Crop prediction using machine learning.

**Abstract:**

* As we know the fact that, India is the second largest population country in the world and majority of people in India have agriculture as their occupation. Farmers are growing same crops repeatedly without trying new verity of crops and they are applying fertilizers in random quantity without knowing the deficient content and quantity. So, this is directly affecting on crop yield and also causes the soil acidification and damages the top layer. So, we have designed the system using machine learning algorithms for betterment of farmers. Our system will suggest the best suitable crop for particular land based on content and weather parameters. And also, the system provides information about the required content and quantity of fertilizers, required seeds for cultivation. Hence by utilizing our system farmers can cultivate a new variety of crop, may increase in profit margin and can avoid soil pollution.

**INTRODUCTION**

Agriculture is the backbone of every economy. Agriculture is considered as the main and the foremost culture practiced in India. The main goal of agricultural planning is to achieve maximum yield rate of crops by using limited number of land resources. Many machine learning algorithms can help in improving the production of crop yield rate. Whenever there is a loss in unfavorable conditions. We can apply crop selecting method and reduce the losses. The maximizing of yield rate helps in improving economy. We have observed that there is an increase in the suicide rates. So, we want to help the farmers to understand the importance of prior prediction of crop, to increase their knowledge about quality of soil, to understand location wise weather constraints, in order to gain intense yield of crop through our technology solution.

**LITERATURE SURVEY**

**TITLE:** Crop Yield Prediction Using Machine Learning Algorithm

**AUTHOR:** D. Jayanarayana Reddy and Dr. M. Rudrakumar

**ABSTRACT:** Observing the present situation faced by farmers in India. We have also seen that there are so many suicides happening in India from many years, the reason behind this is the changing weather conditions in our country and continuous change in Government. Sometimes farmers are not unaware about the crops which is suitable for their soil. Our project is to help farmers to check the soil quality by analyzing the different related attributes like temperature, moisture, pH value from which quality of the soil is determined and predicts the best suitable crops and their yields in that soil so that the farmers get to know the yields of the crops before cultivating in the agricultural field. Machine learning is the best approach for achieving practical and effective solution for this problem by using various machine learning algorithms which will be precise and accurate in predicting crop yield and provide information to the farmers about the suitable crop and about required fertilizer based on soil quality of the land to get maximum crop yield and revenue.

**EXISTING SYSTEM:**

* Agriculture is one of the important occupation practiced in India. It is the broadest economic sector and plays a most important role in the overall development of the country. More than 60% of the land in the country is used for agriculture in order to suffice the needs of 1.3 billion people Thus adopting new agriculture technologies is very important. This will be leads the farmers of our country towards profit [1]. Prior crop prediction and yield prediction was performed on the basis of farmers experience on a particular location.

**DISADVANTAGES:**

Less accuracy

Less efficient.

High cost.

Man power required.

**PROPOSED SYSTEM:**

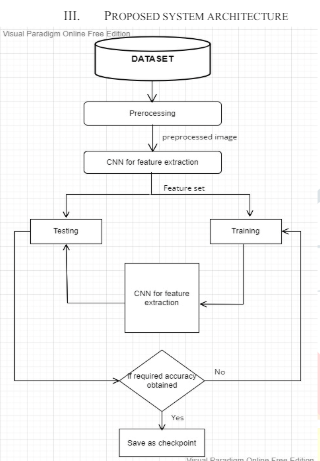
The Proposed system will predict the most suitable crop for particular land based on soil contents and weather parameters such as Temperature, Humidity, soil PH, npk values and Rainfall.

**ADVANTAGES:**

Less cost than previous techniques.

High Accuracy.

**SYSTEM ARCHITECTURE:**



**REQUIREMENT ANALYSIS**

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

**REQUIREMENT SPECIFICATION**

**Functional Requirements**

* Graphical User interface with the User.

**Software Requirements**

For developing the application the following are the Software Requirements:

1. Python
2. DJANGO

**Operating Systems supported**

1. Windows 7
2. Windows XP
3. Windows 8

**Technologies and Languages used to Develop**

1. Python

**Debugger and Emulator**

* Any Browser (Particularly Chrome)

**Hardware Requirements**

For developing the application the following are the Hardware Requirements:

* Processor: Pentium IV or higher
* RAM: 256 MB
* Space on Hard Disk: minimum 512MB

**MODULES:**

**Load Dataset:**

Load data set using pandas read\_csv() method.

**Split Data Set:**

Split the data set to two types. One is train data test and another one is test data set.

**Train data set:**

Train data set will train our data set using fit method.

**Test data set:**

Test data set will test the data set using algorithm.

**Predict data set:**

Predict() method will predict the results.

**INPUT AND OUTPUT DESIGN**

**INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

**OBJECTIVES**

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

**OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the
* Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

**UML DIAGRAMS**

**INTRODUCTION TO UML**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

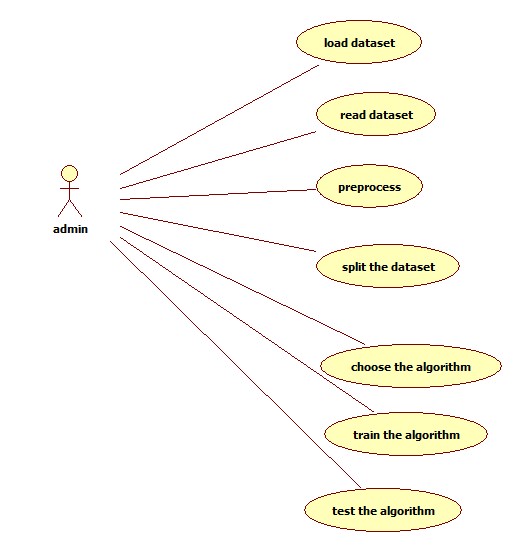
**GOALS:**

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

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

**4.3.2 USECASE DIAGRAM**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

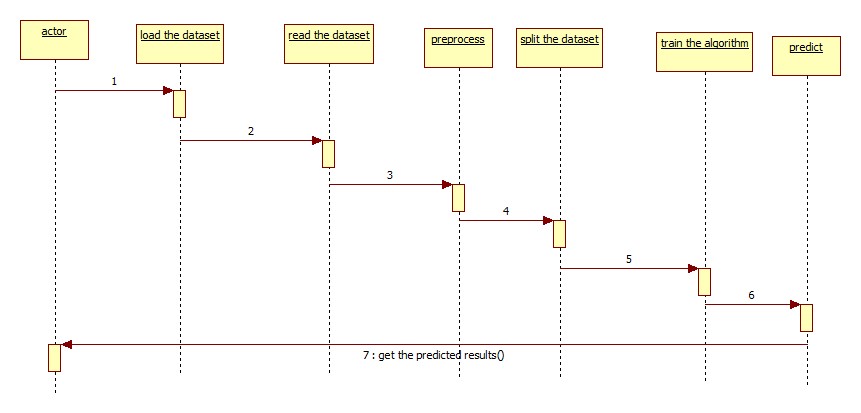


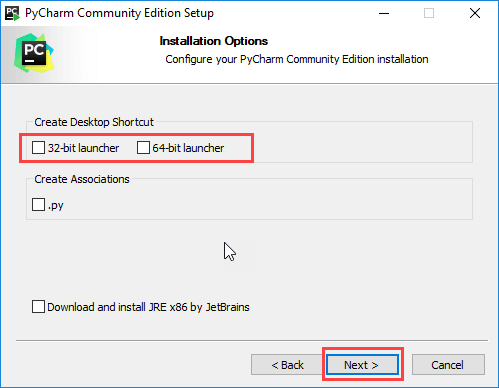
**Fig 4 USECASE DIAGRAM**

**SEQUENCE DIAGRAM**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

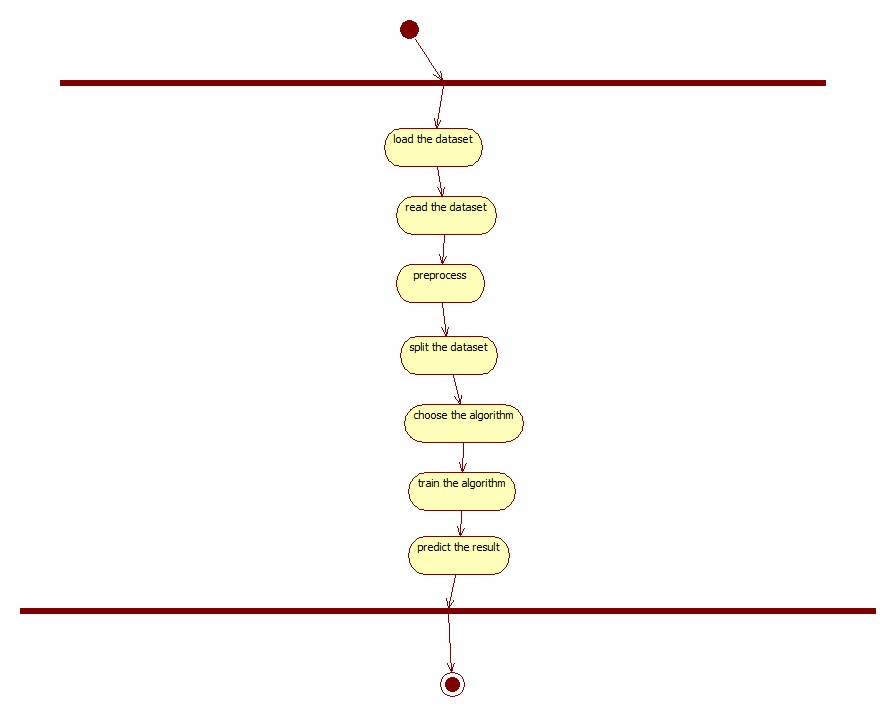
A sequence diagram or system sequence diagram (SSD) shows object interactions arranged in time sequence in the field of software engineering. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.



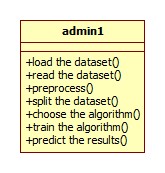
**Fig 6 SEQUENCE DIAGRAM**

**ACTIVITY DIAGRAM**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



**CLASS DIAGRAM:**

****

**PYTHON**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language](https://en.wikipedia.org/wiki/Interpreted_language), Python has a design philosophy that emphasizes code [readability](https://en.wikipedia.org/wiki/Readability) (notably using [whitespace](https://en.wikipedia.org/wiki/Whitespace_character) indentation to delimit [code blocks](https://en.wikipedia.org/wiki/Code_block) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code) than might be used in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B)or [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). [CPython](https://en.wikipedia.org/wiki/CPython" \o "CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source) software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation). Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

**Interactive Mode Programming**

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

$ python

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>>

Type the following text at the Python prompt and press the Enter −

>>> print "Hello, Python!"

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");. However in Python version 2.4.3, this produces the following result −

Hello, Python!

**Script Mode Programming**

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

Live Demo

print "Hello, Python!"

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

$ python test.py

This produces the following result −

Hello, Python!

Let us try another way to execute a Python script. Here is the modified test.py file −

Live Demo

#!/usr/bin/python

print "Hello, Python!"

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows −

$ chmod +x test.py # This is to make file executable

$./test.py

This produces the following result −

Hello, Python!

**Python Identifiers**

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

Starting an identifier with a single leading underscore indicates that the identifier is private.

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

**Reserved Words**

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and exec not

assert finally or

break for pass

class from print

continue global raise

def if return

del import try

elif in while

else is with

except lambda yield

**Lines and Indentation**

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example −

if True:

print "True"

else:

print "False"

However, the following block generates an error −

if True:

print "Answer"

print "True"

else:

print "Answer"

print "False"

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks −

Note − Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

#!/usr/bin/python

import sys

try:

# open file stream

file = open(file\_name, "w")

except IOError:

print "There was an error writing to", file\_name

sys.exit()

print "Enter '", file\_finish,

print "' When finished"

while file\_text != file\_finish:

file\_text = raw\_input("Enter text: ")

if file\_text == file\_finish:

# close the file

file.close

break

file.write(file\_text)

file.write("\n")

file.close()

file\_name = raw\_input("Enter filename: ")

if len(file\_name) == 0:

print "Next time please enter something"

sys.exit()

try:

file = open(file\_name, "r")

except IOError:

print "There was an error reading file"

sys.exit()

file\_text = file.read()

file.close()

print file\_text

Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example −

total = item\_one + \

item\_two + \

item\_three

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example −

days = ['Monday', 'Tuesday', 'Wednesday',

'Thursday', 'Friday']

Quotation in Python

Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal −

word = 'word'

sentence = "This is a sentence."

paragraph = """This is a paragraph. It is

made up of multiple lines and sentences."""

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Live Demo

#!/usr/bin/python

# First comment

print "Hello, Python!" # second comment

This produces the following result −

Hello, Python!

You can type a comment on the same line after a statement or expression −

name = "Madisetti" # This is again comment

You can comment multiple lines as follows −

# This is a comment.

# This is a comment, too.

# This is a comment, too.

# I said that already.

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

'''

This is a multiline

comment.

'''

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Waiting for the User

The following line of the program displays the prompt, the statement saying “Press the enter key to exit”, and waits for the user to take action −

#!/usr/bin/python

raw\_input("\n\nPress the enter key to exit.")

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application.

Multiple Statements on a Single Line

The semicolon ( ; ) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon.

import sys; x = 'foo'; sys.stdout.write(x + '\n')

Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon ( : ) and are followed by one or more lines which make up the suite. For example −

if expression :

suite

elif expression :

suite

else :

suite

**Command Line Arguments**

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

$ python -h

usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...

Options and arguments (and corresponding environment variables):

-c cmd : program passed in as string (terminates option list)

-d : debug output from parser (also PYTHONDEBUG=x)

-E : ignore environment variables (such as PYTHONPATH)

-h : print this help message and exit

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

**Python Lists**

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

list1 = ['physics', 'chemistry', 1997, 2000];

list2 = [1, 2, 3, 4, 5 ];

list3 = ["a", "b", "c", "d"]

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5 );

tup3 = "a", "b", "c", "d";

The empty tuple is written as two parentheses containing nothing −

tup1 = ();

To write a tuple containing a single value you have to include a comma, even though there is only one value −

tup1 = (50,);

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example −

Live Demo

#!/usr/bin/python

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5, 6, 7 );

print "tup1[0]: ", tup1[0];

print "tup2[1:5]: ", tup2[1:5];

When the above code is executed, it produces the following result −

tup1[0]: physics

tup2[1:5]: [2, 3, 4, 5]

Updating Tuples

Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

Live Demo

#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

print "dict['Name']: ", dict['Name']

print "dict['Age']: ", dict['Age']

When the above code is executed, it produces the following result −

dict['Name']: Zara

dict['Age']: 7

If we attempt to access a data item with a key, which is not part of the dictionary, we get an error as follows −

Live Demo

#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

print "dict['Alice']: ", dict['Alice']

When the above code is executed, it produces the following result −

dict['Alice']:

Traceback (most recent call last):

File "test.py", line 4, in <module>

print "dict['Alice']: ", dict['Alice'];

KeyError: 'Alice'

Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

Live Demo

#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

dict['Age'] = 8; # update existing entry

dict['School'] = "DPS School"; # Add new entry

print "dict['Age']: ", dict['Age']

print "dict['School']: ", dict['School']

When the above code is executed, it produces the following result −

dict['Age']: 8

dict['School']: DPS School

Delete Dictionary Elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the del statement. Following is a simple example −

Live Demo

#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

del dict['Name']; # remove entry with key 'Name'

dict.clear(); # remove all entries in dict

del dict ; # delete entire dictionary

print "dict['Age']: ", dict['Age']

print "dict['School']: ", dict['School']

This produces the following result. Note that an exception is raised because after del dict dictionary does not exist any more −

dict['Age']:

Traceback (most recent call last):

File "test.py", line 8, in <module>

print "dict['Age']: ", dict['Age'];

TypeError: 'type' object is unsubscriptable

Note − del() method is discussed in subsequent section.

**Properties of Dictionary Keys**

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys −

(a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example −

IMPLEMENTATION

from tkinter import \*

import tkinter

from tkinter import filedialog

import numpy as np

from tkinter.filedialog import askopenfilename

import pandas as pd

from tkinter import simpledialog

import matplotlib.pyplot as plt

import os

import pandas as pd

import numpy as np

from keras.models import Sequential

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

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

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

from sklearn.preprocessing import LabelEncoder

import keras.layers

from keras.models import model\_from\_json

import pickle

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import PassiveAggressiveClassifier

main = tkinter.Tk()

main.title("Crop Prediction using Machine learning")

main.geometry("1000x650")

global train, test, X\_train, X\_test, y\_train, y\_test

global filename

global cls

def upload():

global filename

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

text.delete('1.0', END)

text.insert(END,filename+" loaded\n");

def traintest(data): #method to generate test and train data from dataset

train=data.iloc[:, 0:7].values

test=data.iloc[: ,8].values

print(train)

print(test)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

train, test, test\_size = 0.3, random\_state = 0)

return train, test, X\_train, X\_test, y\_train, y\_test

def generateModel(): #method to read dataset values which contains all features data

global train, test, X\_train, X\_test, y\_train, y\_test

train1 = pd.read\_csv(filename)

train, test, X\_train, X\_test, y\_train, y\_test = traintest(train1)

text.insert(END,"Train & Test Model Generated\n\n")

text.insert(END,"Total Dataset Size : "+str(len(train1))+"\n")

text.insert(END,"Split Training Size : "+str(len(X\_train))+"\n")

text.insert(END,"Split Test Size : "+str(len(X\_test))+"\n")

def prediction(X\_test, cls): #prediction done here

y\_pred = cls.predict(X\_test)

for i in range(50):

print("X=%s, Predicted=%s" % (X\_test[i], y\_pred[i]))

return y\_pred

# Function to calculate accuracy

def cal\_accuracy(y\_test, y\_pred, details):

accuracy = accuracy\_score(y\_test,y\_pred)\*100

text.insert(END,details+"\n\n")

text.insert(END,"Accuracy : "+str(accuracy)+"\n\n")

return accuracy

def runDCT():

global dct\_acc

global cls

global train, test, X\_train, X\_test, y\_train, y\_test

#Importing Decision Tree classifier

from sklearn.tree import DecisionTreeRegressor

cls=DecisionTreeRegressor()

#Fitting the classifier into training set

cls.fit(X\_train,y\_train)

text.insert(END,"Prediction Results\n\n")

prediction\_data = prediction(X\_test, cls)

dct\_acc = cal\_accuracy(y\_test, prediction\_data,'Decision Tree Accuracy')

def runRF():

global random\_acc

global cls

global train, test, X\_train, X\_test, y\_train, y\_test

#Importing Decision Tree classifier

rf=RandomForestClassifier(n\_estimators=50,max\_depth=2,random\_state=0,class\_weight='balanced')

#Fitting the classifier into training set

rf.fit(X\_train,y\_train)

text.insert(END,"Prediction Results\n\n")

prediction\_data = prediction(X\_test, rf)

random\_acc = cal\_accuracy(y\_test, prediction\_data,'Random Forest Accuracy')

def runPAC():

global pac\_acc

global cls

global train, test, X\_train, X\_test, y\_train, y\_test

linear\_clf = PassiveAggressiveClassifier()

linear\_clf.fit(X\_train,y\_train)

text.insert(END,"Prediction Results\n\n")

prediction\_data = prediction(X\_test, linear\_clf)

pac\_acc = cal\_accuracy(y\_test, prediction\_data,'Random Forest Accuracy')

def predicts():

global clean

global attack

global total

clean = 0;

attack = 0;

text.delete('1.0', END)

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

test = pd.read\_csv(filename)

test = test.values[:, 0:7]

total = len(test)

text.insert(END,filename+" test file loaded\n");

y\_pred = cls.predict(test)

#text.insert(END,y\_pred+" \n");

print(y\_pred)

'''for i in range(len(y\_pred)):

text.insert(END,i," \n");'''

for i in range(len(test)):

if str(y\_pred[i]) == '1.0':

attack = attack + 1

text.insert(END,"X=%s, Predicted = %s" % (test[i], 'Crop name is rice')+"\n\n")

elif str(y\_pred[i]) == '2.0':

clean = clean + 1

text.insert(END,"X=%s, Predicted = %s" % (test[i], 'crop name is wheat')+"\n\n")

elif str(y\_pred[i]) == '3.0':

clean = clean + 1

text.insert(END,"X=%s, Predicted = %s" % (test[i], 'crop name is Mung Bean')+"\n\n")

elif str(y\_pred[i]) == '4.0':

clean = clean + 1

text.insert(END,"X=%s, Predicted = %s" % (test[i], 'crop name is Tea')+"\n\n")

elif str(y\_pred[i]) == '5.0':

clean = clean + 1

text.insert(END,"X=%s, Predicted = %s" % (test[i], 'crop name is millet')+"\n\n")

elif str(y\_pred[i]) == '6.0':

clean = clean + 1

text.insert(END,"X=%s, Predicted = %s" % (test[i], 'crop name is maize')+"\n\n")

elif str(y\_pred[i]) == '7.0':

clean = clean + 1

text.insert(END,"X=%s, Predicted = %s" % (test[i], 'crop name is Lentil')+"\n\n")

def graph():

height = [dct\_acc,random\_acc,pac\_acc]

bars = ('Decission tree','Random forest','PassiveAggressiveClassifier')

y\_pos = np.arange(len(bars))

plt.bar(y\_pos, height)

plt.xticks(y\_pos, bars)

plt.show()

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

title = Label(main, text='Crop Prediction using Machine Learning', justify=LEFT)

title.config(bg='lavender blush', fg='DarkOrchid1')

title.config(font=font)

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

title.place(x=100,y=5)

title.pack()

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

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

uploadButton.place(x=10,y=100)

uploadButton.config(font=font1)

preprocessButton = Button(main, text="Preprocess Dataset", command=generateModel)

preprocessButton.place(x=300,y=100)

preprocessButton.config(font=font1)

rnnButton = Button(main, text="Run Decisiontree Algorithm", command=runDCT)

rnnButton.place(x=480,y=100)

rnnButton.config(font=font1)

lstmButton = Button(main, text="Run Randomforest Algorithm", command=runRF)

lstmButton.place(x=700,y=100)

lstmButton.config(font=font1)

ffButton = Button(main, text="Passive Aggressive Algorithm", command=runPAC)

ffButton.place(x=10,y=150)

ffButton.config(font=font1)

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

graphButton.place(x=300,y=150)

graphButton.config(font=font1)

predictButton = Button(main, text="Detect Crop", command=predicts)

predictButton.place(x=550,y=150)

predictButton.config(font=font1)

'''

predictButton = Button(main, text="Predict Disease using Test Data", command=predict)

predictButton.place(x=10,y=200)

predictButton.config(font=font1)

topButton = Button(main, text="Top 6 Crop Yield Graph", command=topGraph)

topButton.place(x=300,y=200)

topButton.config(font=font1)'''

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

text=Text(main,height=20,width=160)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=250)

text.config(font=font1)

main.config(bg='light coral')

main.mainloop()

**SYSTEM TEST**

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.

### TYPES OF TESTS

**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.

**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.

**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.

**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.

**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.

**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.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

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

**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.

**CONCLUSION:**

* Presently our farmers are not effectively using technology and analysis, so there may be a chance of wrong selection of crop for cultivation that will reduce their income. To reduce those type of loses we have developed a farmer friendly system with GUI, that will predict which would be the best suitable crop for particular land and this system will also provide information about required nutrients to add up, required seeds for cultivation, expected yield and market price. So, this makes the farmers to take right decision in selecting the crop for cultivation such that agricultural sector will be developed by innovative idea.

**REFERENCES:**

[1] D. Jayanarayana Reddy and Dr. M. Rudrakumar “Crop Yield Prediction Using Machine Learning Algorithm” in 2021[IEEE].

[2] Shreya V. Bhosale et.al “Crop Yield Prediction Using Data Analytics and Hybrid Approach” in 2018[IEEE].

[3] Dr. Y. Jeevan Nagandrakumar et.al “Supervised Machine Learning Approach for Crop Yield Prediction in Agriculture Sector” in 2020[IEEE].

[4] Shruthi Mishra, Priyanka Paygude et.al “Use of Data Mining in Crop Yield Prediction” in 2018[IEEE].

[5] Pranay Malik et.al “Comparative Analysis of Soil Properties to Predict Fertility and Crop Yield using Machine Learning Algorithm” in 2021[IEEE].

[6] Ramesh Medar et.al “Crop Yield Prediction Using Machine Learning Techniques” in 2019[IEEE].