DEPARTMENT OF COMPUTER APPLICATIONS

INTELLIGENT CROP RECOMMENDATION SYSTEM

A Project Report

Submitted to the VISTAS in partial fulfilment for the award of the degree of

MASTER OF COMPUTER APPLICATION

BY

Name : RAJESH.C
Reg No : 22304233

<u>**Under the guidance of,**</u>

Dr. LIPSA NAYAK, M.C.A., M.Phil., Ph.D. Assistant professor

MAY 2024

VELS

INSTITUTE OF SCIENCE, TECHNOLOGY & ADVANCED STUDIES (VISTAS)
(Deemed to be University Estd. u/s 3 of the UGC Act, 1956)
PALLAVARAM - CHENNAI
ACCREDITED BY NAAC WITH 'A' GRADE
Marching Beyond 30 Years Successfully
INSTITUTION WITH UGC 12B STATUS

SCHOOL OF COMPUTING SCIENCES

DEPARTMENT OF COMPUTER APPLICATIONS

BONAFIDE CERTIFICATE

This is to certify that the Main Project entitled "**INTELLIGENT CROP RECOMMENDATION SYSTEM"** is the original record by **RAJESH. C**, **22304233**, under my guidance and supervision for the partial fulfilment of award of degree of MASTER OF COMPUTER APPLICATION, as per syllabus prescribed by the VISTAS.

GUIDE **HEAD OF THE DEPARTMENT**

Submitted for the Viva-Voce examination held on ……………………… at VISTAS Pallavaram, Chennai.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

01-03-2024

TO WHOMSOEVER IT MAY CONCERN

Dear Sir/Madam,

Sub: Acceptance Letter.

Greetings from **Magic Bus India Foundation**, this letter is to acknowledge the Internship and Project acceptance of **Mr. Rajesh.C – 22304233** in Medavakkam Center from February 2024 for the duration of three month. Project completion certificate will be issued after the successful completion of the project work.

With Regards

District Livelihood Manager

Magic Bus India Foundation

PLOT NO.1, INDIAN UNITED COLONY, V.G.P. PUSHPA NAGAR, MEDAVAKKAM, CHENNAI – 600 100.

ACKNOWLEDGEMENT

“Let the beauty of the lord fall on us and establish the work of our hands”. At the outset, I thank the ALMIGHTY GOD for his abundant blessings and for giving me the opportunity to carry out this project successfully.

I deeply wish to express my sincere thanks to **Dr. ISHARI K. GANESH** M.Com., B.L., Ph.D., the founder and chancellor of VISTAS, **Dr. JOTHI MURUGAN**, Pro-Chancellor (P & D) VISTAS, **Dr. ARTHI GANESH,** Pro-Chancellor (Academics), **Dr. PREETHA GANESH,** Vice-president, Vels Group of institutions.

I extend my thanks to **Dr. S. SRIMAN NARAYANAN** Th Vice-Chancellor, VISTAS for providing me necessary facilities. I wish to extend my heartfelt and sincere thanks to **Dr. M. BHASKARAN**, Pro Vice-Chancellor, VISTAS and sincere thanks to **Dr. P. SARAVANAN,** Registrar.

I extend my thanks to **Dr. A. UDHAYA KUMAR**, Controller of Examinations, VISTAS. I extend my reverential gratitude to **Dr. P. MAGESH KUMAR,** Director, School of Computing Sciences and **Dr. R. PRIYA ANAND MCA, M.Phil., Ph.D.**, and Head of Department for encouraging me to complete my work. I extend my deep sense of gratitude and sincere thanks to my project supervisor of **DR. LIPSA NAYAK M.C.A., M.Phil., Ph.D., Assistant Professor** for helping me with her support, motivation and guidance throughout this research. Her valuable guidance and inspiration are the key factors that enabled me to complete this research successfully.

I sincerely express my gratitude’s to **MY PARENTS, FRIENDS** and our **FACULTY MEMBERS** for their continuous prayers, supports and constant encouragement to reach the heights of success.

Once again, I thank the GOD almighty with whose profound blessings this work has been completed successfully.

DECLARATION

I am, RAJESH. C (**22304233)** declare that the Main Project entitled "**INTELLIGENT CROP RECOMMENDATION SYSTEM**" is a record of originalwork done by me in partial fulfillment of the requirements for the award of the degree of Master of Computer Application under the guidance of **Dr. LIPSA NAYAK, M.C.A, M.Phil., Ph.D. Assistant professor** for the academic year 2024. This project work has not formed the basis for the award of any degree.

(Signature of the Candidate)

ABSTRACT

As we are aware of the fact that, most of Indians have agriculture as their occupation. Farmers usually have the mind-set of planting the same crop, using more fertilizers and following the public choice. By looking at the past few years, there have been significant developments in how machine learning can be used in various industries and research. So, we have planned to create a system where machine learning can be used in agriculture for the betterment of farmers. India is an Agricultural Country and its economy largely based upon crop productivity. So we can say that agriculture can be pillar of all business in our country. Selecting of always crop is very important in the agriculture planning. Many researchers studied guess of yield rate of crop, guess of weather, soil categorizing and crop classification for agriculture planning using machine learning techniques.

Many changes are required in the agriculture department to improve changes in our Indian economy. We can improve agriculture by using machine learning system which are applied simply on farming sector. Along with all advances in the machines and technologies used in farming, functional information about different matters also plays a significant role in it. The concept of this paper is to implement the crop selection method so that this method helps in solving many agriculture problems. This enhances our Indian wealth by maximizing the yield rate of crop production. In our project crop is predicted by algorithm namely Recurrent Neural Network (RNN) as proposed and Random Forest (RF) as existing and its accuracy is calculated and compared with other algorithms.

TABLE OF CONTENTS

INTRODUCTION

1.1 Company Profile

- **Company Name:** Magic Bus USA
- **Location**: Based in the United States, with operations supporting Magic Bus India
- **Industry**: Non-profit, Education, Youth Development
- **Business Type**: Non-profit organization
- **Founded**: 2011 (as part of the Magic Bus India Foundation)
- **Scope**: Supports youth development and education programs in India
- **Youth Mentorship:** Providing mentorship to guide youth through educational and career pathways.
- **Contact Information**
  - **Address**: 1222 Pine Street, San Francisco, CA 94109, USA
  - **Phone**: +1 (415) 915-7818
  - **Email:** usa@magicbusindia.org
  - **Website**: www.magicbususa.org

1.2 Project Profile

1. **Project overview**
   - **Project Name: INTELLIGENT CROP RECOMMENDATION SYSTEM**
   - **Project Type: Research and Development**
   - **Start Date: Feb 2024**
   - **End Date: Apr 2024**
2. **Project Purpose and Objectives**

Purpose: To develop a machine learning-based system for intelligent crop selection that helps Indian farmers optimize crop choices, improve yield rates

Objectives :

1. Implement a Recurrent Neural Network (RNN) model for crop prediction.
2. Compare the accuracy of RNN with the existing Random Forest (RF) model.
3. Provide recommendations for optimal crop selection based on various factors like soil type, weather patterns, and past crop yield.
4. Create a user-friendly interface for farmers to use the system easily.

3. Project Scope

- **Scope**: The project will focus on developing and testing the machine learning models, collecting relevant agricultural data, and creating a user interface for farmers. The geographical scope will be limited to selected states during the pilot phase, with plans to expand nationally if successful.
- **Out-of-Scope:** The project does not include direct agricultural activities like land preparation, planting, or harvesting.

4. Project Risks and Mitigation

- **Risks:**
    - Insufficient data for model training
    - Resistance from farmers to adopt new technology
    - Accuracy of predictions not meeting expectations
- **Mitigation Strategies:**
    - Collaborate with agricultural institutions for data collection and validation.
    - Conduct workshops and training sessions to encourage adoption among farmers.
    - Continuously improve model accuracy through iterative development and testing.

5. Project Outcomes and Measurements

- **Expected Outcomes**
    - A machine learning-based system that accurately predicts optimal crop choices.
    - Increased agricultural productivity and reduced crop failure rates.
- **Measurement Metrics**
    - Accuracy of Recurrent Neural Network (RNN) and Random Forest (RF) models for crop selection.
    - Reduction in crop failure rates.

SYSTEM ANALYSIS

2.1 Existing System

Random Forest (RF):

Random forests or **random decision forests** are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

Decision trees are a popular method for various machine learning tasks. Tree learning "come[s] closest to meeting the requirements for serving as an off-the-shelf procedure for data mining", say Hastie *et al.*, "because it is invariant under scaling and various other transformations of feature values, is robust to inclusion of irrelevant features, and produces inspectable models. However, they are seldom accurate".

In particular, trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets, i.e. have low bias, but very high variance. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model.

Forests are like the pulling together of decision tree algorithm efforts. Taking the teamwork of many trees thus improving the performance of a single random tree. Though not quite similar, forests give the effects of a K-fold cross validation.

Existing System Disadvantages:

- Highly manual.
- Less cultivation.
- Less profit.
- Maintaining and retrieving the record of Users is difficult.
- No user friendly model.

2.2. Proposed System

Recurrent Neural Network (RNN):

A **Recurrent Neural Network** (**RNN**) is a class of artificial neural networks where connections between nodes form a graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from feed forward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.

The term "recurrent neural network" is used indiscriminately to refer to two broad classes of networks with a similar general structure, where one is finite impulse and the other is infinite impulse. Both classes of networks exhibit temporal dynamic behavior. A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feedforward neural network, while an infinite impulse recurrent network is a directed cyclic graph that can not be unrolled.

Both finite impulse and infinite impulse recurrent networks can have additional stored states, and the storage can be under direct control by the neural network.

The storage can also be replaced by another network or graph, if that incorporates time delays or has feedback loops. Such controlled states are referred to as gated state or gated memory, and are part of memory networks (LSTMs) and gated recurrent units. This is also called Feedback Neural Network (FNN).

Proposed System Advantages:

- Less time consuming.
- Secure process.
- Manual efforts are overcome.
- Best crop can be predicted at anytime.
- Can be implemented in all datasets.
- User friendly model.

SYSTEM REQUIREMENTS

3.1. Hardware Requirements

- **Processor :** i3 processor.
- **Ram:** 4GB RAM.
- **Storage:** 500GB SSD.

3.2 Software Requirements

- **Operating System:** Windows/Linux
- **Web browser:** Edge / Chrome
- **Technology:** Python IDE
- **Languages Used**: Python, HTML, CSS, BOOTSTRAP

3.3 Software Description

Frontend Technology

- **HTML** (HyperText MarkUp language)
  - HTML is the foundational language for creating webpage structure and content. It defines the elements on a webpage, such as headings, paragraphs, images, and links.
- **CSS** (Cascading Style Sheets)
  - CSS is used to style and format HTML elements, allowing for design customization such as colors, fonts, spacing, and layout. CSS can be used to create responsive designs, ensuring the website looks good on various devices.
- **Bootstrap**
  - Bootstrap is a popular CSS framework that provides pre-built components and utilities to create responsive and mobile-friendly websites. It includes a grid system, UI components (like buttons, modals, and forms), and other useful features for rapid development

Backend Technologies

Python

Python is a remarkably powerful dynamic, object-oriented programming language that is used in a wide variety of application domains. It offers strong support for integration with other languages and tools, and comes with extensive standard libraries. To be precise, the following are some distinguishing features of Python:

- Very clear, readable syntax.
- Strong introspection capabilities.
- Full modularity.
- Exception-based error handling.
- High level dynamic data types.
- Supports object oriented, imperative and functional programming styles.
- Embeddable.
- Scalable
- Mature

With so much of freedom, Python helps the user to think problem centric rather than language centric as in other cases.

B. Open CV

Open CV is a library of programming functions for real time computer vision originally developed by Intel and now supported by Willogarage. It is free for use under the open source BSD license. The library has more than five hundred optimized algorithms. It is used around the world, with forty thousand people in the user group. Uses range from interactive art, to mine inspection, and advanced robotics. The library is mainly written in C, which makes it portable to some specific platforms such as Digital Signal Processor. Wrappers for languages such as C, Python, Ruby and Java (using Java CV) have been developed to encourage adoption by a wider audience. The recent releases have interfaces for C++. It focuses mainly on real-time image processing. Open CV is a cross-platform library, which can run on Linux, Mac OS and Windows. To date, Open CV is the best open source computer vision library that developers and researchers can think of.

C. Tesseract

Tesseract is a free software OCR engine that was developed at HP between 1984 and 1994.

HP released it to the community in 2005. Tesseract was introduced at the 1995 UNLV Annual Test OCR Accuracy and is currently developed by Google released under the Apache License. It can now recognize 6 languages, and is fully UTF8 capable. Developers can train Tesseract with their own fonts and character mapping to obtain perfect efficiency.

ABOUT THE SOFTWARE “PYTHON”

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include –

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python - Environment Setup

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

Local Environment Setup

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- Unix (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS
- BeOS
- Amiga
- VMS/OpenVMS
- QNX
- VxWorks
- Psion
- Python has also been ported to the Java and .NET virtual machines

Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python https://www.python.org/

You can download Python documentation https://www.python.org/doc/The documentation is available in HTML, PDF, and PostScript formats.

Installing Python

Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python.

If the binary code for your platform is not available, you need a C compiler to compile the source code manually. Compiling the source code offers more flexibility in terms of choice of features that you require in your installation.

Here is a quick overview of installing Python on various platforms −

Unix and Linux Installation

Here are the simple steps to install Python on Unix/Linux machine.

- Open a Web browser and go to https://www.python.org/downloads/.
- Follow the link to download zipped source code available for Unix/Linux.
- Download and extract files.
- Editing the *Modules/Setup* file if you want to customize some options.
- run ./configure script
- make
- make install

This installs Python at standard location */usr/local/bin* and its libraries at */usr/local/lib/pythonXX* where XX is the version of Python.

Windows Installation

Here are the steps to install Python on Windows machine.

- Open a Web browser and go to https://www.python.org/downloads/.
- Follow the link for the Windows installer *python-XYZ.msi* file where XYZ is the version you need to install.
- To use this installer *python-XYZ.msi*, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.

- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

Macintosh Installation

Recent Macs come with Python installed, but it may be several years out of date. See http://www.python.org/download/mac/ for instructions on getting the current version along with extra tools to support development on the Mac. For older Mac OS's before Mac OS X 10.3 (released in 2003), MacPython is available.

Jack Jansen maintains it and you can have full access to the entire documentation at his website − http://www.cwi.nl/~jack/macpython.html. You can find complete installation details for Mac OS installation.

Setting up PATH

Programs and other executable files can be in many directories, so operating systems provide a search path that lists the directories that the OS searches for executables.

The path is stored in an environment variable, which is a named string maintained by the operating system. This variable contains information available to the command shell and other programs.

The **path** variable is named as PATH in Unix or Path in Windows (Unix is case sensitive; Windows is not).

In Mac OS, the installer handles the path details. To invoke the Python interpreter from any particular directory, you must add the Python directory to your path.

Setting path at Unix/Linux

To add the Python directory to the path for a particular session in Unix –

- **In the csh shell** − type setenv PATH "$PATH:/usr/local/bin/python" and press Enter.
- **In the bash shell (Linux)** − type export PATH="$PATH:/usr/local/bin/python" and press Enter.

- **In the sh or ksh shell** − type PATH="$PATH:/usr/local/bin/python" and press Enter.
- **Note** − /usr/local/bin/python is the path of the Python directory

Setting path at Windows

To add the Python directory to the path for a particular session in Windows −

At the command prompt − type path %path%;C:\Python and press Enter.

Note − C:\Python is the path of the Python directory

Python Environment Variables

Here are important environment variables, which can be recognized by Python −

Sr.No.	Variable & Description
1	**PYTHONPATH** It has a role similar to PATH. This variable tells the Python interpreter where to locate the module files imported into a program. It should include the Python source library directory and the directories containing Python source code. PYTHONPATH is sometimes preset by the Python installer.
2	**PYTHONSTARTUP** It contains the path of an initialization file containing Python source code. It is executed every time you start the interpreter. It is named as .pythonrc.py in Unix and it contains commands that load utilities or modify PYTHONPATH.
3	**PYTHONCASEOK** It is used in Windows to instruct Python to find the first case-insensitive match in an import statement. Set this variable to any value to activate it.
4	**PYTHONHOME** It is an alternative module search path. It is usually embedded in the PYTHONSTARTUP or PYTHONPATH directories to make switching module

	libraries easy.

Running Python

There are three different ways to start Python −

Interactive Interpreter

You can start Python from Unix, DOS, or any other system that provides you a command-line interpreter or shell window.

Enter **python** the command line.

Start coding right away in the interactive interpreter.

```
$python # Unix/Linux
or
python% # Unix/Linux
or
C:> python # Windows/DOS
```

Here is the list of all the available command line options −

Sr.No.	Option & Description
1	**-d** It provides debug output.
2	**-O** It generates optimized bytecode (resulting in .pyo files).
3	**-S** Do not run import site to look for Python paths on startup.

4	**-v** verbose output (detailed trace on import statements).
5	**-X** disable class-based built-in exceptions (just use strings); obsolete starting with version 1.6.
6	**-c cmd** run Python script sent in as cmd string
7	**File** run Python script from given file

Script from the Command-line

A Python script can be executed at command line by invoking the interpreter on your application, as in the following −

```
$python script.py # Unix/Linux

or

python% script.py # Unix/Linux

or

C: >python script.py # Windows/DOS
```

Note − Be sure the file permission mode allows execution.

Integrated Development Environment

You can run Python from a Graphical User Interface (GUI) environment as well, if you have a GUI application on your system that supports Python.

- **Unix** − IDLE is the very first Unix IDE for Python.
- **Windows** − PythonWin is the first Windows interface for Python and is an IDE with a GUI.
- **Macintosh** − The Macintosh version of Python along with the IDLE IDE is available from the main website, downloadable as either MacBinary or BinHex'd files.

If you are not able to set up the environment properly, then you can take help from your system admin. Make sure the Python environment is properly set up and working perfectly

Note − All the examples given in subsequent chapters are executed with Python 2.4.3 version available on CentOS flavor of Linux.

We already have set up Python Programming environment online, so that you can execute all the available examples online at the same time when you are learning theory. Feel free to modify any example and execute it online.

Panda

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc. In this tutorial, we will learn the various features of Python Pandas and how to use them in practice.

Audience

This tutorial has been prepared for those who seek to learn the basics and various functions of Pandas. It will be specifically useful for people working with data cleansing and analysis. After completing this tutorial, you will find yourself at a moderate level of expertise from where you can take yourself to higher levels of expertise.

Prerequisites

You should have a basic understanding of Computer Programming terminologies. A basic understanding of any of the programming languages is a plus. Pandas library uses most of the functionalities of NumPy. It is suggested that you go through our tutorial on NumPy before proceeding with this tutorial. You can access it from Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Standard Python distribution doesn't come bundled with Pandas module.

NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.

Audience

This tutorial has been prepared for those who want to learn about the basics and various functions of NumPy. It is specifically useful for algorithm developers. After completing this tutorial, you will find yourself at a moderate level of expertise from where you can take yourself to higher levels of expertise.

Prerequisites

You should have a basic understanding of computer programming terminologies. A basic understanding of Python and any of the programming languages is a plus.umPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

Operations using NumPy

Using NumPy, a developer can perform the following operations –

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

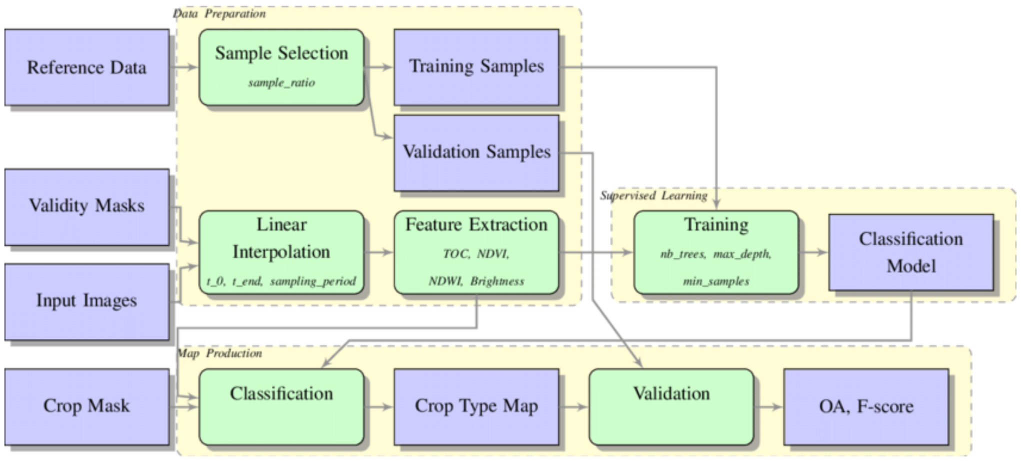
NumPy – A Replacement for MatLab

NumPy is often used along with packages like **SciPy** (Scientific Python) and **Mat−plotlib** (plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.

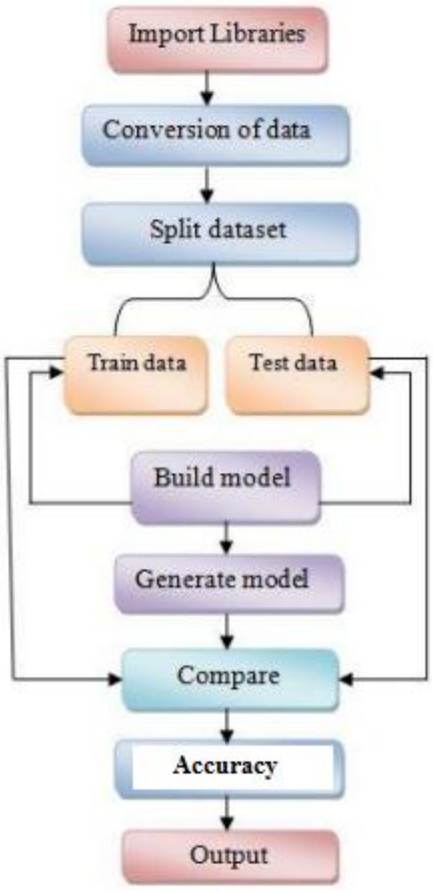
It is open source, which is an added advantage of NumPy.tandard Python distribution doesn't come bundled with NumPy module. A lightweight alternative is to install NumPy using popular Python package installer, **pip**

SYSTEM DESIGN

4.1. System Architecture



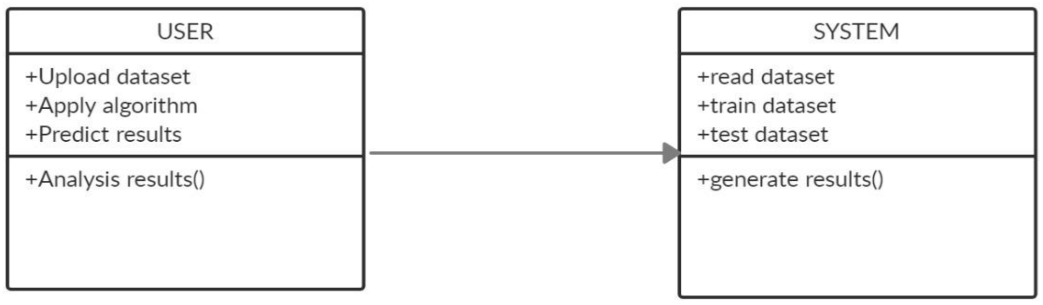
4.2. Data Flow Diagram



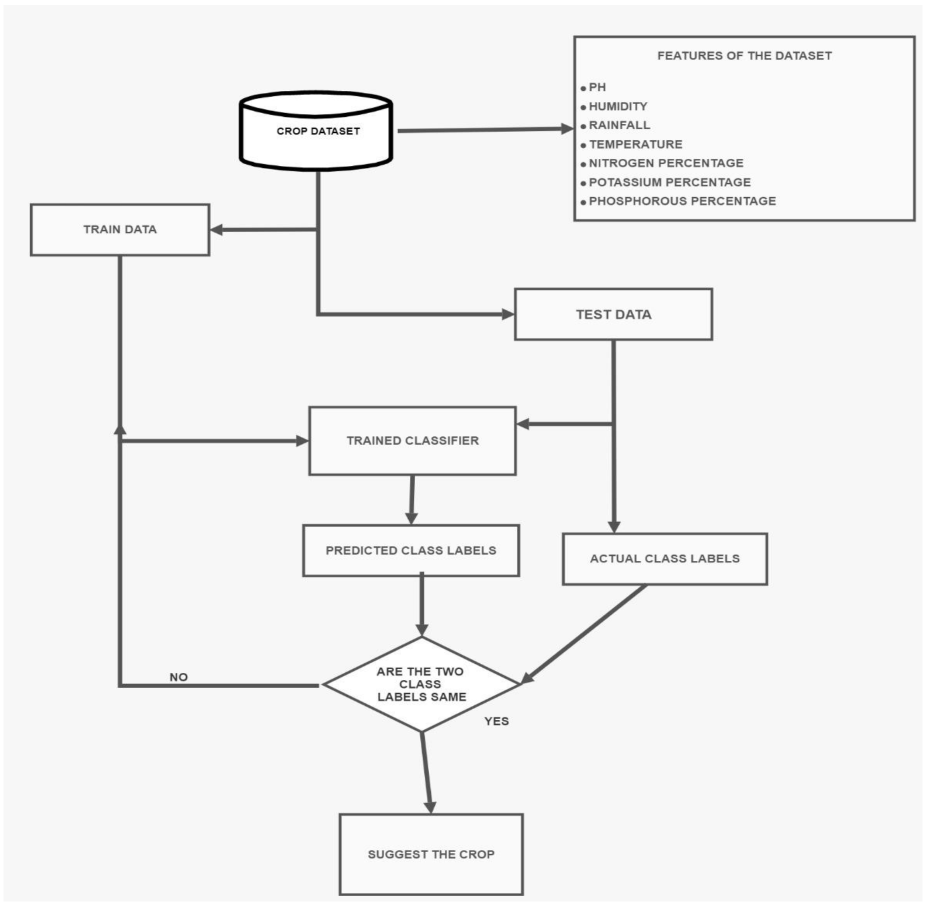
4.3. DataSet Design

N	P	K	temperature	humidity	ph	rainfall	label

4.4. UML Diagram



4.5. ER Diagram


FEATURES OF THE DATASET
PH
HUMIDITY
RAINFALL
TEMPERATURE
NITROGEN PERCENTAGE
POTASSIUM PERCENTAGE
PHOSPHOROUS PERCENTAGE
CROP DATASET
TRAIN DATA
TEST DATA
TRAINED CLASSIFIER
PREDICTED CLASS LABELS
ACTUAL CLASS LABELS
ARE THE TWO CLASS LABELS SAME
NO
YES
SUGGEST THE CROP

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

5.1. Software Testing

5.1.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

5.1.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components

5.1.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

5.1.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

5.1.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

5.1.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

5.1.7 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

SYSTEM IMPLEMENTATION

6.1. System Description

1. Overview

- **Purpose**: This system is designed to assist farmers in selecting the most suitable crops for cultivation based on a variety of factors, including soil composition, weather conditions, and past crop performance. The ultimate goal is to help farmers improve their yields, reduce waste, and make more informed agricultural decisions.
- **Scope**: The system is intended for use by farmers, agricultural consultants, and government agencies in India. It can be expanded to other regions or customized to focus on specific crops as needed.

2. System Architecture

- **High-Level Structure**: The system comprises three key components: a backend engine for machine learning, a data storage system for storing agricultural data, and a frontend user interface for users to interact with the system. These components are designed to work together seamlessly to deliver accurate crop predictions and recommendations.

- **Subsystems**
  - **Machine Learning Engine:** This is the core of the system, where the machine learning algorithms are implemented. It includes various models, such as Recurrent Neural Networks (RNN) and Random Forest (RF), to predict optimal crops based on input data.
  - **Data Storage**: A database subsystem is used to store soil data, weather patterns, historical crop yields, and other relevant information. It serves as the backbone for data retrieval and processing.

- **User Interface**: A web-based or mobile-friendly interface through which farmers and other stakeholders interact with the system. It provides an intuitive way to input data and retrieve crop recommendations.

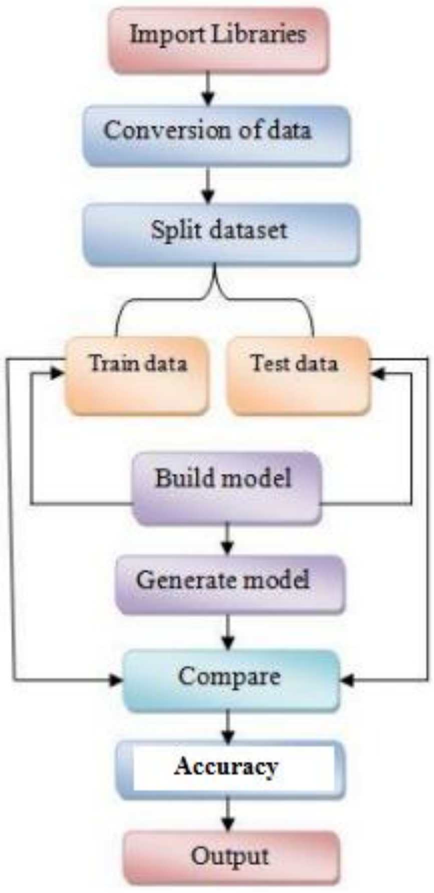
3. Hardware Components

- **Servers:** The system requires servers to host the backend, database, and machine learning models. These servers should have adequate processing power, memory, and storage to handle large datasets and complex computations.
- **Network Infrastructure:** A robust network infrastructure ensures seamless communication between the frontend and backend components. Load balancers are used to distribute traffic evenly, and firewalls are implemented for security.
- **Redundancy and Backup**: The system employs redundancy in hardware to ensure high availability. Backup systems are in place to prevent data loss and facilitate disaster recovery**.**

4. Software Components

- **Backend Environment:** The backend is built using Python, with frameworks like Flask or Django for building web applications. It integrates with machine learning libraries such as TensorFlow or PyTorch to perform predictions.
- **Frontend Environment:** The frontend is developed using HTML, CSS, Bootstrap, and JavaScript frameworks. It communicates with the backend through RESTful APIs, allowing for dynamic interactions and real-time data updates.

6.2. System Flow



6.3. Modules Description

1. Input dataset
2. Analysis of size of data set.
3. Oversampling.
4. Training and Testing.
5. Apply algorithms.
6. Predict results.

1. Input dataset:

Dataset can be taken from online data source provider from the internet sources. We have to collect a huge dataset in volume so as to predict the accuracy in an efficient manner.

2. Analysis of data set:

Here the analysis if dataset takes place. The size of data is taken into consideration for the data process.

3. Oversampling (Using SMOTE):

We have created a detailed history of all details dataset related to crops such as temperature, Ph, climate etc.

4.Training and Testing Subset:

As the dataset is imbalanced, many classifiers show bias for majority classes. The features of minority class are treated as noise and are ignored. Hence it is proposed to select a sample dataset.

5. **Applying algorithm:**

Following are the classification algorithms used to test the sub-sample dataset.

a. Random Forest (RF)

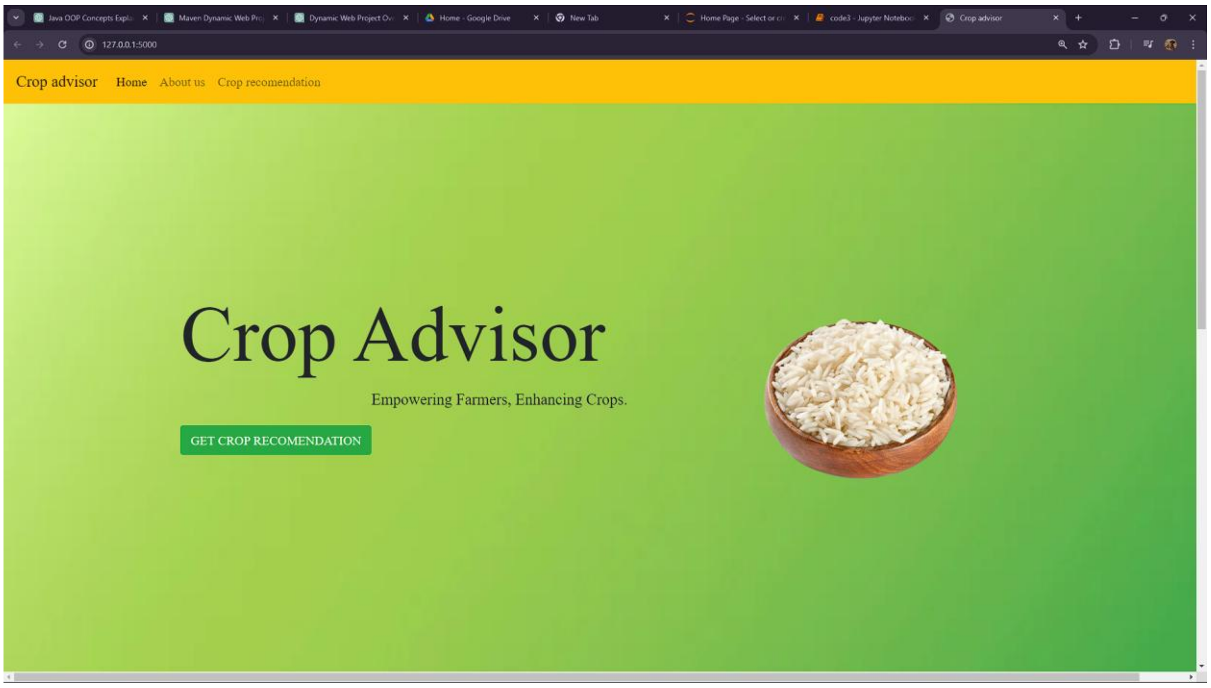
b. Recurrent Neural Network (RNN)

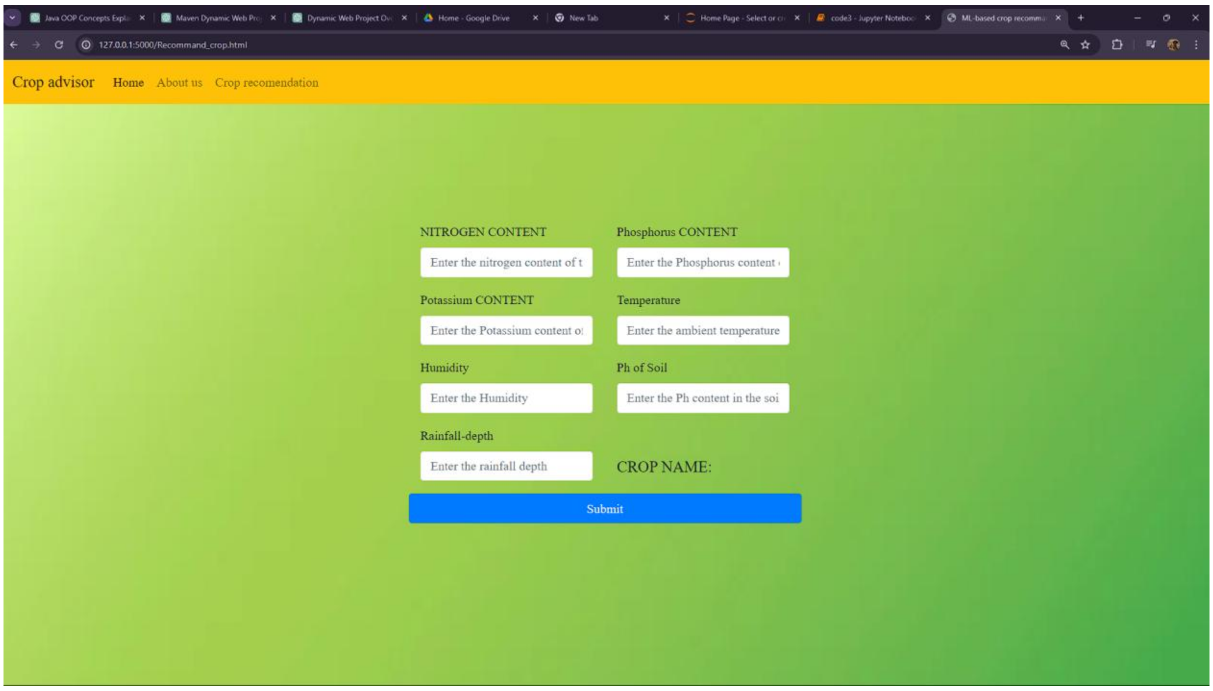
6. **Predicting results:**

The test subset is applied on the trained model.The metrices used is accuracy. The ROC Curve is plotted and the desirable results are achieved.

APPENDICES

7.1. Screenshots



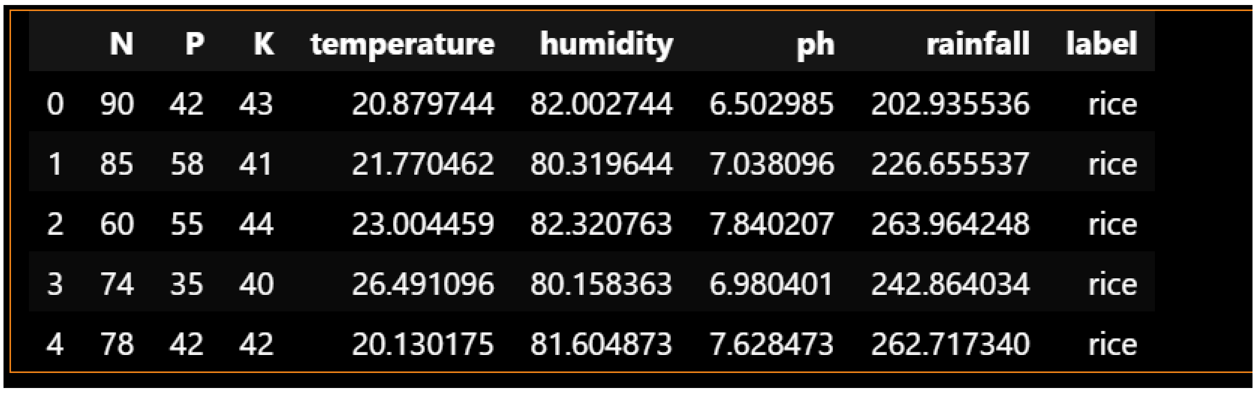


7.2. Source code

```
import numpy as np
import pandas as pd


import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')


import ipywidgets
from ipywidgets import interact
```

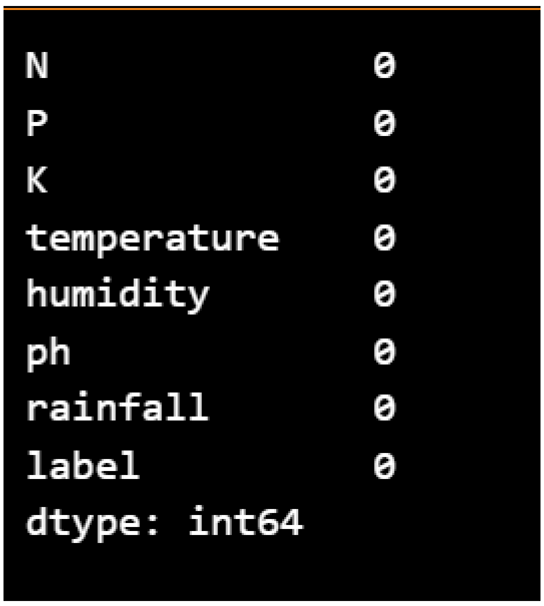
```
data = pd.read_csv("dataset.csv")


print("Shape of the Dataset :", data.shape)
```

```
data.head()
```



```
    N   P   K  temperature   humidity        ph    rainfall  label
0  90  42  43    20.879744  82.002744  6.502985  202.935536   rice
1  85  58  41    21.770462  80.319644  7.038096  226.655537   rice
2  60  55  44    23.004459  82.320763  7.840207  263.964248   rice
3  74  35  40    26.491096  80.158363  6.980401  242.864034   rice
4  78  42  42    20.130175  81.604873  7.628473  262.717340   rice
```

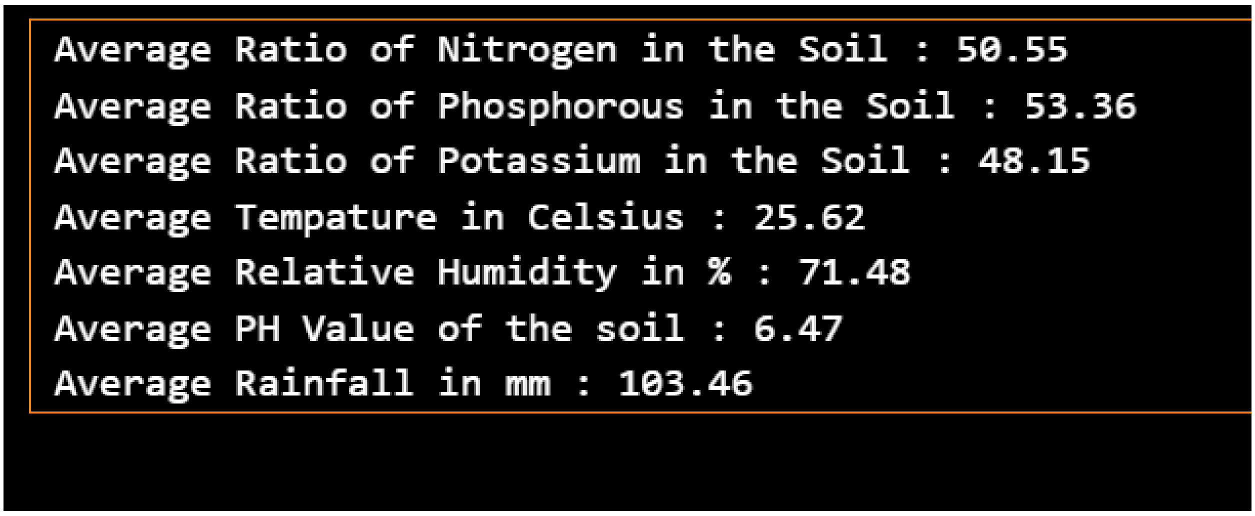
```
data.isnull().sum()
```



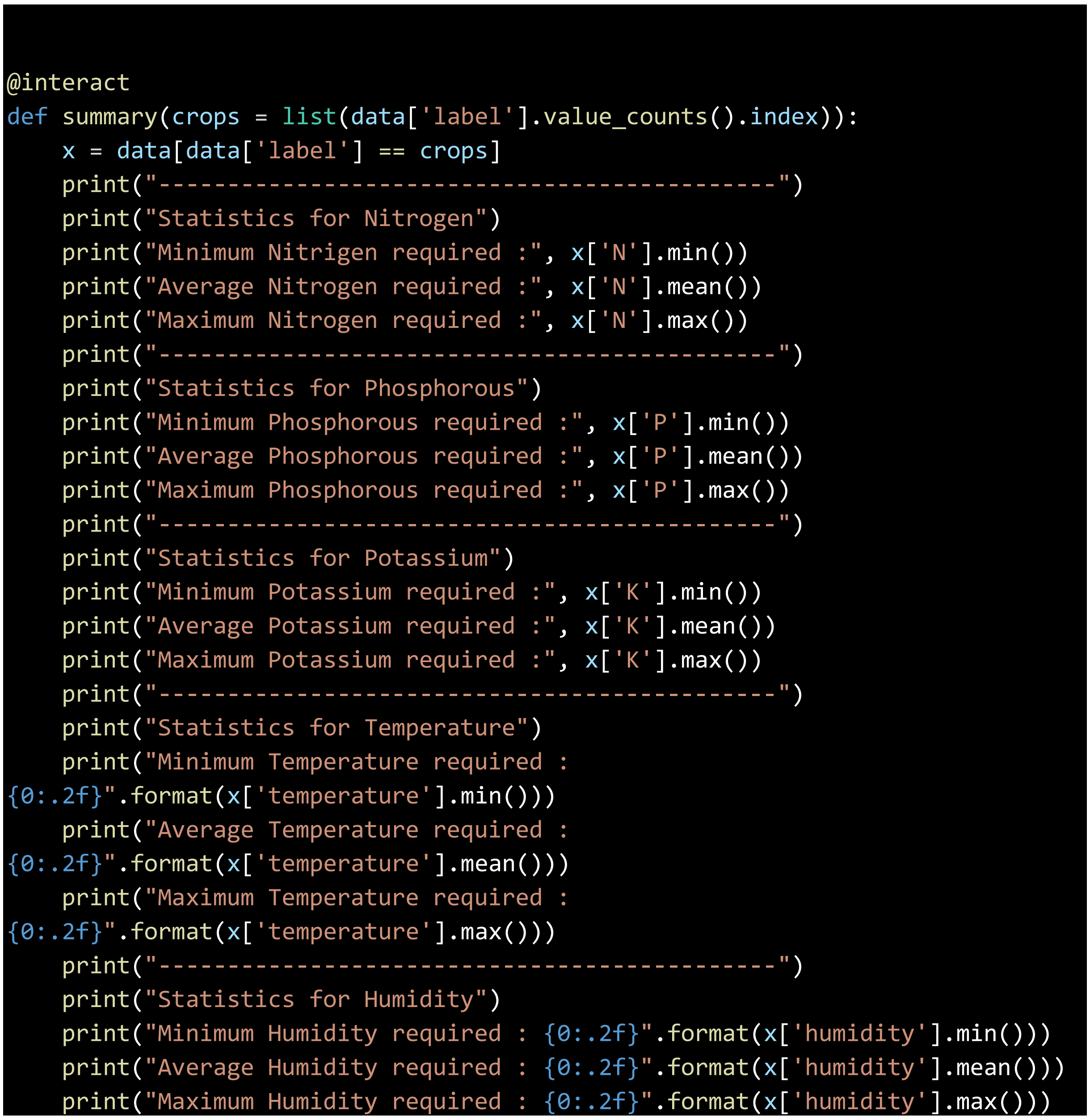
```
N              0
P              0
K              0
temperature    0
humidity       0
ph             0
rainfall       0
label          0
dtype: int64
```

```
data['label'].value_counts()
```

```
print("Average Ratio of Nitrogen in the Soil :
{0:.2f}".format(data['N'].mean()))
print("Average Ratio of Phosphorous in the Soil :
{0:.2f}".format(data['P'].mean()))
print("Average Ratio of Potassium in the Soil :
{0:.2f}".format(data['K'].mean()))
print("Average Tempature in Celsius :
{0:.2f}".format(data['temperature'].mean()))
print("Average Relative Humidity in % :
{0:.2f}".format(data['humidity'].mean()))
print("Average PH Value of the soil : {0:.2f}".format(data['ph'].mean()))
print("Average Rainfall in mm : {0:.2f}".format(data['rainfall'].mean()))
```



```
Average Ratio of Nitrogen in the Soil : 50.55
Average Ratio of Phosphorous in the Soil : 53.36
Average Ratio of Potassium in the Soil : 48.15
Average Tempature in Celsius : 25.62
Average Relative Humidity in % : 71.48
Average PH Value of the soil : 6.47
Average Rainfall in mm : 103.46
```



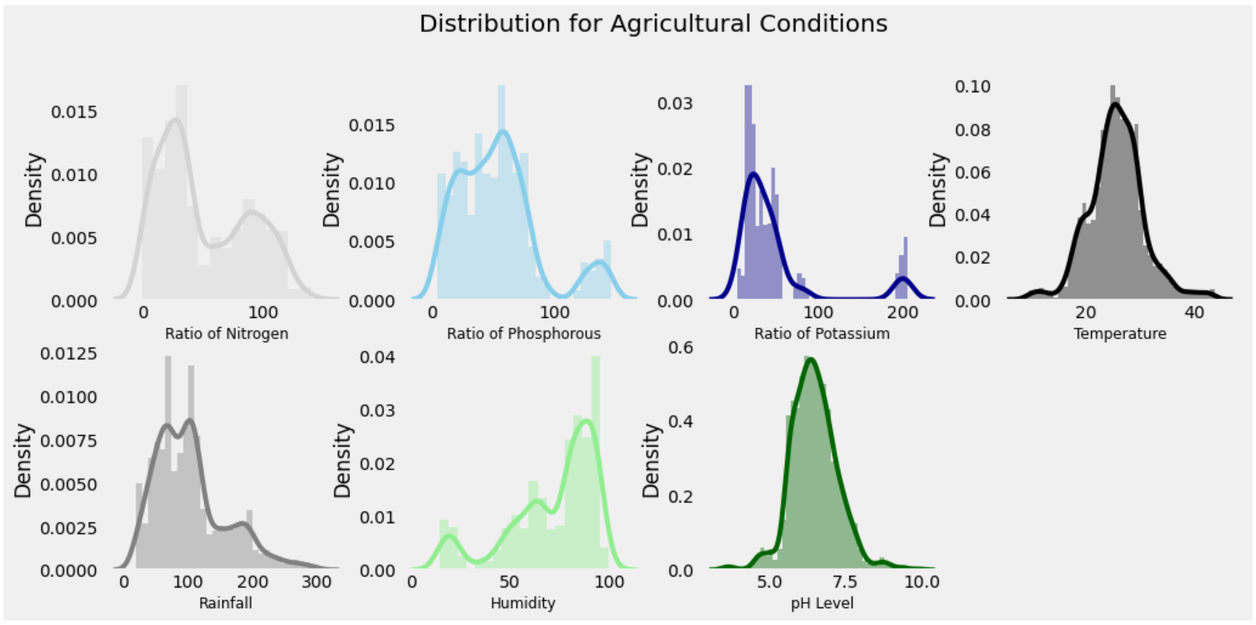
```
@interact
def summary(crops = list(data['label'].value_counts().index)):
    x = data[data['label'] == crops]
    print("---------------------------------------------")
    print("Statistics for Nitrogen")
    print("Minimum Nitrigen required :", x['N'].min())
    print("Average Nitrogen required :", x['N'].mean())
    print("Maximum Nitrogen required :", x['N'].max())
    print("---------------------------------------------")
    print("Statistics for Phosphorous")
    print("Minimum Phosphorous required :", x['P'].min())
    print("Average Phosphorous required :", x['P'].mean())
    print("Maximum Phosphorous required :", x['P'].max())
    print("---------------------------------------------")
    print("Statistics for Potassium")
    print("Minimum Potassium required :", x['K'].min())
    print("Average Potassium required :", x['K'].mean())
    print("Maximum Potassium required :", x['K'].max())
    print("---------------------------------------------")
    print("Statistics for Temperature")
    print("Minimum Temperature required :
{0:.2f}".format(x['temperature'].min()))
    print("Average Temperature required :
{0:.2f}".format(x['temperature'].mean()))
    print("Maximum Temperature required :
{0:.2f}".format(x['temperature'].max()))
    print("---------------------------------------------")
    print("Statistics for Humidity")
    print("Minimum Humidity required : {0:.2f}".format(x['humidity'].min()))
    print("Average Humidity required : {0:.2f}".format(x['humidity'].mean()))
    print("Maximum Humidity required : {0:.2f}".format(x['humidity'].max()))
    print("---------------------------------------------")
    print("Statistics for PH")
    print("Minimum PH required : {0:.2f}".format(x['ph'].min()))
    print("Average PH required : {0:.2f}".format(x['ph'].mean()))
    print("Maximum PH required : {0:.2f}".format(x['ph'].max()))
    print("---------------------------------------------")
    print("Statistics for Rainfall")
    print("Minimum Rainfall required : {0:.2f}".format(x['rainfall'].min()))
    print("Average Rainfall required : {0:.2f}".format(x['rainfall'].mean()))
    print("Maximum Rainfall required : {0:.2f}".format(x['rainfall'].max()))
```

```


@interact
def compare(conditions =
['N','P','K','temperature','ph','humidity','rainfall']):
    print("Average Value for", conditions,"is
{0:.2f}".format(data[conditions].mean()))
    print("----------------------------------------------")
    print("Rice : {0:.2f}".format(data[(data['label'] ==
'rice')][conditions].mean()))
    print("Black Grams : {0:.2f}".format(data[data['label'] ==
'blackgram'][conditions].mean()))
    print("Banana : {0:.2f}".format(data[(data['label'] ==
'banana')][conditions].mean()))
    print("Jute : {0:.2f}".format(data[data['label'] ==
'jute'][conditions].mean()))
    print("Coconut : {0:.2f}".format(data[(data['label'] ==
'coconut')][conditions].mean()))
    print("Apple : {0:.2f}".format(data[data['label'] ==
'apple'][conditions].mean()))
    print("Papaya : {0:.2f}".format(data[(data['label'] ==
'papaya')][conditions].mean()))
    print("Muskmelon : {0:.2f}".format(data[data['label'] ==
'muskmelon'][conditions].mean()))
    print("Grapes : {0:.2f}".format(data[(data['label'] ==
'grapes')][conditions].mean()))
    print("Watermelon : {0:.2f}".format(data[data['label'] ==
'watermelon'][conditions].mean()))
    print("Kidney Beans: {0:.2f}".format(data[(data['label'] ==
'kidneybeans')][conditions].mean()))
    print("Mung Beans : {0:.2f}".format(data[data['label'] ==
'mungbean'][conditions].mean()))
    print("Oranges : {0:.2f}".format(data[(data['label'] ==
'orange')][conditions].mean()))
    print("Chick Peas : {0:.2f}".format(data[data['label'] ==
'chickpea'][conditions].mean()))
    print("Lentils : {0:.2f}".format(data[(data['label'] ==
'lentil')][conditions].mean()))
```

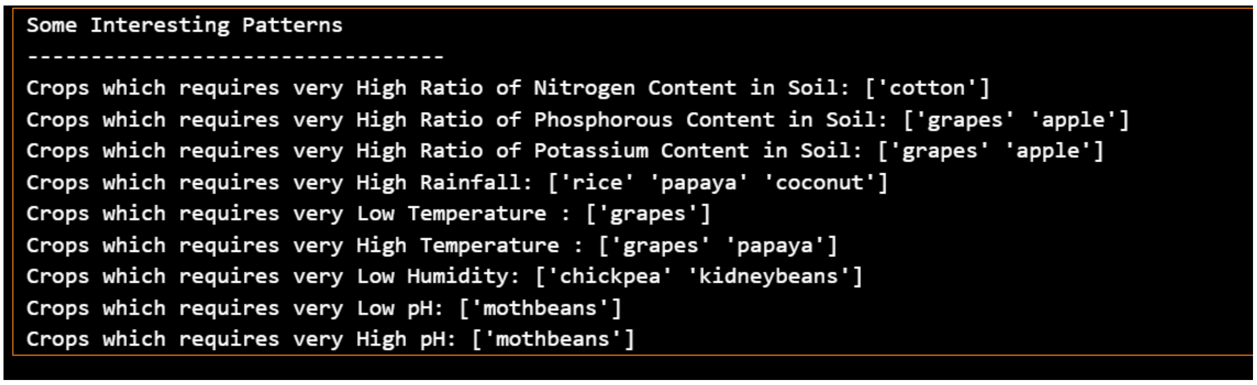
```
    print("Cotton : {0:.2f}".format(data[data['label'] ==
'cotton'][conditions].mean()))
    print("Maize : {0:.2f}".format(data[(data['label'] ==
'maize')][conditions].mean()))
    print("Moth Beans : {0:.2f}".format(data[data['label'] ==
'mothbeans'][conditions].mean()))
    print("Pigeon Peas : {0:.2f}".format(data[(data['label'] ==
'pigeonpeas')][conditions].mean()))
    print("Mango : {0:.2f}".format(data[data['label'] ==
'mango'][conditions].mean()))
    print("Pomegranate : {0:.2f}".format(data[(data['label'] ==
'pomegranate')][conditions].mean()))
    print("Coffee : {0:.2f}".format(data[data['label'] ==
'coffee'][conditions].mean()))
```

```
@interact
def compare(conditions =
['N','P','K','temperature','ph','humidity','rainfall']):
    print("Crops which require greater than average", conditions,'\n')
    print(data[data[conditions] > data[conditions].mean()]['label'].unique())
    print("----------------------------------------------")
    print("Crops which require less than average", conditions,'\n')
    print(data[data[conditions] <= data[conditions].mean()]['label'].unique())
```

```
plt.rcParams['figure.figsize'] = (15, 7)

plt.subplot(2, 4, 1)
sns.distplot(data['N'], color = 'lightgrey')
plt.xlabel('Ratio of Nitrogen', fontsize = 12)
plt.grid()

plt.subplot(2, 4, 2)
sns.distplot(data['P'], color = 'skyblue')
plt.xlabel('Ratio of Phosphorous', fontsize = 12)
plt.grid()

plt.subplot(2, 4, 3)
sns.distplot(data['K'], color ='darkblue')
plt.xlabel('Ratio of Potassium', fontsize = 12)
plt.grid()

plt.subplot(2, 4, 4)
sns.distplot(data['temperature'], color = 'black')
plt.xlabel('Temperature', fontsize = 12)
plt.grid()
plt.subplot(2, 4, 5)
sns.distplot(data['rainfall'], color = 'grey')
plt.xlabel('Rainfall', fontsize = 12)
plt.grid()

plt.subplot(2, 4, 6)
sns.distplot(data['humidity'], color = 'lightgreen')
plt.xlabel('Humidity', fontsize = 12)
plt.grid()

plt.subplot(2, 4, 7)
sns.distplot(data['ph'], color = 'darkgreen')
plt.xlabel('pH Level', fontsize = 12)
plt.grid()

plt.suptitle('Distribution for Agricultural Conditions', fontsize = 20)
plt.show()
```



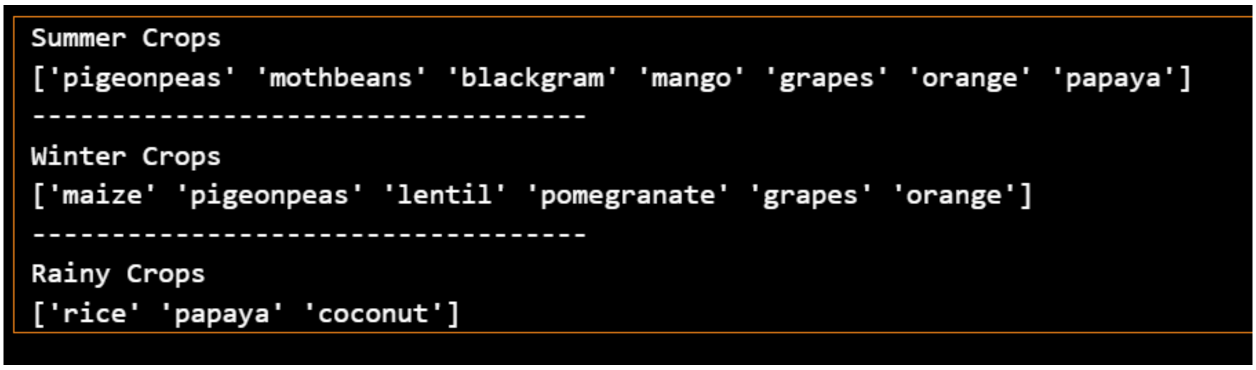
```
print("Some Interesting Patterns")
print("---------------------------------")
print("Crops which requires very High Ratio of Nitrogen Content in Soil:",
data[data['N'] > 120]['label'].unique())
print("Crops which requires very High Ratio of Phosphorous Content in Soil:",
data[data['P'] > 100]['label'].unique())
print("Crops which requires very High Ratio of Potassium Content in Soil:",
data[data['K'] > 200]['label'].unique())
print("Crops which requires very High Rainfall:", data[data['rainfall'] >
200]['label'].unique())
print("Crops which requires very Low Temperature :", data[data['temperature']
< 10]['label'].unique())
print("Crops which requires very High Temperature :", data[data['temperature']
> 40]['label'].unique())
print("Crops which requires very Low Humidity:", data[data['humidity'] <
20]['label'].unique())
print("Crops which requires very Low pH:", data[data['ph'] <
4]['label'].unique())
print("Crops which requires very High pH:", data[data['ph'] >
9]['label'].unique())
```



```
Some Interesting Patterns
---------------------------------
Crops which requires very High Ratio of Nitrogen Content in Soil: ['cotton']
Crops which requires very High Ratio of Phosphorous Content in Soil: ['grapes' 'apple']
Crops which requires very High Ratio of Potassium Content in Soil: ['grapes' 'apple']
Crops which requires very High Rainfall: ['rice' 'papaya' 'coconut']
Crops which requires very Low Temperature : ['grapes']
Crops which requires very High Temperature : ['grapes' 'papaya']
Crops which requires very Low Humidity: ['chickpea' 'kidneybeans']
Crops which requires very Low pH: ['mothbeans']
Crops which requires very High pH: ['mothbeans']
```

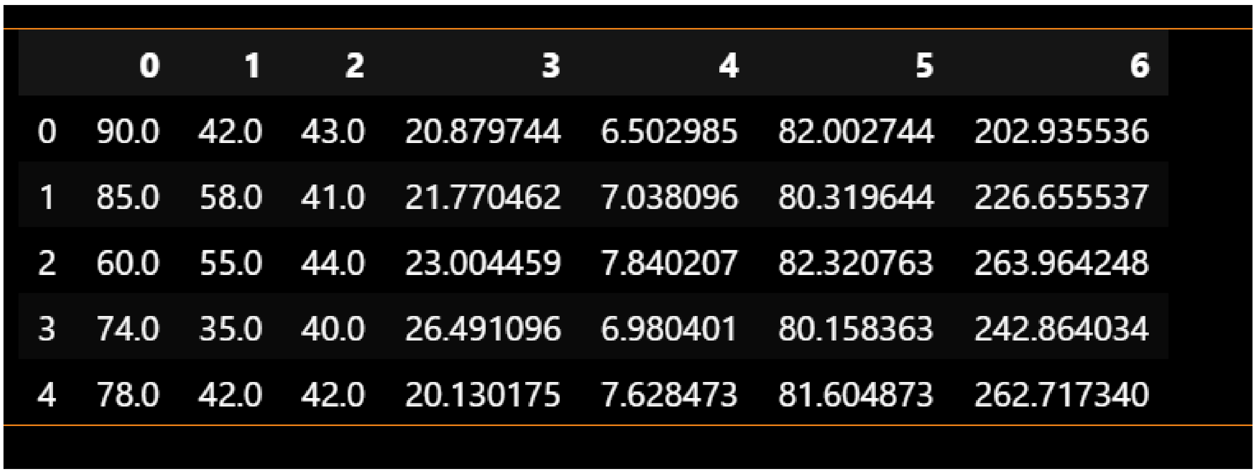
```
print("Summer Crops")
print(data[(data['temperature'] > 30) & (data['humidity'] >
50)]['label'].unique())
print("-----------------------------------")
print("Winter Crops")
print(data[(data['temperature'] < 20) & (data['humidity'] >
30)]['label'].unique())
```

```
print("-----------------------------------")
print("Rainy Crops")
print(data[(data['rainfall'] > 200) & (data['humidity'] >
30)]['label'].unique())
```



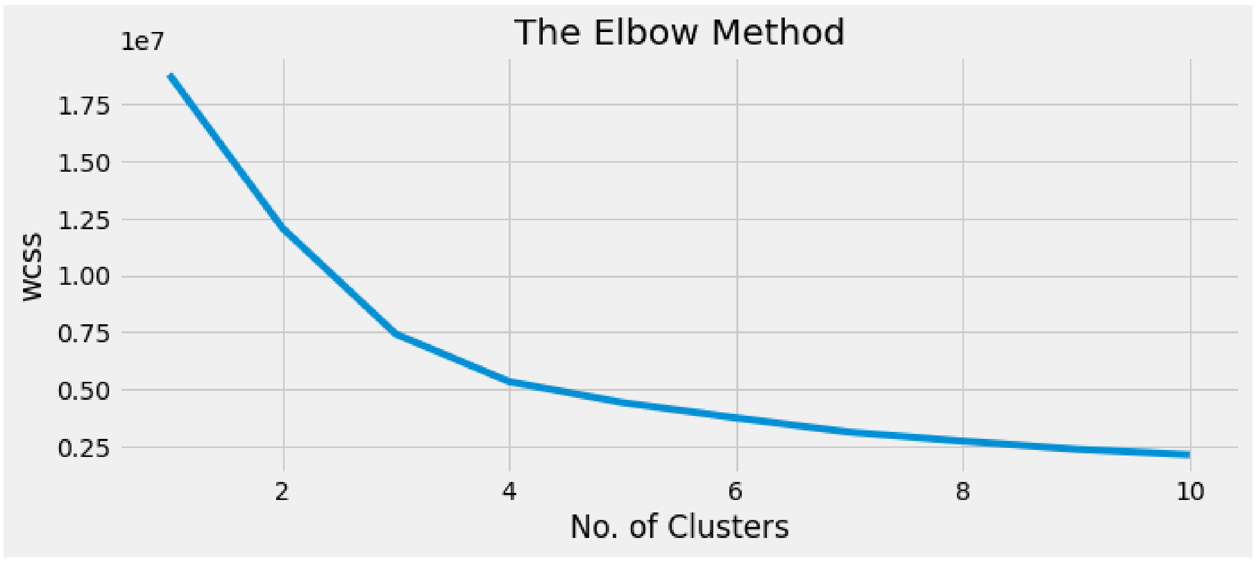
```
Summer Crops
['pigeonpeas' 'mothbeans' 'blackgram' 'mango' 'grapes' 'orange' 'papaya']
-----------------------------------
Winter Crops
['maize' 'pigeonpeas' 'lentil' 'pomegranate' 'grapes' 'orange']
-----------------------------------
Rainy Crops
['rice' 'papaya' 'coconut']
```

```
import warnings
warnings.filterwarnings('ignore')

x = data.loc[:, ['N','P','K','temperature','ph','humidity','rainfall']].values

print(x.shape)

x_data  = pd.DataFrame(x)
x_data.head()
```



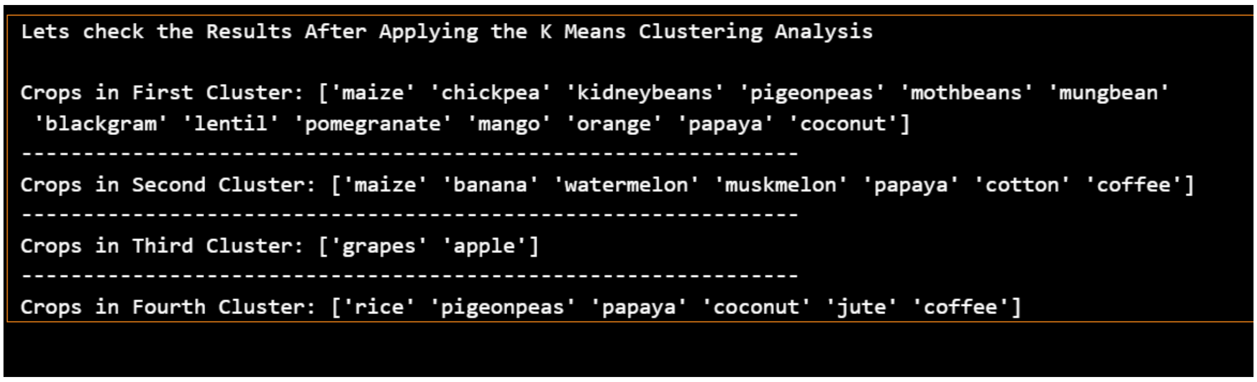
	0	1	2	3	4	5	6
0	90.0	42.0	43.0	20.879744	6.502985	82.002744	202.935536
1	85.0	58.0	41.0	21.770462	7.038096	80.319644	226.655537
2	60.0	55.0	44.0	23.004459	7.840207	82.320763	263.964248
3	74.0	35.0	40.0	26.491096	6.980401	80.158363	242.864034
4	78.0	42.0	42.0	20.130175	7.628473	81.604873	262.717340

```
from sklearn.cluster import KMeans
plt.rcParams['figure.figsize'] = (10, 4)

wcss = []
for i in range(1, 11):
    km = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init =
10, random_state = 0)
    km.fit(x)
    wcss.append(km.inertia_)


plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method', fontsize = 20)
plt.xlabel('No. of Clusters')
plt.ylabel('wcss')
plt.show()
```



```
km = KMeans(n_clusters = 4, init = 'k-means++', max_iter = 300, n_init = 10,
random_state = 0)
y_means = km.fit_predict(x)


a = data['label']
y_means = pd.DataFrame(y_means)
z = pd.concat([y_means, a], axis = 1)
z = z.rename(columns = {0: 'cluster'})
```

```
print("Lets check the Results After Applying the K Means Clustering Analysis
\n")
print("Crops in First Cluster:", z[z['cluster'] == 0]['label'].unique())
print("---------------------------------------------------------------")
print("Crops in Second Cluster:", z[z['cluster'] == 1]['label'].unique())
print("---------------------------------------------------------------")
print("Crops in Third Cluster:", z[z['cluster'] == 2]['label'].unique())
print("---------------------------------------------------------------")
print("Crops in Fourth Cluster:", z[z['cluster'] == 3]['label'].unique())
```

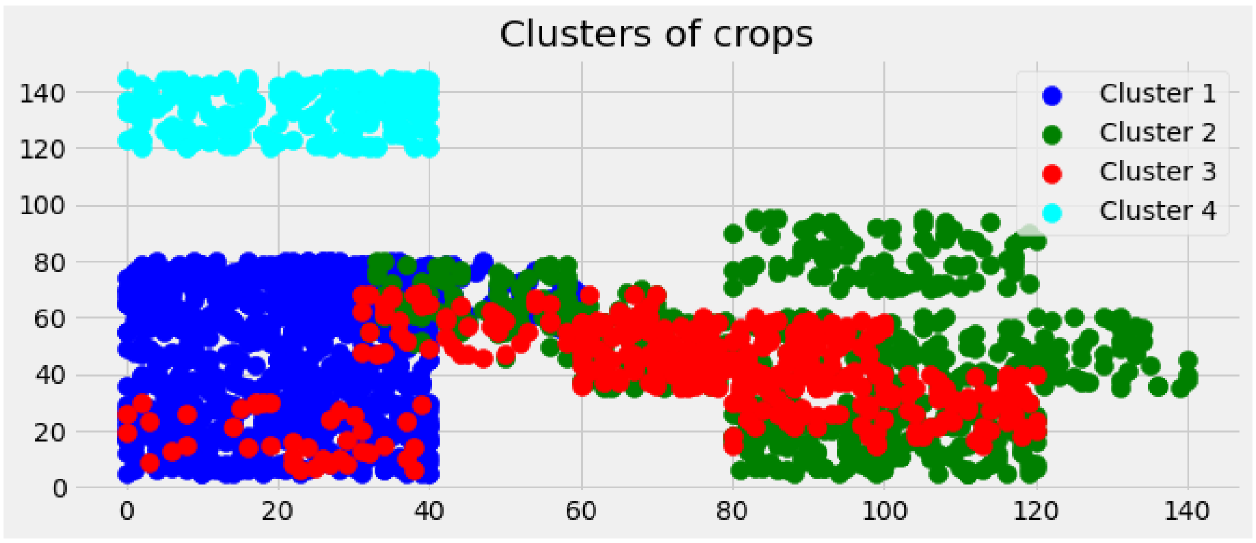


```
Lets check the Results After Applying the K Means Clustering Analysis

Crops in First Cluster: ['maize' 'chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans' 'mungbean'
 'blackgram' 'lentil' 'pomegranate' 'mango' 'orange' 'papaya' 'coconut']
---------------------------------------------------------------
Crops in Second Cluster: ['maize' 'banana' 'watermelon' 'muskmelon' 'papaya' 'cotton' 'coffee']
---------------------------------------------------------------
Crops in Third Cluster: ['grapes' 'apple']
---------------------------------------------------------------
Crops in Fourth Cluster: ['rice' 'pigeonpeas' 'papaya' 'coconut' 'jute' 'coffee']
```

```
from sklearn.cluster import AgglomerativeClustering
hc= AgglomerativeClustering(n_clusters=4, affinity='euclidean',
linkage='ward')
y_her= hc.fit_predict(x)


b = data['label']
y_herr = pd.DataFrame(y_her)
w = pd.concat([y_herr, b], axis = 1)
w= w.rename(columns = {0: 'cluster'})


print("Hierachical Clustering Analysis \n")
print("Crops in Zero Cluster:", w[w['cluster'] == 0]['label'].unique())
print("---------------------------------------------------------------")
print("Crops in First Cluster:", w[w['cluster'] == 1]['label'].unique())
print("---------------------------------------------------------------")
print("Crops in Second Cluster:", w[w['cluster'] == 2]['label'].unique())
print("---------------------------------------------------------------")
print("Crops in Third Cluster:", w[w['cluster'] == 3]['label'].unique())
```

```
Hierachical Clustering Analysis

Crops in Zero Cluster: ['chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans' 'mungbean' 'blackgram'
 'lentil' 'pomegranate' 'mango' 'orange' 'coconut']
---------------------------------------------------------------
Crops in First Cluster: ['maize' 'blackgram' 'banana' 'watermelon' 'muskmelon' 'papaya' 'cotton']
---------------------------------------------------------------
Crops in Second Cluster: ['rice' 'papaya' 'coconut' 'jute' 'coffee']
---------------------------------------------------------------
Crops in Third Cluster: ['grapes' 'apple']
```

```
plt.scatter(x[y_her == 0, 0], x[y_her == 0, 1], s = 100, c = 'blue', label =
'Cluster 1')
plt.scatter(x[y_her == 1, 0], x[y_her == 1, 1], s = 100, c = 'green', label =
'Cluster 2')
plt.scatter(x[y_her== 2, 0], x[y_her == 2, 1], s = 100, c = 'red', label =
'Cluster 3')
plt.scatter(x[y_her == 3, 0], x[y_her == 3, 1], s = 100, c = 'cyan', label =
'Cluster 4')
plt.title('Clusters of crops')
plt.legend()
plt.show()
```



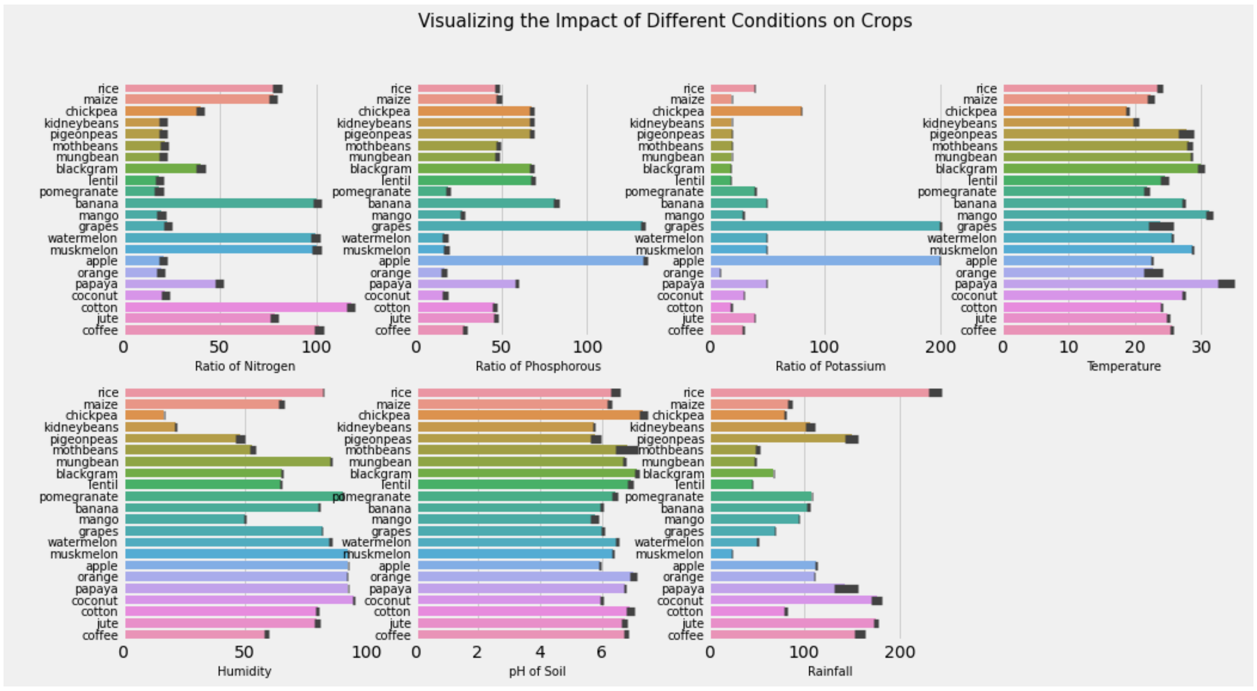
```
print("Results for Hard Clustering\n")
counts = z[z['cluster'] == 0]['label'].value_counts()
d = z.loc[z['label'].isin(counts.index[counts >= 50])]
d = d['label'].value_counts()
```

```
print("Crops in Cluster 1:", list(d.index))
print("--------------------------------------------------")
counts = z[z['cluster'] == 1]['label'].value_counts()
d = z.loc[z['label'].isin(counts.index[counts >= 50])]
d = d['label'].value_counts()
print("Crops in Cluster 2:", list(d.index))
print("--------------------------------------------------")
counts = z[z['cluster'] == 2]['label'].value_counts()
d = z.loc[z['label'].isin(counts.index[counts >= 50])]
d = d['label'].value_counts()
print("Crops in Cluster 3:", list(d.index))
print("--------------------------------------------------")
counts = z[z['cluster'] == 3]['label'].value_counts()
d = z.loc[z['label'].isin(counts.index[counts >= 50])]
d = d['label'].value_counts()
print("Crops in Cluster 4:", list(d.index))
```

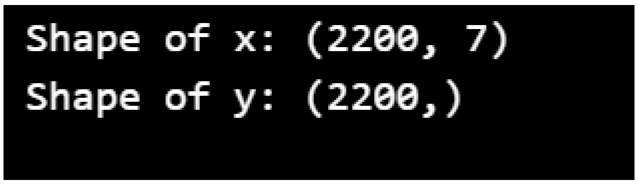
```
Results for Hard Clustering

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

```


plt.rcParams['figure.figsize'] = (15, 8)

plt.subplot(2, 4, 1)
sns.barplot(data['N'], data['label'])
plt.ylabel(' ')
plt.xlabel('Ratio of Nitrogen', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 2)
sns.barplot(data['P'], data['label'])
plt.ylabel(' ')
plt.xlabel('Ratio of Phosphorous', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 3)
sns.barplot(data['K'], data['label'])
plt.ylabel(' ')
plt.xlabel('Ratio of Potassium', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 4)
```

```
sns.barplot(data['temperature'], data['label'])
plt.ylabel(' ')
plt.xlabel('Temperature', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 5)
sns.barplot(data['humidity'], data['label'])
plt.ylabel(' ')
plt.xlabel('Humidity', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 6)
sns.barplot(data['ph'], data['label'])
plt.ylabel(' ')
plt.xlabel('pH of Soil', fontsize = 10)
plt.yticks(fontsize = 10)

plt.subplot(2, 4, 7)
sns.barplot(data['rainfall'], data['label'])
plt.ylabel(' ')
plt.xlabel('Rainfall', fontsize = 10)
plt.yticks(fontsize = 10)

plt.suptitle('Visualizing the Impact of Different Conditions on Crops',
fontsize = 15)
plt.show()
```

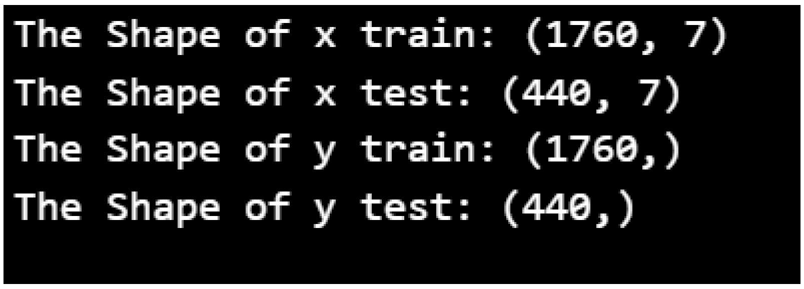


```python
y = data['label']
x = data.drop(['label'], axis = 1)

print("Shape of x:", x.shape)
print("Shape of y:", y.shape)
```



```
Shape of x: (2200, 7)
Shape of y: (2200,)
```

```python
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,
random_state = 0)

print("The Shape of x train:", x_train.shape)
print("The Shape of x test:", x_test.shape)
print("The Shape of y train:", y_train.shape)
print("The Shape of y test:", y_test.shape)
```



```
The Shape of x train: (1760, 7)
The Shape of x test: (440, 7)
The Shape of y train: (1760,)
The Shape of y test: (440,)
```

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score,confusion_matrix,roc_auc_score
from mlxtend.plotting import plot_confusion_matrix
```

```
def evaluator(y_test, y_pred):


    print('Accuracy is: ', accuracy_score(y_test,y_pred))
    print('')

    print('Classification Report: \n',classification_report(y_test,y_pred))

    print('Confusion Matrix: \n\n')
    plt.style.use("ggplot")
    cm = confusion_matrix(y_test,y_pred)
    plot_confusion_matrix(conf_mat = cm,figsize=(10,10),show_normed=True)
    plt.title('Confusion Matrix for Logistic Regression', fontsize = 15)
    plt.show()
```

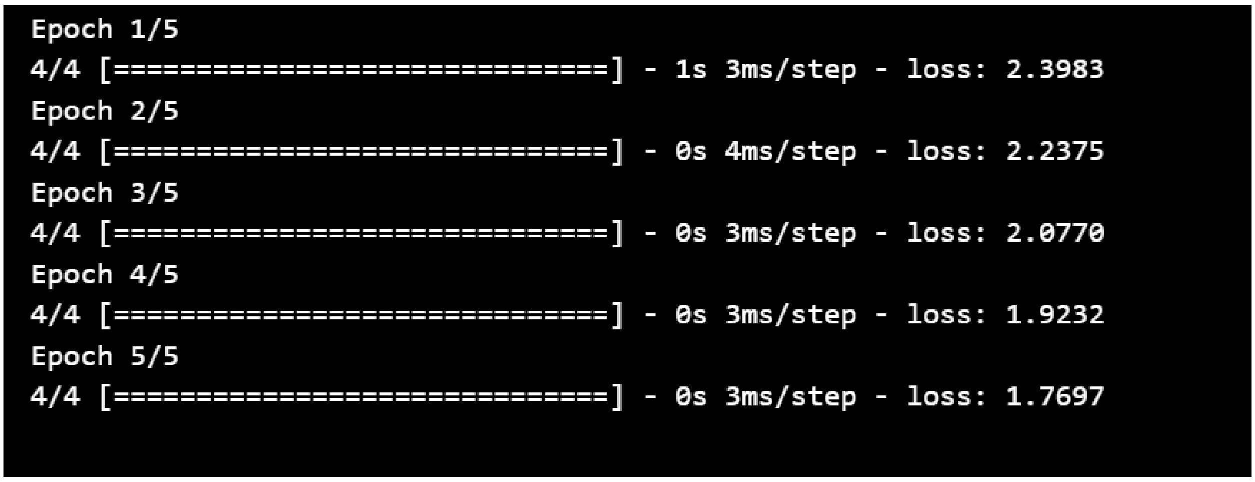
```
from sklearn.neighbors import KNeighborsClassifier
from lib.utils import *
ac=[]
rf_classifier = RandomForestClassifier()
rf_classifier.fit(x_train,y_train)
pred_kn = rf_classifier.predict(x_test)
ac.append(accuracy_score(pred_kn,y_test)*100)
```

```
import pickle
f=open('model.pkl','wb')
pickle.dump(rf_classifier,f)
```

```
X,y=scaler_transform(x)
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,
random_state = 0)
```

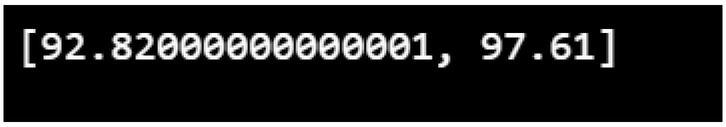
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
x = layers.Input(shape=(12, 7))
cell = layers.SimpleRNNCell(3, activation='tanh')
rnn = layers.RNN(cell)
rnn_output = rnn(x)
from keras.models import Sequential
from keras.layers import Dense
model = Sequential()
```

```
model.add(Dense(8,activation='relu',input_dim=4))
model.add(Dense(1,activation='sigmoid'))

model.compile(loss='binary_crossentropy',optimizer='adam')
model.fit(X_train,y_train,epochs=5)
ac.append(accuracy_score(pred_kn,y_test,sample_weight=0.2)*100)
```



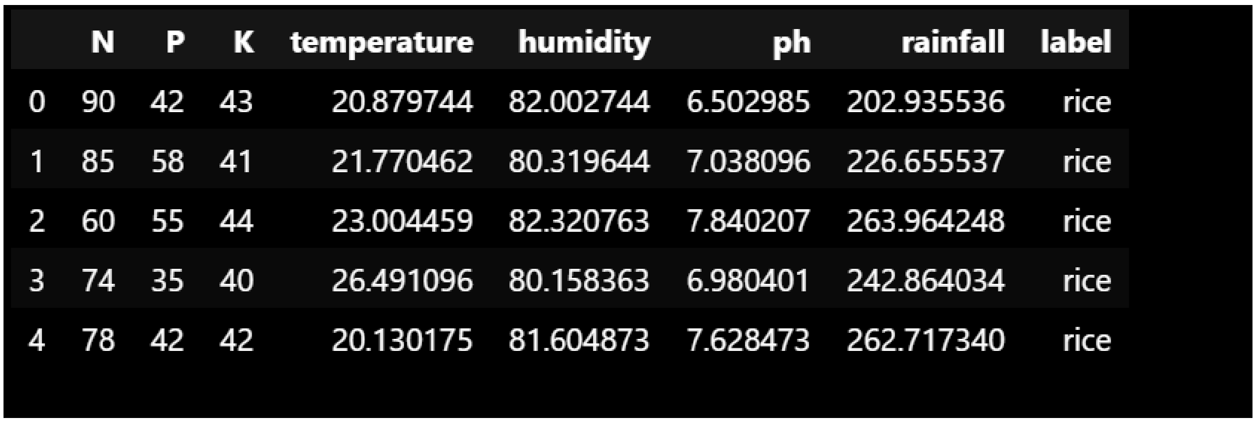
```
Epoch 1/5
4/4 [==============================] - 1s 3ms/step - loss: 2.3983
Epoch 2/5
4/4 [==============================] - 0s 4ms/step - loss: 2.2375
Epoch 3/5
4/4 [==============================] - 0s 3ms/step - loss: 2.0770
Epoch 4/5
4/4 [==============================] - 0s 3ms/step - loss: 1.9232
Epoch 5/5
4/4 [==============================] - 0s 3ms/step - loss: 1.7697
```

```
print(ac)
```



```
[92.82000000000001, 97.61]
```

```
data.head()
```



	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

```
prediction = rf_classifier.predict((np.array([[90,
                                              40,
                                              40,
                                              20,
                                              80,
                                              7,
                                              200]])))
print("The Suggested Crop for Given Climatic Condition is :", prediction)
```

```
The Suggested Crop for Given Climatic Condition is : ['rice']
```

```
data[data['label'] == 'orange'].head()
```

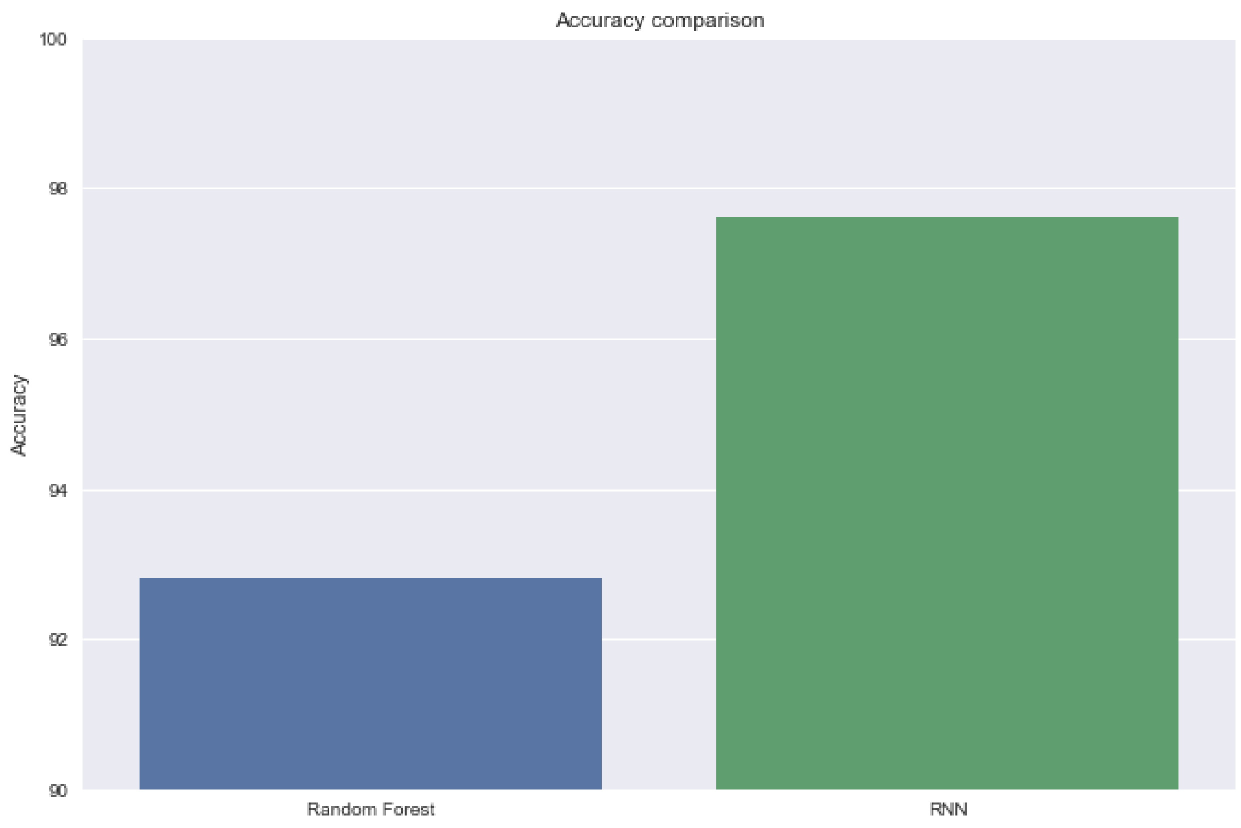
	N	P	K	temperature	humidity	ph	rainfall	label
1600	22	30	12	15.781442	92.510777	6.354007	119.035002	orange
1601	37	6	13	26.030973	91.508193	7.511755	101.284774	orange
1602	27	13	6	13.360506	91.356082	7.335158	111.226688	orange
1603	7	16	9	18.879577	92.043045	7.813917	114.665951	orange
1604	20	7	9	29.477417	91.578029	7.129137	111.172750	orange

```
prediction = rf_classifier.predict((np.array([[20,
                                              30,
                                              10,
                                              15,
                                              90,
                                              7.5,
                                              100]])))

print("The Suggested Crop for Given Climatic Condition is :", prediction)
```

```
The Suggested Crop for Given Climatic Condition is : ['orange']
```

```
x=['Random Forest','RNN']
import matplotlib as plt
plt.style.use('seaborn')
plt.rcParams["figure.figsize"] = (10,7)
print(ac)

y=ac;
import numpy as np
import seaborn as sns
ax=sns.barplot(x,y[:2])
ax.set_title('Accuracy comparison')
ax.set_ylabel('Accuracy')
#ax.yaxis.set_major_locator(ticker.LinearLocator())
low = min(y)
high = max(y)
ax.set_ylim(90,100)
```

```
[92.82000000000001, 97.61]

(90.0, 100.0)
```



CONCLUSION

8.1. Conclusion

After going through many surveys and through analysis we can conclude that the use of no of various machine learning algorithms will not only help farmers to get better results but also increase their revenue which for many is a matter of life and death. Currently farmers make rough estimations based on their previous experiences and plan accordingly, using ML instead will definitely decrease the margin of error and provide them with better outputs. This proposed system will work to provide suggestions, which definitely going to helps farmers to get more yield and better crops.

8.2. Future Enhancement

We will do research on following area of recommendation as well as pricing. We will try to consider both user and providers concerns of changing demand and its cost. This will ensure both provider and customers benefit. Apart from this we will consider competitive prices and its result on pricing. We will study best fit auction based pricing to support optimized fine grained scheme. Also partial waste issue is a area of study which can result in reduced prices using precise scheduling of users’ job. User scheduling behaviors and partial usage waste will be brainstormed to find an effective solution.

BIBLIOGRAPHY

9.1. Journal References

[1] Mayank Champaneri, ChaitanyaChandvidkar, DarpanChachpara, MansingRathod, “Crop yield prediction using machine learning” International Journal of Science and Research, April 2020.

[2] Pavan Patil, VirendraPanpatil, Prof.ShrikantKokate, “Crop Prediction System using Machine Learning Algorithms”, International Research Journal of Engineering and Technology, Feb 2020.

[3] Ramesh Medar, Shweta, Vijay S. Rajpurohit, “Crop Yield Prediction using Machine Learning Techniques”, 5th International Conference for Convergence in Technology, 2019.

[4] TruptiBhange, Swati Shekapure, Komal Pawar, HarshadaChoudhari, “Survey Paper on Prediction of Crop yield and Suitable Crop”, International Journal of Innovative Research in Science, Engineering and Technology, May 2019.

[5] E. Manjula, S. Djodiltachoumy, “A Modal for Prediction of Crop Yield”, International Journal of Computational Intelligence and Informatics, March 2017.

[6] Nishit Jain, Amit Kumar, SahilGarud, Vishal Pradhan, Prajakta Kulkarni, “Crop Selection Method Based on Various Environmental Factors Using Machine Learning”, International Research Journal of Engineering and Technology (IRJET), Feb 2017.

[7] Rakesh Kumar, M.P. Singh, Prabhat Kumar, J.P. Singh, “Crop Selection Method to Maximize Crop Yield Rate using Machine Learning Technique”, 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, T.N., India., May 2015.

9.2. Book References

[8] Liu Qingyuan, and Wang Tianchuan, “Market price theory and practice”. Dalian: Northeast University of Finance and Economics Press, 1998, pp. 15-20.

[9] R. S. Pindyck, and D. L. Rubinfeld, “Econometric Models and Economic Forecasts”, The McGwar-Hill Companies, Inc, 1998, pp. 251-260.

[10] Yao Xia, Peng Hangen, and Zhu Yan, “ARIMA Time Series Modeling and Applying on Fresh Agricultural Products”, System Sciences and Comprehensive Studies in Agriculture, vol. 23, Feb. 2007 , pp. 88-94.

[11] Nie Rong, Qin Keming, and Zhang Xiaohong, “Stochastic Model and Risk Measuring on the Farm-produce′s Price”, Mathematics In Practice and Theory, vol. 34, Nov. 2004, pp. 108-112.

[12] Zhang Xiaioshuan, “Research of Aquatic Products Price Forecasts Support System”, Beijing: China Agricultural University, 2003, pp. 10-15.

[13] Yu Shouhua, Huang Haoran, and Ou Jingying, “Vegetables Prices Prediction Research of Regional Agricultural Products Wholesale Market”, Research of Agricultural Modernization, 27 monog, Dec. 2006, pp. 118-120.

[14] Cheng Xianlu, “Vegetables Prices Forecast and Predict System Research of Beijing's Agricultural Products Wholesale Market”, Beijing Agricultural Sciences, Feb. 2002, pp. 1-10.

9.3. Web References

[15] Yuan Zenren, “Artificial neural network and applications”, Beijing: Qinghua University Press, 1998, pp. 13-14.

[16] MT Hagan, and MB Menhaj, “Training Feed Forward Networks with Marquart Algorithm”, IEEE Trans.on Neural Networks, vol. 5, June 1994, pp. 989-993.
[17] Xu Dong, and Wu Zhen, “System analysis and design-neural network based on the MATLAB6.X” (the 2nd version), Xi’an: Xi’an Electronic Technology University Press,

2002, pp. 30-33.

[18] Li Xiaofeng, “New improvement of BP neural network and its application”, Journal of Jilin Institute of Chemical Technology, vol. 17, Dec. 2000, pp. 48-51.

[19] Gao Ling, “The crop pest situation forecasting and achieving in MATLAB based on BP neural network”, Hefei: Anhui Agricultural University, 2003, pp. 45-52.

[20] Jiang Shaofei, “The method of data processing of artificial neural networks in civil engineering problems”, Journal of Harbin University of Civil Engineering and Architecture, vol. 32, Oct. 1999, pp. 24-28.

[21] Liu Yongjian, and Liu Yijian, “The improving methods of arificial neural network in Geo-technical engineering”, Journal of Guangdong University of Technology, vol. 19, Mar. 2002, pp. 21-25.

[22] Liu Wenxi, “New Keynesian price rigidity micro-theoretical foundation”, Economic Science, May 1997, pp. 60-8