Thasina Tabashum

1: Download the images to be used in this exam. Calculate the percentages of red and blue for all 10 lava images and all 10 ocean images, and record the results into 4 lists:

In [58]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
from PIL import Image
```

In [59]:
```
image_files= ['lava1.jpg', 'lava2.jpg', 'lava3.jpg', 'lava4.jpg', 'lava5.jpg',
 'lava6.jpg', 'lava7.jpg', 'lava8.jpg', 'lava9.jpg', 'lava10.jpg',
 'ocean1.jpg', 'ocean2.jpg', 'ocean3.jpg', 'ocean4.jpg', 'ocean5.jpg',
 'ocean6.jpg', 'ocean7.jpg', 'ocean8.jpg', 'ocean9.jpg', 'ocean10.jpg']
```

In [60]:
```
percent_red_train = []
percent_blue_train = []
for image_name in image_files:
    img = mpimg.imread("D:\\Courses\\BigData\\Exams\\images\\images\\"+image_nam
    RGBtuple = np.array(img).mean(axis=(0,1))
    RGBsum = RGBtuple[0] + RGBtuple[1] + RGBtuple[2]
    percent_red_train.append(RGBtuple[0] / RGBsum)
    percent_blue_train.append(RGBtuple[2] / RGBsum)
```

In [61]:
```
lava_red= percent_red_train[0:10]
ocean_red = percent_red_train[10:20]


lava_blue = percent_blue_train[0:10]
ocean_blue = percent_blue_train[10:20]
```

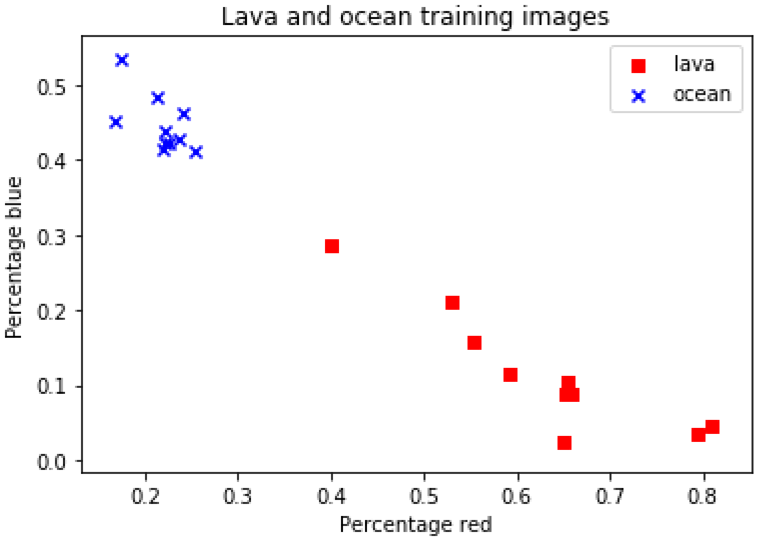
In [62]:
```
print(" lava_red: ",
      lava_red)
print(" lava_blue: ",
      lava_blue)
print("ocean_red: ",
      ocean_red)
print("ocean_blue: ",
      ocean_blue)
```

```
 lava_red:  [0.6504504681674602, 0.7938890724215069, 0.651948833530485, 0.65495
15004700861, 0.658924309536722, 0.5521116450467187, 0.5910228806617933, 0.81035
89245632462, 0.5298585353214215, 0.4010770327400418]
 lava_blue:  [0.02471310190925999, 0.033606755917978334, 0.08833098058697823,
0.10534800283962736, 0.08879121888272326, 0.1585450614868838, 0.113886059205939
61, 0.04576780242563183, 0.21222114097910563, 0.28477046774708864]
ocean_red:  [0.2532488530948958, 0.24185924549490984, 0.1749370368358937, 0.220
79358776441338, 0.22702298042892918, 0.22235946733887024, 0.2361137563096708,
0.16839765966636838, 0.21957947692637475, 0.21298596530777492]
ocean_blue:  [0.41058888142309286, 0.4611309134903137, 0.5344407794090835, 0.42
269533340781446, 0.42280850814274756, 0.43710213665803965, 0.4267563337983969,
0.45064805175500783, 0.4146312351365986, 0.48239423079340416]
```

2: Create a scatter plot where each image is a point. The x-axis is the percentage of red, and the y-axis is the percentage of blue. Be sure (as always) to give appropriate titles and axis labels, and provide a legend distinguishing the lava picture values from the ocean picture values

In [63]:
```
plt.scatter(lava_red, lava_blue, marker='s', c='red')
plt.scatter(ocean_red, ocean_blue, marker='x', c='blue')

# Show the boundary between the regions:
plt.title('Lava and ocean training images')
plt.xlabel('Percentage red')
plt.ylabel('Percentage blue')
plt.legend(['lava','ocean'])
plt.show()
```

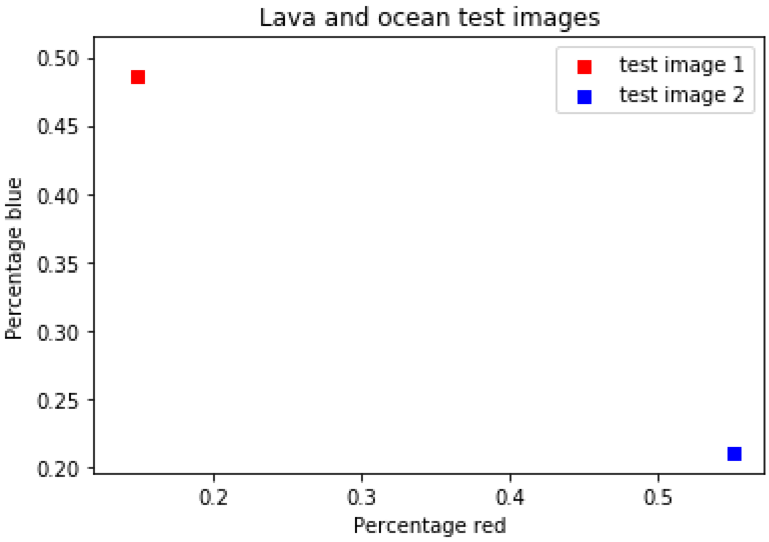


3: Print the percent red and blue in the test images (test1.jpg and test2.jpg) and compare to the plot you just made. Indicate how you would expect each image to be classified, and compare that with what the images actually are. Use comments or print statements to inline this in the code

```
In [64]: image_test_files = ['test1.jpg','test2.jpg']
```

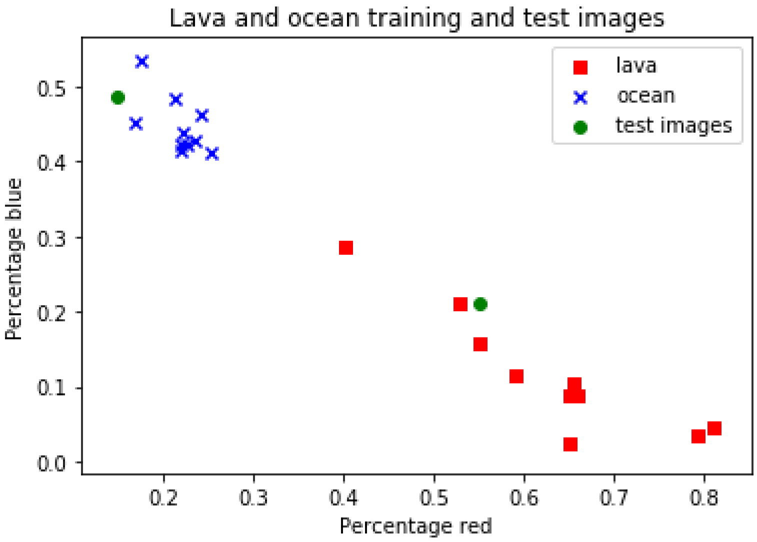
```
In [65]: percent_red = []
         percent_blue = []
         for image_name in image_test_files:
             img = mpimg.imread("D:\\Courses\\BigData\\Exams\\images\\images\\"+image_nam
             RGBtuple = np.array(img).mean(axis=(0,1))
             RGBsum = RGBtuple[0] + RGBtuple[1] + RGBtuple[2]
             percent_red.append(RGBtuple[0] / RGBsum)
             percent_blue.append(RGBtuple[2] / RGBsum)
```

```
In [66]: plt.scatter(percent_red[0], percent_blue[0], marker='s', c='red')
         plt.scatter(percent_red[1], percent_blue[1], marker='s', c='blue')
         # Show the boundary between the regions:
         plt.title('Lava and ocean test images')
         plt.xlabel('Percentage red')
         plt.ylabel('Percentage blue')
         plt.legend(['test image 1','test image 2'])
         plt.show()
```



In [67]:

```
plt.scatter(lava_red, lava_blue, marker='s', c='red')
plt.scatter(ocean_red, ocean_blue, marker='x', c='blue')
plt.scatter(percent_red[0], percent_blue[0], marker='o', c='green')
plt.scatter(percent_red[1], percent_blue[1], marker='o', c='green')
# Show the boundary between the regions:
plt.title('Lava and ocean training and test images')
plt.xlabel('Percentage red')
plt.ylabel('Percentage blue')
plt.legend(['lava','ocean','test images'])
plt.show()
```



From the figure I think test image 1 should be classified as ocean and test image 2 should be lava

4: Use the percentages of red and blue from those 10 lava images and 10 ocean images as your training data to build a k-NN classifier and predict the class of the 2 test images

In [68]:
```
training_data = np.zeros((20, 2))
training_data
```

Out[68]:
```
array([[0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.]])
```

In [69]:
```
for i in range(0,20):
    training_data[i][0] = percent_red_train[i]
    training_data[i][1] = percent_blue_train[i]
training_data
```

Out[69]:
```
array([[0.65045047, 0.0247131 ],
       [0.79388907, 0.03360676],
       [0.65194883, 0.08833098],
       [0.6549515 , 0.105348  ],
       [0.65892431, 0.08879122],
       [0.55211165, 0.15854506],
       [0.59102288, 0.11388606],
       [0.81035892, 0.0457678 ],
       [0.52985854, 0.21222114],
       [0.40107703, 0.28477047],
       [0.25324885, 0.41058888],
       [0.24185925, 0.46113091],
       [0.17493704, 0.53444078],
       [0.22079359, 0.42269533],
       [0.22702298, 0.42280851],
       [0.22235947, 0.43710214],
       [0.23611376, 0.42675633],
       [0.16839766, 0.45064805],
       [0.21957948, 0.41463124],
       [0.21298597, 0.48239423]])
```

In [70]:

```
training_target =[]
for i in range(0,10):
    training_target.append('lava')
for i in range(10,20):
    training_target.append('ocean')
training_target
```

Out[70]:

```
['lava',
 'lava',
 'lava',
 'lava',
 'lava',
 'lava',
 'lava',
 'lava',
 'lava',
 'lava',
 'ocean',
 'ocean',
 'ocean',
 'ocean',
 'ocean',
 'ocean',
 'ocean',
 'ocean',
 'ocean',
 'ocean']
```

In [71]:

```
from sklearn import neighbors

#k-NN classifier for k=1
#By using ‘distance’, closer neighbors will have greater weight #than further one
k1 = neighbors.KNeighborsClassifier(n_neighbors=1, weights='distance')
```

In [72]:

```
k1.fit(training_data,training_target )
```

Out[72]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='distance')
```

In [73]:

```
test_data = np.zeros((2,2))
```

In [74]:

```
for i in range(0,2):
    test_data[i][0] = percent_red[i]
    test_data[i][1] = percent_blue[i]
test_data
```

Out[74]:

```
array([[0.14909368, 0.48658471],
       [0.55128925, 0.21099691]])
```

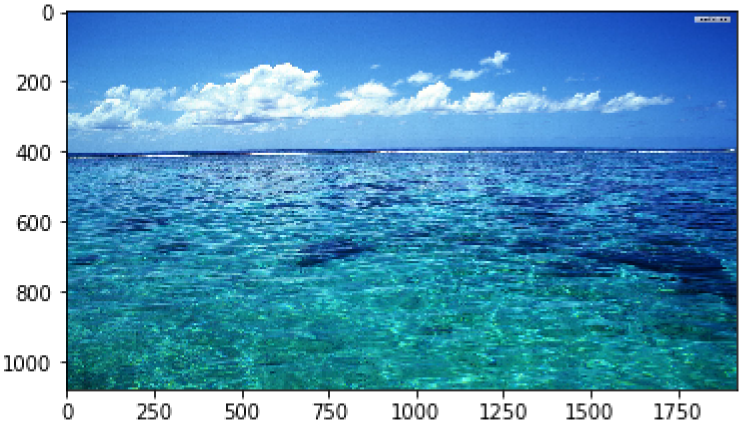
In [75]:
```
k1_pred = k1.predict(test_data)
print(k1_pred)
```

```
['ocean' 'lava']
```

In [76]:
```
from sklearn.metrics import accuracy_score, recall_score, average_precision_scor
test_target = ['ocean','lava']
accuracy_k1 = accuracy_score(test_target,k1_pred)
print('Accuracy for k=1')
print(accuracy_k1*100)
```

```
Accuracy for k=1
100.0
```

In [77]:
```
test1 = mpimg.imread('D:\\Courses\\BigData\\Exams\\images\\images\\test1.jpg')
#Plots the image of a beach from the array data
test1 = plt.imshow(test1)
```



In [78]:
```
test2 = mpimg.imread('D:\\Courses\\BigData\\Exams\\images\\images\\test2.jpg')
#Plots the image of a beach from the array data
test2 = plt.imshow(test2)
```