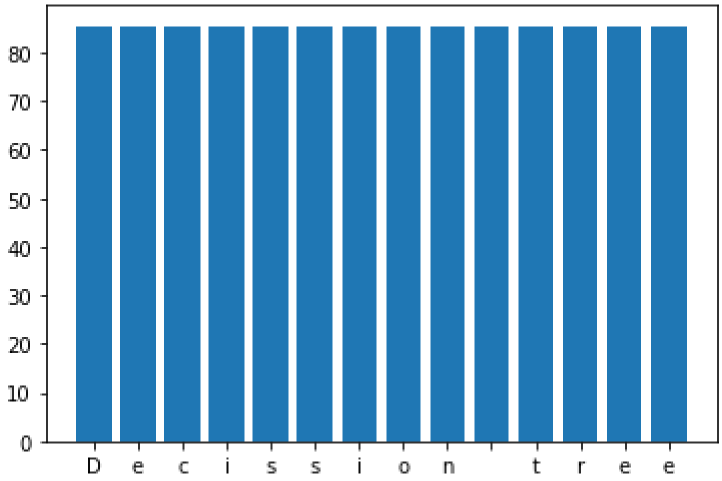
In [ ]:

```
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 tensorflow.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")

bg1 = PhotoImage( file = "images1.png")

# Show image using label
label1 = Label( main, image = bg1)
label1.place(x = 0,y = 0)

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 d
    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")
```

```python
    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 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'
        elif str(y_pred[i]) == '2.0':
            clean = clean + 1
            text.insert(END,"X=%s, Predicted = %s" % (test[i], 'crop name is wheat
        elif str(y_pred[i]) == '3.0':
            clean = clean + 1
            text.insert(END,"X=%s, Predicted = %s" % (test[i], 'crop name is Mung
        elif str(y_pred[i]) == '4.0':
            clean = clean + 1
            text.insert(END,"X=%s, Predicted = %s" % (test[i], 'crop name is Tea')
        elif str(y_pred[i]) == '5.0':
            clean = clean + 1
            text.insert(END,"X=%s, Predicted = %s" % (test[i], 'crop name is mille
        elif str(y_pred[i]) == '6.0':
            clean = clean + 1
            text.insert(END,"X=%s, Predicted = %s" % (test[i], 'crop name is maize
        elif str(y_pred[i]) == '7.0':
            clean = clean + 1
            text.insert(END,"X=%s, Predicted = %s" % (test[i], 'crop name is Lenti
def graph():
    height = [dct_acc]
    bars = ('Decission tree')
    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(fg='DarkOrchid1')
title.config(font=font)
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)


graphButton = Button(main, text="Accuracy 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)


font1 = ('times', 12, 'bold')
text=Text(main,height=10,width=160)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=250)
text.config(font=font1)

main.mainloop()
```

```
[[20.87974371 82.00274423  6.50298529 ...  0.7         0.1
   0.8       ]
 [21.77046169 80.31964408  7.03809636 ...  0.5         0.7
   0.4       ]
 [23.00445915 82.3207629   7.84020714 ...  0.7         0.6
   0.1       ]
 ...
 [25.3310446  84.30533791  6.90424171 ...  0.8         0.7
   0.6       ]
 [26.89750174 83.89241484  6.46327108 ...  0.5         0.8
   0.3       ]
 [26.98603693 89.4138489   6.26083896 ...  0.2         0.8
   0.5       ]]
[ 1  1  1 ... 31 31 31]
X=[25.03149561 82.21276599  7.95462932 95.0191318   0.8         0.2
  0.5       ], Predicted=10.0
X=[ 39.30050027  94.16193416   6.57467759 120.9512466    0.2
   0.6          0.8       ], Predicted=28.0
X=[24.70626432 60.26854183  6.05218488 53.12442925  0.5         0.6
  0.8       ], Predicted=7.0
X=[34.16438906 54.16482251  4.95473956 98.33351125  0.5         0.8
  0.3       ], Predicted=21.0
X=[25.95263264 61.89082199  6.32523516 99.57981207  0.8         0.5
  0.2       ], Predicted=19.0
X=[ 29.50304807  63.46513414   5.56022458 189.5208915    0.8
   0.2          0.7       ], Predicted=21.0
X=[46.42017709 14.91813537  6.7004592  40.96391853  0.1         0.7
  0.5       ], Predicted=5.0
X=[ 21.37784654  92.72043743   5.57324139 106.1417017    0.8
   0.7          0.6       ], Predicted=25.0
X=[40.48200586 13.15089404  5.91542202 38.04847719  0.6         0.5
  0.4       ], Predicted=5.0
X=[ 28.96318258  95.16333673   6.16508485 222.803013     0.6
   0.6          0.8       ], Predicted=18.0
X=[25.96779712 81.97904282  7.27231621 74.14169043  0.7         0.5
  0.4       ], Predicted=10.0
X=[27.92678579 86.5543196   7.18318992 43.4826194   0.5         0.3
  0.7       ], Predicted=3.0
X=[ 32.69435583  62.51014083   6.31098546 111.518131     0.6
   0.5          0.8       ], Predicted=2.0
X=[ 24.78475421  66.89269543   5.59516348 321.540866     0.8
   0.5          0.8       ], Predicted=13.0
X=[29.77013109 66.29327012  6.54736162 35.69674138  0.4         0.8
  0.5       ], Predicted=7.0
X=[ 23.92271434  68.73467      5.52853696 299.3477453    0.8
   0.6          0.3       ], Predicted=13.0
X=[ 27.69819273  51.41593238   5.40390833 100.7720705    0.4
   0.8          0.7       ], Predicted=26.0
X=[2.65272351e+01 8.14175385e+01 5.38616779e+00 2.64614870e+02
 1.00000000e-01 6.00000000e-01 5.00000000e-01], Predicted=1.0
X=[27.89636126 88.71782287  6.78415271 57.79863368  0.5         0.8
  0.5       ], Predicted=3.0
X=[ 22.10621387  91.34039616   6.76985566 106.8704803    0.5
   0.6          0.2       ], Predicted=30.0
X=[31.49338309 63.0563645   6.52121796 71.48327008  0.6         0.8
  0.7       ], Predicted=19.0
X=[28.99319096 62.85948245  8.18384484 70.4713043   0.8         0.5
  0.5       ], Predicted=19.0
X=[ 32.14778175  58.31526308   6.21435344 144.4624105    0.8
   0.8          0.4       ], Predicted=11.0
X=[ 26.59104992  82.94164078   6.03348526 161.2469997    0.2
   0.7          0.7       ], Predicted=14.0
X=[ 31.24021696  56.67369054   7.33932093 122.0146733    0.6
   0.3          0.5       ], Predicted=2.0
```

```
X=[2.00913695e+01 6.48699569e+01 5.95659490e+00 2.55788016e+02
 8.00000000e-01 6.00000000e-01 2.00000000e-01], Predicted=13.0
X=[24.64458469 85.49938185  6.34394252 48.31219031  0.7         0.8
  0.4       ], Predicted=31.0
X=[26.88630675 41.69617915  4.75092922 94.46748008  0.7         0.6
  0.8       ], Predicted=21.0
X=[ 54.85013111  44.28267926   5.70086193 120.3547999    0.2
   0.8          0.8       ], Predicted=20.0
X=[25.87682261 45.96341933  5.8385087  38.53254678  0.3         0.7
  0.6       ], Predicted=17.0
X=[ 23.11407669  94.31994776   6.75847957 231.5153161    0.5
   0.8          0.7       ], Predicted=29.0
X=[2.95974620e+01 8.45902559e+01 5.39052580e+00 1.37370284e+02
 7.00000000e-01 8.00000000e-01 1.00000000e-01], Predicted=14.0
X=[ 22.81212536  91.51861705   6.0273144  107.855225     0.5
   0.3          0.7       ], Predicted=25.0
X=[ 15.77370214  19.2303162    5.97997397 108.3441414    0.5
   0.6          0.7       ], Predicted=16.0
X=[ 29.38254012  83.50423735   5.76530894 109.2486647    0.3
   0.7          0.5       ], Predicted=23.0
X=[20.93175255 18.91295403  6.45614847 78.06910795  0.6         0.7
  0.3       ], Predicted=22.0
X=[2.52536080e+01 6.23216814e+01 4.93744598e+00 2.38598639e+02
 8.00000000e-01 6.00000000e-01 2.00000000e-01], Predicted=4.0
X=[31.2123945  40.92604945  8.53207873 53.78769958  0.6         0.8
  0.3       ], Predicted=12.0
X=[2.54879684e+01 8.44823588e+01 6.74094764e+00 1.68784889e+02
 7.00000000e-01 7.00000000e-01 1.00000000e-01], Predicted=8.0
X=[21.44526922 63.1621551   6.1780563  65.88951188  0.7         0.6
  0.4       ], Predicted=10.0
X=[28.77653519 86.69133979  6.98313047 56.12443206  0.5         0.3
  0.7       ], Predicted=3.0
X=[ 32.98761855  56.91380716   6.72038955 157.8209829    0.2
   0.6          0.5       ], Predicted=21.0
X=[47.57373777 12.77831421  7.43866734 36.74607697  0.3         0.8
  0.7       ], Predicted=5.0
X=[24.84906168 22.89464642  5.60816519 62.21292186  0.8         0.1
  0.7       ], Predicted=16.0
X=[26.49195283 80.04678201  6.05769711 57.72799157  0.4         0.8
  0.5       ], Predicted=31.0
X=[ 22.20700989  93.50574163   6.44338291 120.1593771    0.6
   0.3          0.7       ], Predicted=25.0
X=[ 20.51343484  92.51675903   5.70008866 110.5764023    0.7
   0.2          0.8       ], Predicted=28.0
X=[28.69180475 49.47225353  5.83303171 96.36222901  0.6         0.3
  0.8       ], Predicted=26.0
X=[27.84492803 91.60666594  6.73204907 26.47844429  0.8         0.5
  0.5       ], Predicted=27.0
X=[22.41779693 69.99596284  6.05572211 82.25335771  0.7         0.7
  0.3       ], Predicted=15.0
```



```
[ 1.  1.  1. ... 31. 31. 31.]
```

In [ ]:

In [ ]: