Brazos Fitch - Pattern Recognition Project - Coin Classifier

The purpose of this code is to create a function which can identify all of the coins in an image and then find their colors, and radius. Then populate an array with this information and include an empty list for the classification of the coins.

```
### First we have to import all the necessary libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import sklearn
from sklearn.cluster import KMeans
import collections
from collections import Counter


### this is just part of the process to get the most dominant color
def RGB2HEX(color):
  return "#{:02x}{:02x}{:02x}".format(int(color[0]),int(color[1]), int(color[2]))


def coincutter(image,coin):

  ### Coins are as follow, dime: 1, penny: 2, Nickel: 3, Quarter: 4

  ### load in an image and make some copies of it for other use
  image0 = cv2.imread(image)
  image0 = cv2.cvtColor(image0, cv2.COLOR_BGR2RGB)
  image1 = image0.copy()
  image2 = image0.copy()
  image3 = image0.copy()

  ### Next, we need to flip an image to GRAY in order to find all the circles
  ### It will also be helpful to blur it slightly

  image0 = cv2.cvtColor(image0, cv2.COLOR_RGB2GRAY)
  #image0 = cv2.GaussianBlur(image0, (21,21), cv2.BORDER_DEFAULT)

  ### Go ahead and find the circles in the image
  all_circs = cv2.HoughCircles(image0, cv2.HOUGH_GRADIENT, 0.92, 115, param1 = 80, param2 = 65, minRadius = 115, maxRadius = 170)
  all_circs_rounded = np.uint(np.around(all_circs))
  X_test = np.arange(all_circs_rounded.shape[1]*3).reshape(all_circs_rounded.shape[1],3)
  X_test[:][:] = 0

  ### We can also print out the circles that are drawn
  count = 1
  for i in all_circs_rounded[0, :]:
    cv2.circle(image1, (i[0],i[1]),i[2],(50,200,200),5)
    cv2.circle(image1,(i[0],i[1]),2,(255,0,0),3)
    cv2.putText(image1,'Coin ' + str(count), (int(i[0]-70),int(i[1]+30)), cv2.FONT_HERSHEY_SIMPLEX, 1.1, (255,0,0),2)
    count += 1
  plt.rcParams["figure.figsize"] = (16,9)
  plt.imshow(image1)

  ### And next we will cycle through each coin and take it's radius and size
  ### we can put them in positions 1 and 2 of the array X_test

  for i in range(0,all_circs_rounded.shape[1]):

    y1 = ((all_circs_rounded[0][i])[0])-((all_circs_rounded[0][i])[2])
    y2 = ((all_circs_rounded[0][i])[0])+((all_circs_rounded[0][i])[2])
    x2 = ((all_circs_rounded[0][i])[1])+((all_circs_rounded[0][i])[2])
    x1 = ((all_circs_rounded[0][i])[1])-((all_circs_rounded[0][i])[2])

    cropped = image2[x1:x2,y1:y2]
    #plt.imshow(cropped)

    ### Here is where we will drop the radius into X_test, hopefully this is right
    X_test[i][0] = all_circs_rounded[0][i][2]

    ### Next is to pull the most dominant color
    ### To do this: We are going to find the most dominant color
    ### Then we will convert it to RGB, to HEX, and then to Decimal
    radius = int(all_circs_rounded[0][i][2])
    modcrop = cropped.reshape((2*radius)**2,3)
    clf = KMeans(n_clusters = 1)
    labels = clf.fit_predict(modcrop)

    counts = Counter(labels)
    center_colors = clf.cluster_centers_
    ordered_colors = [center_colors[i] for i in counts.keys()]
    hex_colors = [RGB2HEX(ordered_colors[i]) for i in counts.keys()]

    ### We now have a list which has the most dominant color as a hex value string
    ### We need to extract that

    colval = hex_colors[0]
    colval = colval[1:]
    X_test[i][1] = int(colval,16)

  if(coin > 0):
    for i in range(0,all_circs_rounded.shape[1]):
      X_test[i][2] = coin


  return(X_test)

tst1 = coincutter("attemptcoins1.jpg")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-2-9b35312dce7b> in <module>()
----> 1 tst1 = coincutter("attemptcoins1.jpg")

TypeError: coincutter() missing 1 required positional argument: 'coin'
```

SEARCH STACK OVERFLOW

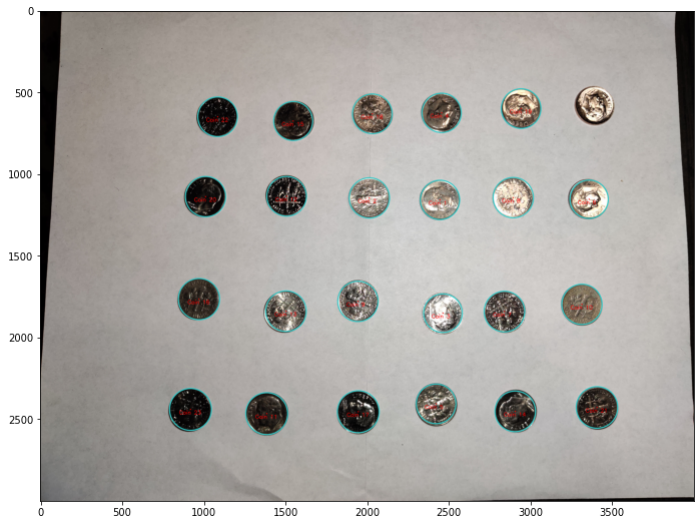
```
tst1
```

```
tst1[0]
```

```
tst1[0][1]
```

```
### So, tst[0] notates the first coin in the classifier
### and tst[0][0] notates the first coins radius
### and tst[0][1] notates the first coins color valule
### and tst[0][2] notates the classification of the first coin
```

```
tst2 = coincutter("24dimes.jpg",1)
```



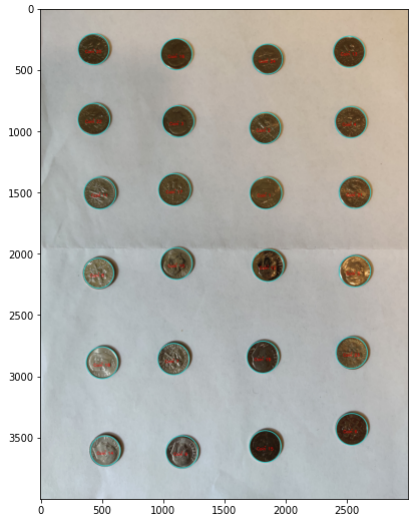

```
tst2
```

```
array([[     120, 11182487,        1],
       [     123, 10919573,        1],
       [     118, 12039087,        1],
       [     120,  7892066,        1],
       [     121,  8024938,        1],
       [     117, 12431777,        1],
       [     124,  7827306,        1],
       [     123, 13287089,        1],
       [     123,  9341058,        1],
       [     123,  9076334,        1],
       [     128,  4603963,        1],
       [     121,  5261895,        1],
       [     124,  8617077,        1],
       [     124,  5591372,        1],
       [     119,  4603192,        1],
       [     126,  5129792,        1],
       [     128,  4802372,        1],
       [     121,  8680551,        1],
       [     121, 10654081,        1],
       [     123,  3814707,        1],
       [     127,  5590343,        1],
       [     121,  3485743,        1],
       [     128,  3157291,        1]])
```

```
tst2.shape
```

```
(23, 3)
```

```
tst22 = coincutter("24dimes2.jpg",1)
```



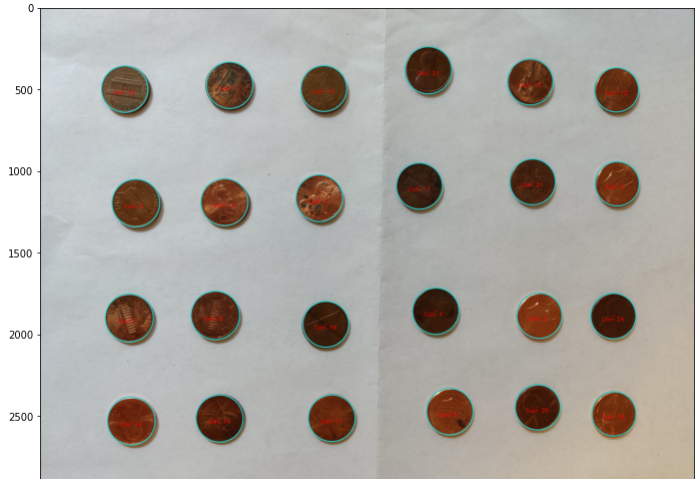

```
tst22
```

```
array([[    130, 6969148,       1],
       [    126, 6837564,       1],
       [    128, 5852217,       1],
       [    128, 7296829,       1],
       [    128, 9597519,       1],
       [    129, 5390644,       1],
       [    128, 7169882,       1],
       [    122, 8156770,       1],
       [    131, 6841180,       1],
       [    129, 6643020,       1],
       [    130, 6906450,       1],
       [    130, 6048570,       1],
       [    127, 6902071,       1],
       [    122, 8223602,       1],
       [    129, 4471602,       1],
       [    127, 6048565,       1],
       [    131, 6444352,       1],
       [    130, 9078907,       1],
       [    130, 5392446,       1],
       [    126, 5720888,       1],
       [    128, 7625536,       1],
       [    130, 5852731,       1],
       [    128, 7298630,       1],
       [    122, 6573879,       1]])
```

```
tst22.shape
```

```
(24, 3)
```

```
tst3 = coincutter("24pennies.jpg",2)
```



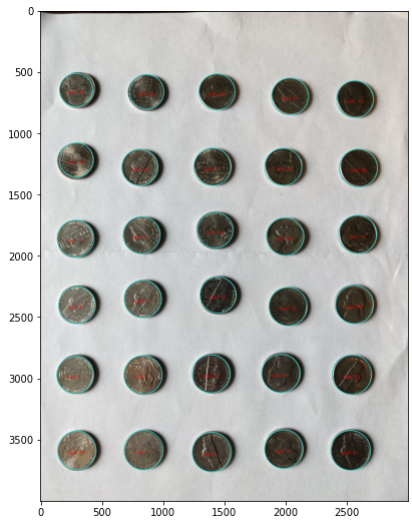

```
tst3
```

```
array([[    142, 6967358,       2],
       [    138, 8998192,       2],
       [    143, 8542016,       2],
       [    142, 4863529,       2],
       [    135, 8145456,       2],
       [    142, 8407600,       2],
       [    147, 7622459,       2],
       [    148, 6701875,       2],
       [    146, 7228982,       2],
       [    140, 6441522,       2],
       [    143, 7159594,       2],
       [    135, 7357999,       2],
       [    144, 8540732,       2],
       [    146, 8341555,       2],
       [    140, 7096628,       2],
       [    144, 5978152,       2],
       [    141, 4798508,       2],
       [    135, 8669996,       2],
       [    140, 4928806,       2],
       [    138, 5978152,       2],
       [    140, 5783087,       2],
       [    142, 7495500,       2],
       [    144, 5718322,       2],
       [    139, 5519659,       2]])
```

```
tst3.shape
```

```
(24, 3)
```

```
tstN = coincutter("30nickels.jpg",3)
```



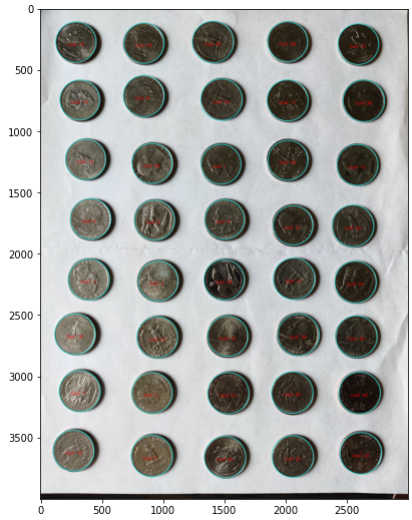

```
tstN
```

```
array([[    157, 5130560,       3],
       [    160, 6051148,       3],
       [    156, 8088927,       3],
       [    156, 5064769,       3],
       [    152, 7433569,       3],
       [    150, 8683119,       3],
       [    160, 7695713,       3],
       [    150, 8617844,       3],
       [    150, 6775641,       3],
       [    157, 9536627,       3],
       [    150, 8157549,       3],
       [    156, 4999231,       3],
       [    147, 4341819,       3],
       [    144, 5196609,       3],
       [    150, 5064251,       3],
       [    155, 5721929,       3],
       [    152, 5985867,       3],
       [    155, 5261371,       3],
       [    155, 5195838,       3],
       [    153, 5656902,       3],
       [    162, 4604220,       3],
       [    139, 6972760,       3],
       [    161, 5458496,       3],
       [    147, 6709590,       3],
       [    140, 5722952,       3],
       [    151, 5393215,       3],
       [    150, 5787717,       3],
       [    155, 5392959,       3],
       [    147, 5459009,       3],
       [    135, 6644054,       3]])
```

```
tstN.shape
```

```
(30, 3)
```

```
testQ = coincutter("40quarters.jpg",4)
```



testQ

```
array([[    162, 4998972,       4],
       [    169, 8089695,       4],
       [    162, 7696483,       4],
       [    160, 5525830,       4],
       [    165, 7760987,       4],
       [    163, 7170140,       4],
       [    168, 9538175,       4],
       [    167, 6249296,       4],
       [    165, 8486002,       4],
       [    164, 5853510,       4],
       [    167, 8815479,       4],
       [    167, 7629918,       4],
       [    167, 4538422,       4],
       [    162, 7565414,       4],
       [    164, 8157552,       4],
       [    166, 8353643,       4],
       [    167, 4932666,       4],
       [    165, 9538431,       4],
       [    164, 6183762,       4],
       [    169, 4867390,       4],
       [    168, 5524034,       4],
       [    169, 5591111,       4],
       [    169, 5656390,       4],
       [    169, 6445649,       4],
       [    164, 5722694,       4],
       [    162, 4538164,       4],
       [    160, 4406579,       4],
       [    162, 4669494,       4],
       [    164, 4867393,       4],
       [    163, 6511954,       4],
       [    169, 5525577,       4],
       [    169, 5065024,       4],
       [    169, 4734776,       4],
       [    169, 4669752,       4],
       [    169, 5788489,       4],
       [    169, 5195840,       4],
       [    166, 4603702,       4],
       [    169, 4274994,       4],
       [    169, 4932925,       4],
       [    169, 3748398,       4]])
```

testQ.shape

```
(40, 3)
```

As you can see, I have gathered some training data for the classifier.
Now I am going to append them all together

```
X_train = np.concatenate((tst2,tst22,tst3,tstN,testQ),axis = 0)
X_train
```

```
[     147,  4341819,        3],
[     144,  5196609,        3],
[     150,  5064251,        3],
[     155,  5721929,        3],
[     152,  5985867,        3],
[     155,  5261371,        3],
[     155,  5195838,        3],
[     153,  5656902,        3],
[     162,  4604220,        3],
[     139,  6972760,        3],
[     161,  5458496,        3],
[     147,  6709590,        3],
[     140,  5722952,        3],
[     151,  5393215,        3],
[     150,  5787717,        3],
[     155,  5392959,        3],
[     147,  5459009,        3],
[     135,  6644054,        3],
[     162,  4998972,        4],
[     169,  8089695,        4],
[     162,  7696483,        4],
[     160,  5525830,        4],
[     165,  7760987,        4],
[     163,  7170140,        4],
[     168,  9538175,        4],
[     167,  6249296,        4],
[     165,  8486002,        4],
[     164,  5853510,        4],
[     167,  8815479,        4],
[     167,  7629918,        4],
[     167,  4538422,        4],
[     162,  7565414,        4],
[     164,  8157552,        4],
[     166,  8353643,        4],
[     167,  4932666,        4],
[     165,  9538431,        4],
[     164,  6183762,        4],
[     169,  4867390,        4],
[     168,  5524034,        4],
[     169,  5591111,        4],
[     169,  5656390,        4],
[     169,  6445649,        4],
[     164,  5722694,        4],
[     162,  4538164,        4],
[     160,  4406579,        4],
[     162,  4669494,        4],
[     164,  4867393,        4],
[     163,  6511954,        4],
[     169,  5525577,        4],
[     169,  5065024,        4],
[     169,  4734776,        4],
[     169,  4669752,        4],
[     169,  5788489,        4],
[     169,  5195840,        4],
[     166,  4603702,        4],
[     169,  4274994,        4],
[     169,  4932925,        4],
[     169,  3748398,        4]])
```

```
### I'm just gonna start by throwing it at a NeuralNetwork
from sklearn import datasets
from sklearn import preprocessing
from sklearn import neural_network

X = X_train[:,[0,1]]
y = X_train[:,[2]]



clf = neural_network.MLPClassifier(hidden_layer_sizes=(5),max_iter = 10000, random_state = 42)
clf.fit(X,y)
clf.score(X,y)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:934: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please ch
  y = column_or_1d(y, warn=True)
0.28368794326241137
```

```
### The neural network didn't work too well. Lets try nearest neighbors
import pandas as pd
import seaborn as sns
#from sklearn import trees
from sklearn.neighbors import KNeighborsClassifier

clf1 = KNeighborsClassifier()
clf1.fit(X,y)
clf1.score(X,y)
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead
  import pandas.util.testing as tm
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_sam
  import sys
0.5602836879432624
```

```
### now we try naive bayes

from sklearn import naive_bayes
clf2 = naive_bayes.GaussianNB()
clf2.fit(X,y)
clf2.score(X,y)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/naive_bayes.py:206: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_
  y = column_or_1d(y, warn=True)
0.4326241134751773
```

```
from sklearn.tree import DecisionTreeClassifier
import sklearn.tree

clf3 = DecisionTreeClassifier(max_depth = 3)
clf3.fit(X,y)
clf3.score(X,y)
```

```
0.9361702127659575
```

```
#### Now that we know that the Decision tree works, we can go ahead and try to use it.

X_test = coincutter("classifythis1.jpg",0)
```


```
### Creating the test data
X_test = X_test[:,[0,1]]
X_test
```

```
array([[    163, 4603957],
       [    162, 7432013],
       [    138, 6833460],
       [    152, 6839887],
       [    135, 8934970],
       [    130, 7299657],
       [    133, 4206633],
       [    128, 5918528],
       [    124, 8484448]])
```

```
results = clf3.predict(X_test)
```

```
results.shape
```

```
(9,)
```

```
total = 0
for i in range(0,results.shape[0]):
  if(results[i] == 4):
    total += 0.25
  if(results[i] == 3):
    total += 0.05
  if(results[i] == 2):
    total += 0.01
  if(results[i] == 1):
    total += 0.10
print(total)
```

```
1.01
```

```
### The real total is 0.88, so this example is not perfect.

def getmoney(image):
  X_coin = coincutter(image,0)
  X_coin = X_coin[:,[0,1]]
  results = clf3.predict(X_coin)
  amount = 0
  for i in range(0,results.shape[0]):
    if(results[i] == 4):
      amount += 0.25
    if(results[i] == 3):
      amount += 0.05
    if(results[i] == 2):
      amount += 0.01
    if(results[i] == 1):
      amount += 0.10
  print(amount)
  print(X_coin,results)


getmoney("classifythis1.jpg")
```

```
0.97
[[    163 4603957]
 [    162 7432013]
 [    138 6833460]
 [    152 6839887]
 [    135 8934970]
 [    130 7299657]
 [    133 4206633]
 [    128 5918528]
 [    124 8484448]] [4 4 2 3 2 1 1 1 1]
```

```
getmoney("classifythis2.jpg")
```

```
1.3800000000000003
[[    169 7760987]
 [    169 6839622]
 [    169 5852989]
 [    132 5649962]
 [    135 5718324]
 [    132 8542269]
 [    138 7491133]
 [    130 9920577]
 [    130 7553581]
 [    135 4797228]
 [    131 4795941]
 [    128 4209200]] [4 4 4 1 2 1 2 1 1 2 1 1]
```


```
### The real total for this image is 0.93, so this is inaccurate

### Lets try again with one of  the larger images
getmoney("24pennies.jpg")
```

```
0.3200000000000002
[[    142 6967358]
 [    138 8998192]
 [    143 8542016]
 [    142 4863529]
 [    135 8145456]
 [    142 8407600]
 [    147 7622459]
 [    148 6701875]
 [    146 7228982]
 [    140 6441522]
 [    143 7159594]
 [    135 7357999]
 [    144 8540732]
 [    146 8341555]
 [    140 7096628]
 [    144 5978152]
 [    141 4798508]
 [    135 8669996]
 [    140 4928806]
 [    138 5978152]
 [    140 5783087]
```

```
getmoney("classifythis3.jpg")
```

```
1.07
[[    158 7958620]
 [    164 5919305]
 [    161 4340275]
 [    164 6906709]
 [    128 6373684]
 [    133 5324341]
 [    158 4274477]
 [    142 5064254]
 [    128 8670774]
 [    144 5984834]
 [    120 6182470]] [3 4 3 4 1 1 3 2 1 2 1]
```