Assignment 9: Image Classification

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
```

1. Create a list with the names called image_files

In [12]:
```
image_files = ['farm1.jpg','farm2.jpg','farm3.jpg','farm4.jpg','farm5.jpg',
 'city1.jpg','city2.jpg','city3.jpg','city4.jpg','city5.jpg','city6.jpg','c
 'desert1.jpg','desert2.jpg','desert3.jpg','desert4.jpg','desert5.jpg','des
 'desert8.jpg']
print("image_files \n")
for p in image_files:
    print(p)
```

```
image_files

farm1.jpg
farm2.jpg
farm3.jpg
farm4.jpg
farm5.jpg
farm6.jpg
farm7.jpg
farm8.jpg
city1.jpg
city2.jpg
city3.jpg
city4.jpg
city5.jpg
city6.jpg
city7.jpg
city8.jpg
desert1.jpg
desert2.jpg
desert3.jpg
desert4.jpg
desert5.jpg
desert6.jpg
desert7.jpg
desert8.jpg
```

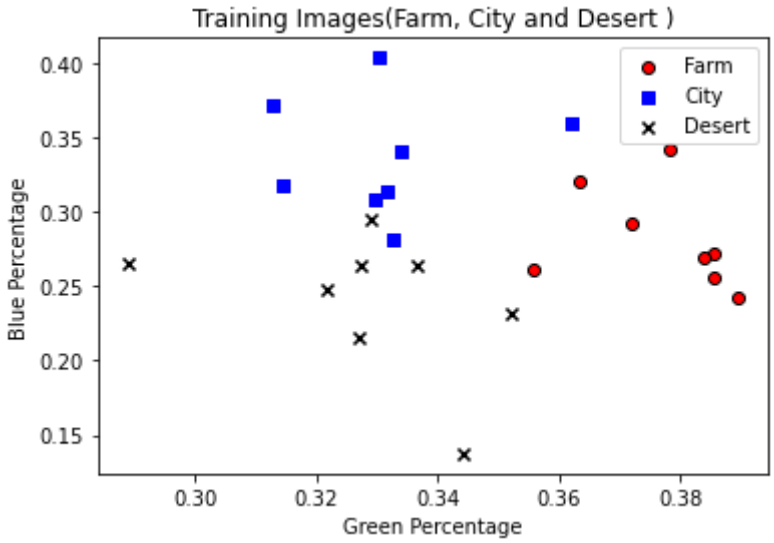
2. Create the scatter plot in the first page

In [15]:
```
perc_Green = []
perc_Blue = []
for imageName in image_files:

    image = Image.open('images2/' + imageName)
    imgclr = np.array(image).mean(axis=(0,1))
    red = imgclr[0]
    green = imgclr[1]
    blue = imgclr[2]
    color = red+ green + blue
    perc_Green.append(green/color)
    perc_Blue.append(blue/color)

plt.scatter(perc_Green[0:8],perc_Blue[0:8],marker='o',c='red',ec='black',la
plt.scatter(perc_Green[8:16],perc_Blue[8:16],marker='s',c='blue',label='Cit
plt.scatter(perc_Green[16:24],perc_Blue[16:24],marker='x',c='green',ec='bla

plt.xlabel('Green Percentage')
plt.ylabel('Blue Percentage')
plt.title('Training Images(Farm, City and Desert )')
plt.legend(loc='best')
plt.show()
```



3. Now create an array of strings called training_target with the category of each.

In [18]:
```
training_target = ['farm', 'farm', 'farm', 'farm',
'farm', 'farm', 'farm', 'farm',
'city', 'city', 'city', 'city',
'city', 'city', 'city', 'city',
'desert', 'desert', 'desert', 'desert',
'desert', 'desert', 'desert', 'desert']
print('Training target data : \n')
for p in training_target:
    print(p)
```

```
Training target data :

farm
farm
farm
farm
farm
farm
farm
farm
city
city
city
city
city
city
city
city
desert
desert
desert
desert
desert
desert
desert
desert
```

4. Create an empty array of zeros called training_data that will eventually store the percent green and percent blue values.

In [19]:
```
training_data = np.zeros([24,2])
```

5. Now fill the training_data array with the proper values for each image and observe the values in the array after it is finished.

In [21]:
```
traning_data1 = np.column_stack((perc_Green,perc_Blue))
print('Traningdata values are :  \n')
print(traning_data1)
```

```
Traningdata values are :

[[0.38537916 0.27250258]
 [0.38947877 0.2416675 ]
 [0.37176749 0.2923693 ]
 [0.38534941 0.25567274]
 [0.38368854 0.26974449]
 [0.37822351 0.34243724]
 [0.35577841 0.26138973]
 [0.36318264 0.32079251]
 [0.33384679 0.33987008]
 [0.31457989 0.31740955]
 [0.32982159 0.30761097]
 [0.33021422 0.40329483]
 [0.31267745 0.37068047]
 [0.3620055  0.35922372]
 [0.33263931 0.28122414]
 [0.33155648 0.31387494]
 [0.28899154 0.26478622]
 [0.32887465 0.29461288]
 [0.32171351 0.24749944]
 [0.35209261 0.23171261]
 [0.32718513 0.21564911]
 [0.33655681 0.2638719 ]
 [0.34419192 0.13749538]
 [0.32732039 0.26438328]]
```

6. Create your classifier.

In [23]:
```
from sklearn import neighbors
from sklearn import metrics
p1 = neighbors.KNeighborsClassifier(n_neighbors=1,weights='distance')
```

7. Train your classifier.

In [24]:
```
p1.fit(traning_data,training_target)
```

Out[24]:
```
KNeighborsClassifier(n_neighbors=1, weights='distance')
```

8. Now create an empty test_data array and fill it with the proper values for each test image and observe the filled array and consider if it matches your expectations based on your observations of the images.

In [27]:
```
testing_data_images=['test1.jpg','test2.jpg','test3.jpg']
test_data = []
test_data_green=[]
test_data_blue=[]
for imageName in testing_data_images:
    image1 = Image.open('images2/' + imageName)
    imageColors1 = np.array(image1).mean(axis=(0,1))
    red = imageColors1[0]
    green = imageColors1[1]
    blue = imageColors1[2]
    c = red + green + blue
    test_data_green.append(green/c)
    test_data_blue.append(blue/c)
test_data = np.column_stack((test_data_green,test_data_blue))
print('Green and Blue percentage of images data is as below \n')
for p in test_data:
    print(p)
```

```
Green and Blue percentage of images data is as below

[0.3269592  0.32688513]
[0.33429384 0.17936789]
[0.35004008 0.24578861]
```

9. Predict the class of the test images.

In [10]:
```
test_green = [green_percent[0] for green_percent in test_data]
test_blue = [blue_percent[1] for blue_percent in test_data]
test_red = np.column_stack((test_green,test_blue))
k_p = k1.predict(test_red)
print('Predicted results are ')
print(k_p)
print('\n Actual images are ')
print("['city' 'desert' 'farm']")
```

```
Predicted results are
['city' 'desert' 'desert']

 Actual images are
['city' 'desert' 'farm']
```

10. Print the prediction from the test images and compare with the actual images shown below. Make this comparison clear in the output of your code (e.g. prepend with ‘predicted:’ and ‘actual:’). Try to explain any errors if you note any.

In [28]:
```
print("Results explanation : \n")
print("In this process we gave three images and the image types are 'city'
print("Predicted results are :  ['city' 'desert' 'desert']\n ")
print("'Test2 and Test3' images are having same color. Because of the RGB p
```

```
Results explanation :

In this process we gave three images and the image types are 'city' 'dese
rt' 'farm' and image names are 'Test1,Test2,Test3'
Predicted results are :  ['city' 'desert' 'desert']

'Test2 and Test3' images are having same color. Because of the RGB proper
ties 'Test3' image is classified as desert but it is a farm type
```