Aaron P. Mills / Z-23547104 / Dr. Ghoraani

Intro to Deep Learning / CAP 4613 / Assignment 1 / 23 January 2022

https://colab.research.google.com/drive/1B6fn1uP9ZparFcgD4JCTnf87U3WH7iJ2

```
# Aaron P. Mills /                Z-23547104 /
# Dr. Ghoraani
# Intro to Deep Learning  /       CAP 4613 /
# Assignment 1             /      23 January 2022
# https://colab.research.google.com/drive/1B6fn1uP9ZparFcgD4JCTnf87U3WH7iJ2
##########################################################################################
#Problem 1)
import math
import numpy as np
def stars():
    print("*******************************************************************************
def isnumerical(strng):
  strng = strng.replace(' ','')
  if (len(strng)==0 or strng[len(strng)-1]=='.'):
    return False
  elif (strng[0] == '.'):
      strng = '0' + strng
  decimals = 0
  signs = 0
  for x in strng:
    if (x=='.'):
      decimals += 1
    elif ( (x=='-' or x=='+') and x==strng[0]):
      signs += 1
    elif (not (x>='0' and x<='9')):
      return False
    if (decimals >= 2 or signs >= 2 ):
      return False
  return True
#use function to perform calculation
def calculate1(num1,op):
    if(op=='exp'):
      result = math.exp(num1)
    elif (op=='ln'):
      if (num1 < 0):
        print("We can't natural log negative numbers, now can we?")
        result = 'undefined'
      elif (num1 == 0):
          result = 'undefined'
      else:
          result = np.log(num1)
    elif (op=='abs'):
      result = np.abs(num1)
    return result
```

```

def calculate2(num1,num2,op):
    if (op=='+'):
      result = num1+num2
    elif (op=='-'):
      result = num1-num2
    elif (op=='*'):
      result = num1*num2
    elif (op=='/'):
      if (num2 == 0):
        print("Can't divide by zero, now can we?")
        result = 'undefined'
      else:
          result = num1/num2
    elif (op=='mod'):
      if (num2 == 0):
        print("Can't mod by zero, now can we?")
        result = 'undefined'
      else:
          result = num1%num2
    elif (op=='pow'):
      result = num1**num2
    return result

print("Problem 1)")
x = ''
while (x != 'x'):
  stars()
  print("Simple Calculator! Operations: +,-,*,/,mod,pow,exp,ln,abs")
  num1 = input("First number: ")
  if (num1 == 'x'):
    break
  elif (isnumerical(num1)):
    num1 = float(num1)
  else:
    print("Not a valid input.")
    continue
  op = input("Operation: ")
  if (op == 'x'):
    break
  elif (op=='+' or op=='-' or op=='*' or op=='/' or op=='mod' or op=='pow'):
    num2 = input("Second number: ")
    if (num2=='x'):
      break
    elif (isnumerical(num2)):
      num2 = float(num2)
      result = calculate2(num1,num2,op)
    else:
      print("Not a valid input.")
      continue
  elif (op=='exp' or op=='ln' or op=='abs'):
```

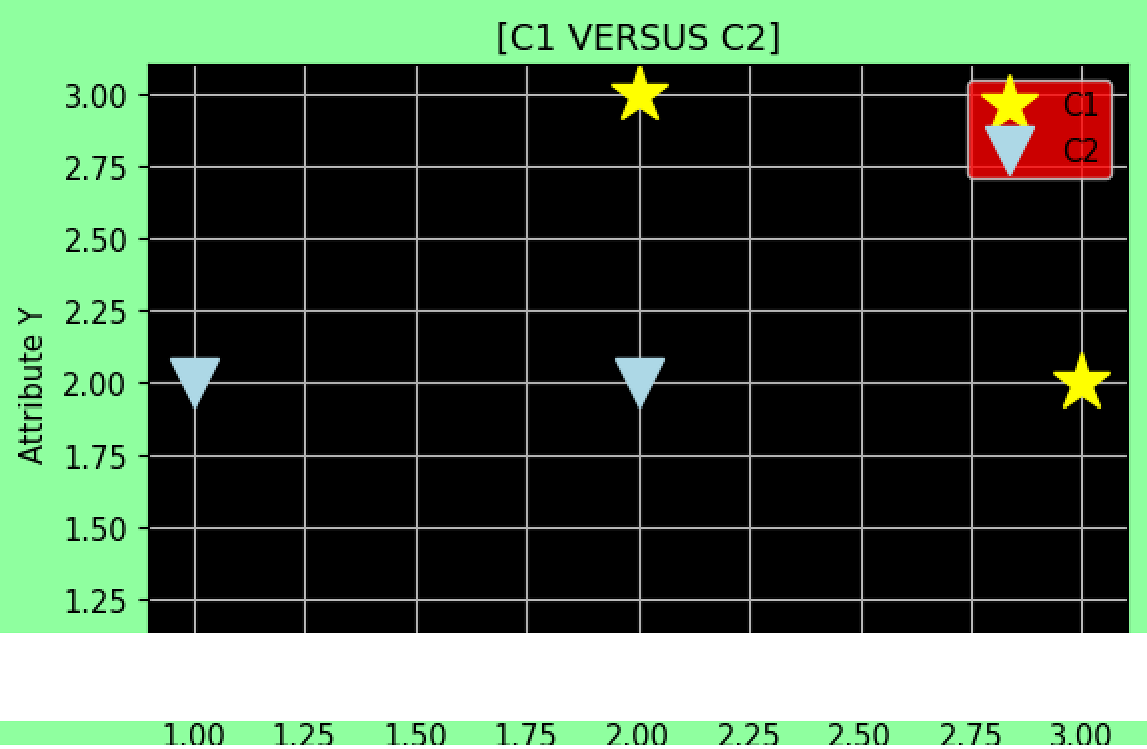
```
  result = calculate1(num1,op)
else:
  print("Not a valid input.")
  continue
print(f"Result: {result}")
```

```
Problem 1)
*************************************************************************************
Simple Calculator! Operations: +,-,*,/,mod,pow,exp,ln,abs
First number: .5
Operation: +
Second number: 1.5
Result: 2.0
*************************************************************************************
Simple Calculator! Operations: +,-,*,/,mod,pow,exp,ln,abs
First number: -7
Operation: -
Second number: 3
Result: -10.0
*************************************************************************************
Simple Calculator! Operations: +,-,*,/,mod,pow,exp,ln,abs
First number: 4
Operation: *
Second number: 11
Result: 44.0
*************************************************************************************
Simple Calculator! Operations: +,-,*,/,mod,pow,exp,ln,abs
First number: 6
Operation: /
Second number: 10
Result: 0.6
*************************************************************************************
Simple Calculator! Operations: +,-,*,/,mod,pow,exp,ln,abs
First number: 21
Operation: mod
Second number: 2
Result: 1.0
*************************************************************************************
Simple Calculator! Operations: +,-,*,/,mod,pow,exp,ln,abs
First number: 3
Operation: pow
Second number: 3
Result: 27.0
*************************************************************************************
Simple Calculator! Operations: +,-,*,/,mod,pow,exp,ln,abs
First number: 100
Operation: exp
Result: 2.6881171418161356e+43
*************************************************************************************
Simple Calculator! Operations: +,-,*,/,mod,pow,exp,ln,abs
First number: 1
Operation: ln
Result: 0.0
*************************************************************************************
Simple Calculator! Operations: +,-,*,/,mod,pow,exp,ln,abs
```

```
First number: -69.21
Operation: abs
Result: 69.21
*************************************************************************************
Simple Calculator! Operations: +,-,*,/,mod,pow,exp,ln,abs
First number: x
```

Double-click (or enter) to edit

```
#Problem 2)
import matplotlib.pyplot as plt
def isnumerical(strng):
  strng = strng.replace(' ','')
  if (len(strng)==0 or strng[len(strng)-1]=='.'):
    return False
  elif (strng[0] == '.'):
      strng = '0' + strng
  decimals = 0
  signs = 0
  for x in strng:
    if (x=='.'):
      decimals += 1
    elif ( (x=='-' or x=='+') and x==strng[0]):
      signs += 1
    elif (not (x>='0' and x<='9')):
      return False
    if (decimals >= 2 or signs >= 2 ):
      return False
  return True

def plotter(C1,C2):
    #plot data
    fig,ax = plt.subplots(dpi=105)      #dpi = dot per inch
    ax.plot(*map(list,zip(*C1)),color='yellow',marker='*',markersize='20',linestyle=' ',la
    ax.plot(*map(list,zip(*C2)),color='lightblue',marker='v',markersize='16',linestyle=' '

    #Explanation
    #zip(): breaks list of tuples into two separate collections without a datatype: [(x1,x
    #map(f,x): apply function f onto all values of x; map(list,zip(*C1)): turns [(x1,x2,x3
    #*map(f,x): * is an iterator: breaks apart the list returned from map using the comma:

    #plot attributes
    ax.set_xlabel('Attribute X')
    ax.set_ylabel('Attribute Y')
    ax.set_title("[C1 VERSUS C2]")
    ax.grid()
    ax.legend()
    ax.legend(facecolor='red')
    fig.patch.set_facecolor('xkcd:mint green')
    ax.set_facecolor('xkcd:black')
    plt.show()
    plt.show()
plotter(C1,C2)


#data - list of tuples
C1 = [(1,1),(3,2),(2,3)]
C2 = [(1,2),(2,2),(2,1)]

# calculate classification accuracy based on user provided thresholds for a total of 3 tim
print('Problem 2)')
total_data = len(C1)+len(C2)
for z in range(3):
    print(f"Prediction {z+1}************************************************************
    print("Please enter the required fields:")
    thx = input("Threshold x: ")
    thy = input("Threshold y: ")
    #error checking
    if (isnumerical(thx) and isnumerical(thy)):
        thx = float(thx)
        thy = float(thy)
    else:
        print("What a pity! One of your inputs is invalid!")
        continue
    corr_pred = 0
    for x,y in C1:
        if (x > thx and y > thy):
            corr_pred += 1
    for x,y in C2:
        if not (x > thx and y > thy):
            corr_pred += 1
    print(f'Classification accuracy: {round((corr_pred/total_data)*100,2)}%')     #classif
```



011/

Part e) Judging by my results, I deem [(2,1),(1,1),(1,2)] to be among one of the threshold sets that can reach the highest possible classification accuracy, which is about 66.67%.

I feel it is also important to note that many pairs of decimal numbers may additionally reach this accuracy.

```
Classification accuracy: 66.67%
Prediction 2*************************************************************
Please enter the required fields:
Threshold x: 1
Threshold y: 1
Classification accuracy: 66.67%
Prediction 3*************************************************************
Please enter the required fields:
Threshold x: 1
Threshold y: 2
Classification accuracy: 66.67%
```

✓ 8s completed at 10:27 PM