In [4]:
```
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
from sklearn import neighbors

#1 Creating a list with the names called image_files
imagefiles = ['farm1.jpg', 'farm2.jpg', 'farm3.jpg', 'farm4.jpg', 'farm5.jpg'
print(r"List of all imagefiles")
for i in imagefiles:
    print(i)
```

```
List of all imagefiles
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
```

In [5]:
```
#2 create an array of strings called training_target
trainingtarget = np.array(['farm', 'farm', 'farm', 'farm', 'farm', 'farm', 'f
print(r"Imagefiles of the list below")
for i in trainingtarget:
    print(i)
```

```
Imagefiles of the list below
farm
farm
farm
farm
farm
farm
farm
farm
city
city
city
city
city
city
city
city
desert
desert
desert
desert
desert
desert
desert
desert
```

In [6]:
```
#Function that returns Percent Green and Percent Blue of an Image

def percentBG(image):

    tupleBG = np.array(image).mean(axis=(0,1))
    R = tupleBG[0]
    G = tupleBG[1]
    B = tupleBG[2]

    Sum = tupleBG[0] + tupleBG[1] + tupleBG[2]

    percentG = tupleBG[1]/Sum
    percentB = tupleBG[2]/Sum

    return percentG, percentB
```

In [8]:
```
#3 Empty List to Append percent green & percent blue of each image of trainin
trainingdata = []

#Using for loop to read name of each image of training data, appending image
for i in imagefiles:

    img = Image.open('images2/' + i)
    trainingdata.append(percentBG(img))
print("Percentage of Green and Blue in each image of the Training Data")
print('************************************************************')
for i in trainingdata:
    print(i)
```

```
Percentage of Green and Blue in each image of the Training Data
************************************************************
(0.38537916213835416, 0.2725025827290944)
(0.38947876516901914, 0.24166749580794727)
(0.37176749098686257, 0.29236929740095713)
(0.3853494059331435, 0.25567274038089727)
(0.3836885427597768, 0.2697444869452292)
(0.3782235141367888, 0.3424372370985558)
(0.3557784135089085, 0.2613897337397366)
(0.36318263603850426, 0.3207925148928169)
(0.3338467930412881, 0.33987007505544775)
(0.3145798947161084, 0.31740954537386984)
(0.32982159222616164, 0.30761097231014695)
(0.3302142216023482, 0.4032948263728943)
(0.3126774452579913, 0.3706804693524618)
(0.36200550003320575, 0.3592237167477091)
(0.3326393074627567, 0.2812241449923416)
(0.33155647847549335, 0.3138749350290284)
(0.2889915365854203, 0.2647862205478914)
(0.3288746497784961, 0.2946128831876114)
(0.32171351112006713, 0.24749944089149414)
(0.3520926067264411, 0.2317126103798501)
(0.32718512631637453, 0.2156491053354232)
(0.33655681001293364, 0.2638719030231327)
(0.3441919206452676, 0.1374953806468185)
(0.32732039192104917, 0.26438328280357887)
```

In [9]:
```
#Green values from training data
Green = [x for x, y in trainingdata]

#Converting the Green values from training data into a 1D array
GreenArray = np.array(Green)
print("       1-D Array consisting of Green Values in Percentage        ")
print('*****************************************************************')
print(GreenArray)
```

```
       1-D Array consisting of Green Values in Percentage
*****************************************************************
[0.38537916 0.38947877 0.37176749 0.38534941 0.38368854 0.37822351
 0.35577841 0.36318264 0.33384679 0.31457989 0.32982159 0.33021422
 0.31267745 0.3620055  0.33263931 0.33155648 0.28899154 0.32887465
 0.32171351 0.35209261 0.32718513 0.33655681 0.34419192 0.32732039]
```

In [10]:
```
#Blue values from training data
Blue = [y for x, y in trainingdata]

#Converting the Blue values from training data into a 1D array
BlueArray = np.array(Blue)
print("       1-D Array consisting of Bule Values in Percentage        ")
print('*****************************************************************')
print(BlueArray)
```

```
       1-D Array consisting of Bule Values in Percentage
*****************************************************************
[0.27250258 0.2416675  0.2923693  0.25567274 0.26974449 0.34243724
 0.26138973 0.32079251 0.33987008 0.31740955 0.30761097 0.40329483
 0.37068047 0.35922372 0.28122414 0.31387494 0.26478622 0.29461288
 0.24749944 0.23171261 0.21564911 0.2638719  0.13749538 0.26438328]
```

In [11]:

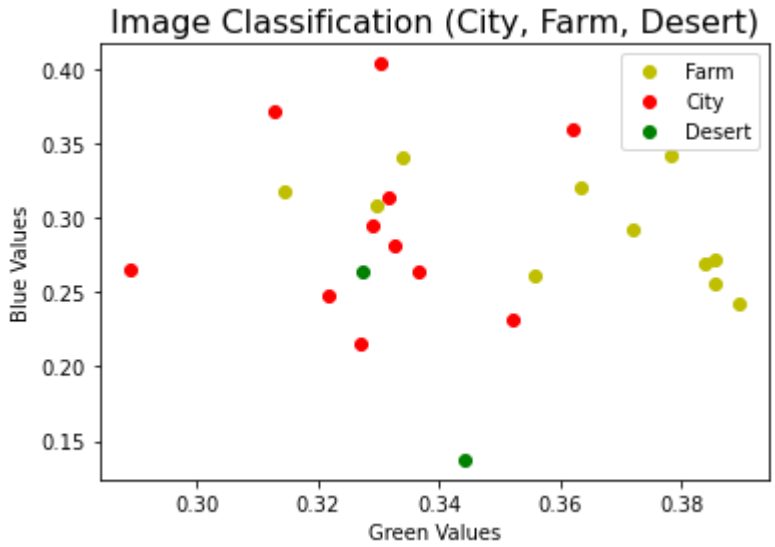
```
#SCATTER PLOT

from matplotlib.pyplot import *
#Plotting the training data
%matplotlib inline

#First set of 11 values in the training set-- Farm values
plot(GreenArray[0:11],BlueArray[0:11],'yo',label='Farm')

#Second set of 11 values in the training set -- City values
plot(GreenArray[11:22],BlueArray[11:22],'ro',label='City')

#Third set of 11 values in the training set-- Desert values
plot(GreenArray[22:44],BlueArray[22:44],'go',label='Desert')
xlabel('Green Values',fontsize=10)
ylabel('Blue Values',fontsize=10)
title('Image Classification (City, Farm, Desert)',fontsize=16)
print()
legend(loc='best')
show()
```



In [12]:
```
#4 perform image processing to get the percent of blue and the percent of gre

trainingArray = np.column_stack((GreenArray,BlueArray))
print()
print("**********Numpy Array with the training values of Green and Blue of th
print(trainingArray)
```

```
**********Numpy Array with the training values of Green and Blue of the ima
ges****************
[[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]]
```

In [13]:
```
#5 Create your classifier
k = neighbors.KNeighborsClassifier(1,weights='distance')

#6 Training your classifier by fit command
k.fit(trainingArray, trainingtarget)
```

Out[13]:
```
KNeighborsClassifier(n_neighbors=1, weights='distance')
```

In [14]:
```
#7 Now create an empty test_data array and fill it with the proper values for

TestImage = ['test1.jpg', 'test2.jpg', 'test3.jpg']
TestSet = []
for i in TestImage:
    imagepath = ('images2/' + i)
    img = Image.open(imagepath)
    TestSet.append(percentBG(img)) #using the function defined before
print("**********The Percentage of Green and Blue for each image of the Tes
for i in TestSet:
    print(i)

#PercentGreen values from Test set
TestPercentGreen = [x for x, y in TestSet]
TestPercentGreenArray = np.array(TestPercentGreen)

#PercentBlue values from Test set
TestPercentBlue = [y for x, y in TestSet]
TestPercentBlueArray = np.array(TestPercentBlue)

#Creates array with PercentGreen and PercentBlue TestSet values
TestArray = np.column_stack((TestPercentGreenArray,TestPercentBlueArray))
print()
print("**********Numpy Array with the TestSet values for Percentage Green a
print(TestArray)
```

```
**********The Percentage of Green and Blue for each image of the Test Dat
a**********
(0.32695920083037133, 0.3268851262195992)
(0.3342938446981946, 0.17936788871306228)
(0.35004008017770316, 0.24578861396084875)

**********Numpy Array with the TestSet values for Percentage Green and Bl
ue of the Test images**********
[[0.3269592  0.32688513]
 [0.33429384 0.17936789]
 [0.35004008 0.24578861]]
```

In [132]:
```
#8 Predict the class of the test images
k_pred = k.predict(TestArray)

#9 Print the prediction from the test images and compare with the actual imag
print("     Predicted Values:    ")
print(k_pred)
print("**********************************")
print("     Actual Values:        ")
print("['city', 'desert', 'farm']")
```

```
     Predicted Values:
['city' 'desert' 'desert']
**********************************
     Actual Values:
['city', 'desert', 'farm']
```

In [ ]:
```
The predicted and actual values for the first two images, City and Desert are
But we got wrong prediction for the image test3.jpg.
Because the image has dry grass in brown color.
```