In [9]:

```
import pyttsx3   # Importing pyttsx3 library to convert text into speech.
import translate
from translate import Translator


import pandas as pd                                                       # Importing pandas library
from sklearn import preprocessing                                         # Importing sklearn library. This is a very powerfull library for machine learning. Scikit-learn is probably the most useful li
from sklearn.neighbors import KNeighborsClassifier                        # Importing Knn Classifier from sklearn library.
import numpy as np                                                        # Importing numpy to do stuffs related to arrays
import PySimpleGUI as sg                                                  # Importing pysimplegui to make a Graphical User Interface.

excel = pd.read_excel('Crop.xlsx', header = 0)                            # Importing our excel data from a specific file.
print(excel)                                                              # Printing our excel file data.
print(excel.shape)                                                        # Checking out the shape of our data.

engine = pyttsx3.init('sapi5')                                            # Defining the speech rate, type of voice etc.
voices = engine.getProperty('voices')
rate = engine.getProperty('rate')
engine.setProperty('rate', rate-50)
engine.setProperty('voice',voices[1].id)
translator= Translator(from_lang="english",to_lang="hindi")




def speak(translator):                                                         # Defining a speak function. We can call this function when we want to make our program to speak something.
    engine.say(audio)
    engine.runAndWait()


le = preprocessing.LabelEncoder()                                         # Various machine learning algorithms require numerical input data, so you need to represent categorical columns in a numerical
crop = le.fit_transform(list(excel["CROP"]))                              # Mapping the values in weather into numerical form.


NITROGEN = list(excel["NITROGEN"])                                        # Making the whole row consisting of nitrogen values to come into nitrogen.
PHOSPHORUS = list(excel["PHOSPHORUS"])                                    # Making the whole row consisting of phosphorus values to come into phosphorus.
POTASSIUM = list(excel["POTASSIUM"])                                      # Making the whole row consisting of potassium values to come into potassium.
TEMPERATURE = list(excel["TEMPERATURE"])                                  # Making the whole row consisting of temperature values to come into temperature.
HUMIDITY = list(excel["HUMIDITY"])                                        # Making the whole row consisting of humidity values to come into humidity.
PH = list(excel["PH"])                                                    # Making the whole row consisting of ph values to come into ph.
RAINFALL = list(excel["RAINFALL"])                                        # Making the whole row consisting of rainfall values to come into rainfall.


features = list(zip(NITROGEN, PHOSPHORUS, POTASSIUM, TEMPERATURE, HUMIDITY, PH, RAINFALL))                     # Zipping all the features together
features = np.array([NITROGEN, PHOSPHORUS, POTASSIUM, TEMPERATURE, HUMIDITY, PH, RAINFALL])                    # Converting all the features into a array form

features = features.transpose()                                                                                # Making transpose of the features
print(features.shape)                                                                                          # Printing the shape of the features after getting transposed.
print(crop.shape)                                                                                              # Printing the shape of crop. Please note that the shape of the features and crop should m

model = KNeighborsClassifier(n_neighbors=3)                                                                    # The number of neighbors is the core deciding factor. K is generally an odd number if the
model.fit(features, crop)                                                                                      # fit your model on the train set using fit() and perform prediction on the test set using
layout = [[sg.Text('                      Crop Recommendation Assistant', font=("Helvetica", 30), text_color = 'yellow')],                                                    # Defining the layout of th
         [sg.Text('Please enter the following details :-', font=("Helvetica", 20))],                                                                                          # We have defined the text
         [sg.Text('Enter ratio of Nitrogen in the soil                                  :', font=("Helvetica", 20)), sg.Input(font=("Helvetica",20), size = (20,1) )],
         [sg.Text('Enter ratio of Phosphorous in the soil                           :', font=("Helvetica", 20)), sg.Input(font=("Helvetica", 20),size = (20,1))],
         [sg.Text('Enter ratio of Potassium in the soil                               :', font=("Helvetica", 20)), sg.Input(font=("Helvetica", 20),size = (20,1))],
         [sg.Text('Enter average Temperature value around the field        :', font=("Helvetica", 20)), sg.Input(font=("Helvetica", 20),size = (20,1)), sg.Text('*C', font=("Helvetica", 20))],
         [sg.Text('Enter average percentage of Humidity around the field :', font=("Helvetica", 20)), sg.Input(font=("Helvetica", 20),size = (20,1)), sg.Text('%', font=("Helvetica", 20))],
         [sg.Text('Enter PH value of the soil                                            :', font=("Helvetica", 20)), sg.Input(font=("Helvetica", 20),size = (20,1))],
         [sg.Text('Enter average amount of Rainfall around the field        :', font=("Helvetica", 20) ), sg.Input(font=("Helvetica", 20),size = (20,1)),sg.Text('mm', font=("Helvetica", 20))],
         [sg.Text(size=(50,1),font=("Helvetica",20) , text_color = 'yellow', key='-OUTPUT1-' )],
         [sg.Button('Submit', font=("Helvetica", 20)),sg.Button('Quit', font=("Helvetica", 20))] ]
window = sg.Window('Crop Recommendation Assistant', layout)

while True:
    event, values = window.read()
    if event == sg.WINDOW_CLOSED or event == 'Quit':                                                                            # If the user will press the quit button then the program
        break
    print(values[0])
    nitrogen_content =         values[0]                                                                                                        # Taking input from the user about nitrogen content in th
    phosphorus_content =       values[1]                                                                                                        # Taking input from the user about phosphorus content in
    potassium_content =        values[2]                                                                                                        # Taking input from the user about potassium content in t
    temperature_content =      values[3]                                                                                                        # Taking input from the user about the surrounding temper
    humidity_content =         values[4]                                                                                                        # Taking input from the user about the surrounding humidi
    ph_content =               values[5]                                                                                                        # Taking input from the user about the ph level of the so
    rainfall =                 values[6]                                                                                                        # Taking input from the user about the rainfall.
    predict1 = np.array([nitrogen_content,phosphorus_content, potassium_content, temperature_content, humidity_content, ph_content, rainfall])  # Converting all the data that we collected from the user
    print(predict1)                                                                                                                             # Printing the data after being converted into a array fo
    predict1 = predict1.reshape(1,-1)                                                                              # Reshaping the input data so that it can be applied in the model for getting accurate
    print(predict1)                                                                                                # Printing the input data value after being reshaped.
    predict1 = model.predict(predict1)                                                                             # Applying the user input data into the model.
    print(predict1)                                                                                                # Finally printing out the results.
    crop_name = str()
    if predict1 == 0:                                                                                              # Above we have converted the crop names into numerical form, so that we can apply the
        crop_name = 'Apple(सेब)'
    elif predict1 == 1:
        crop_name = 'Banana(केला)'
    elif predict1 == 2:
        crop_name = 'Blackgram(काला चना)'
    elif predict1 == 3:
        crop_name = 'Chickpea(काबुली चना)'
    elif predict1 == 4:
        crop_name = 'Coconut(नारियल)'
    elif predict1 == 5:
        crop_name = 'Coffee(कॉफ़ी)'
    elif predict1 == 6:
        crop_name = 'Cotton(कपास)'
    elif predict1 == 7:
        crop_name = 'Grapes(अंगूर)'
    elif predict1 == 8:
        crop_name = 'Jute(जूट)'
    elif predict1 == 9:
        crop_name = 'Kidneybeans(राज़में)'
    elif predict1 == 10:
        crop_name = 'Lentil(मसूर की दाल)'
    elif predict1 == 11:
        crop_name = 'Maize(मक्का)'
    elif predict1 == 12:
        crop_name = 'Mango(आम)'
    elif predict1 == 13:
        crop_name = 'Mothbeans(मोठबीन)'
    elif predict1 == 14:
        crop_name = 'Mungbeans(मूंग)'
    elif predict1 == 15:
        crop_name = 'Muskmelon(खरबूजा)'
    elif predict1 == 16:
        crop_name = 'Orange(संतरा)'
```

```
elif predict1 == 17:
    crop_name = 'Papaya(पपीता)'
elif predict1 == 18:
    crop_name = 'Pigeonpeas(कबूतर के मटर)'
elif predict1 == 19:
    crop_name = 'Pomegranate(अनार)'
elif predict1 == 20:
    crop_name = 'Rice(चावल)'
elif predict1 == 21:
    crop_name = 'Watermelon(तरबूज)'

if int(humidity_content) >=1 and int(humidity_content)<= 33 :                                              # Here I have divided the humidity values into three categories i.e low humid, medium hu
    humidity_level = 'low humid'
elif int(humidity_content) >=34 and int(humidity_content) <= 66:
    humidity_level = 'medium humid'
else:
    humidity_level = 'high humid'

if int(temperature_content) >= 0 and int(temperature_content)<= 6:                                         # Here I have divided the temperature values into three categories i.e cool, warm, hot.
    temperature_level = 'cool'
elif int(temperature_content) >=7 and int(temperature_content) <= 25:
    temperature_level = 'warm'
else:
    temperature_level= 'hot'

if int(rainfall) >=1 and int(rainfall) <= 100:                                                            # Here I have divided the humidity values into three categories i.e less, moderate, heavy
    rainfall_level = 'less'
elif int(rainfall) >= 101 and int(rainfall) <=200:


    rainfall_level = 'moderate'
elif int(rainfall) >=201:
    rainfall_level = 'heavy rain'

if int(nitrogen_content) >= 1 and int(nitrogen_content) <= 50:                                           # Here I have divided the nitrogen values into three categories.
    nitrogen_level = 'less'
elif int(nitrogen_content) >=51 and int(nitrogen_content) <=100:
    nitrogen_level = 'not to less but also not to high'
elif int(nitrogen_content) >=101:
    nitrogen_level = 'high'

if int(phosphorus_content) >= 1 and int(phosphorus_content) <= 50:                                       # Here I have divided the phosphorus values into three categories.
    phosphorus_level = 'less'
elif int(phosphorus_content) >= 51 and int(phosphorus_content) <=100:
    phosphorus_level = 'not to less but also not to high'
elif int(phosphorus_content) >=101:
    phosphorus_level = 'high'

if int(potassium_content) >= 1 and int(potassium_content) <=50:                                          # Here I have divided the potassium values into three categories.
    potassium_level = 'less'
elif int(potassium_content) >= 51 and int(potassium_content) <= 100:
    potassium_level = 'not to less but also not to high'

elif int(potassium_content) >=101:
    potassium_level = 'high'


if float(ph_content) >=0 and float(ph_content) <=5:                                                      # Here I have divided the ph values into three categories.
    phlevel = 'acidic'
    elif float(ph_content) >= 6 and float(ph_content) <= 8:
        phlevel = 'neutral'
    elif float(ph_content) >= 9 and float(ph_content) <= 14:
        phlevel = 'alkaline'

    print(crop_name)
    print(humidity_level)

    print(temperature_level)
    print(rainfall_level)
    print(nitrogen_level)
    print(phosphorus_level)
    print(potassium_level)

    print(phlevel)



    speak("WELCOME to  CROP RECCOMEDATION SYSTEM  Sir/madam according to the data that you provided to me. The ratio of nitrogen in the soil is  " + nitrogen_level + ". The ratio of phosphorus in the s
    window['-OUTPUT1-'].update('The best crop that you can grow : ' + crop_name )                                      # Suggesting the best crop after prediction.
    speak("The best crop that you can grow is  " + crop_name)                                                          # Speaking the name of the predicted crop.



window.close()
```

```
      NITROGEN  PHOSPHORUS  POTASSIUM  TEMPERATURE   HUMIDITY        PH  \
0           90          42         43    20.879744  82.002744  6.502985
1           85          58         41    21.770462  80.319644  7.038096
2           60          55         44    23.004459  82.320763  7.840207
3           74          35         40    26.491096  80.158363  6.980401
4           78          42         42    20.130175  81.604873  7.628473
...        ...         ...        ...          ...        ...       ...
2195       107          34         32    26.774637  66.413269  6.780064
2196        99          15         27    27.417112  56.636362  6.086922
2197       118          33         30    24.131797  67.225123  6.362608
2198       117          32         34    26.272418  52.127394  6.758793
2199       104          18         30    23.603016  60.396475  6.779833

        RAINFALL    CROP
0     202.935536    rice
1     226.655537    rice
2     263.964248    rice
3     242.864034    rice
4     262.717340    rice
...          ...     ...
2195  177.774507  coffee
2196  127.924610  coffee
2197  173.322839  coffee
2198  127.175293  coffee
2199  140.937041  coffee

[2200 rows x 8 columns]
(2200, 8)
(2200, 7)
(2200,)
12
['12' '21' '22' '21' '12' '12' '22']
[['12' '21' '22' '21' '12' '12' '22']]
[13]
Mothbeans(मोठबीन)
low humid
```

```
warm
less
less
less
less
alkaline
C:\Users\RDP\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: FutureWarning: Arrays of bytes/strings is being converted to decimal numbers if dtype='numeric'. This behavior is deprecated in
0.24 and will be removed in 1.1 (renaming of 0.26). Please convert your data to numeric values explicitly instead.
  return f(*args, **kwargs)
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-9-832542df8445> in <module>
    193     translator= Translator(from_lang="english",to_lang="hindi")
    194 
--> 195     translator.speak("WELCOME to  CROP RECCOMEDATION SYSTEM  Sir/madam according to the data that you provided to me. The ratio of nitrogen in the soil is  " + nitrogen_level + ". The ratio of p
hosphorus in the soil is  " + phosphorus_level + ". The ratio of potassium in the soil is  " + potassium_level + ". The temperature level around the field is  " + temperature_level + ". The humidity lev
el around the field is  " + humidity_level + ". The ph type of the soil is  " + phlevel + ". The amount of rainfall is  " + rainfall_level )  # Making our program to speak about the data that it has rec
eived about the crop in front of the user.
    196     window['-OUTPUT1-'].update('The best crop that you can grow : ' + crop_name )                                   # Suggesting the best crop after prediction.
    197     translator.speak("The best crop that you can grow is  " + crop_name)                                                     # Speaking the name of the predicted crop.

AttributeError: 'Translator' object has no attribute 'speak'
```

In [2]:
```
pip install translate
```

```
Collecting translate
  Downloading translate-3.6.1-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: lxml in c:\users\rdp\anaconda3\lib\site-packages (from translate) (4.6.3)
Requirement already satisfied: click in c:\users\rdp\anaconda3\lib\site-packages (from translate) (7.1.2)
Requirement already satisfied: requests in c:\users\rdp\anaconda3\lib\site-packages (from translate) (2.25.1)
Collecting libretranslatepy==2.1.1
  Downloading libretranslatepy-2.1.1-py3-none-any.whl (3.2 kB)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\rdp\anaconda3\lib\site-packages (from requests->translate) (2020.12.5)
Requirement already satisfied: idna<3,>=2.5 in c:\users\rdp\anaconda3\lib\site-packages (from requests->translate) (2.10)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\rdp\anaconda3\lib\site-packages (from requests->translate) (4.0.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\rdp\anaconda3\lib\site-packages (from requests->translate) (1.26.4)
Installing collected packages: libretranslatepy, translate
Successfully installed libretranslatepy-2.1.1 translate-3.6.1
Note: you may need to restart the kernel to use updated packages.
```

In [ ]: