# Homework 4

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import numpy.matlib
from sklearn.cluster import KMeans
from sklearn import mixture
from sklearn.svm import SVC
from mpl_toolkits.mplot3d import Axes3D
```

## Functions

```
def normalize(image, stack = False):
  im = np.zeros(image.shape)
  for plane in range(3):
    im[:,:,plane] = (image[:,:,plane] - np.min(image[:,:,plane]))/(np.max(image[:,:,plane]) - np.min(image[:,:,pla

  if stack:
    x = np.linspace(0,image.shape[0] - 1,image.shape[0])
    x = np.expand_dims(x, axis = 1)
    x = np.matlib.repmat(x, 1, image.shape[1])

    y = np.linspace(0,image.shape[1] - 1,image.shape[1])
    y = np.expand_dims(y, axis = 1).T
    y = np.matlib.repmat(y, image.shape[0], 1)

    x = (x - np.min(x))/(np.max(x) - np.min(x))
    y = (y - np.min(y))/(np.max(y) - np.min(y))

    # print(x.shape, y.shape, im.shape)
    im = np.dstack((x,y,im))

  return im
```

```
def kmeans(vector, k):
  labels = np.zeros(vector.shape[0])
  mu_guess_idx = np.random.randint(0, vector.shape[0], k)
  mu_guess = vector[mu_guess_idx]
  print(vector.shape)
  count = 0
  while (count < 10):
    for i in range(vector.shape[0]):
      labels[i] = np.argmin(np.array([np.linalg.norm(vector[i,:] - mu_guess[ki,:]) for ki in range(k)]))

    # labels = np.argmin(np.array([np.linalg.norm(vector - mu_guess[i,:]) for i in range(k)]))
    # print(vector - mu_guess[1,:])

    for j in range(k):
      mu_guess[j,:] = np.mean(vector[np.where(labels == j)[0]], axis = 0)

    count += 1
    print(count)
  return labels
```

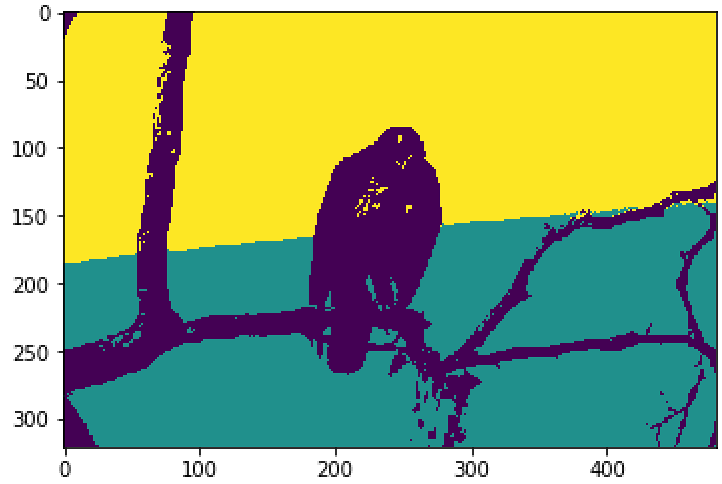
```
test = kmeans(norm_bird.reshape(321*481,-1), 3)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-4-25cdb8143727> in <module>()
----> 1 test = kmeans(norm_bird.reshape(321*481,-1), 3)

NameError: name 'norm_bird' is not defined
```

SEARCH STACK OVERFLOW

```
plt.imshow(test.reshape(321,481))
```

```
<matplotlib.image.AxesImage at 0x7f7fe15930b8>
```



## Answer 1

## K-Means

```
bird = plt.imread('bird.jpg')
norm_bird = normalize(bird, True)
```

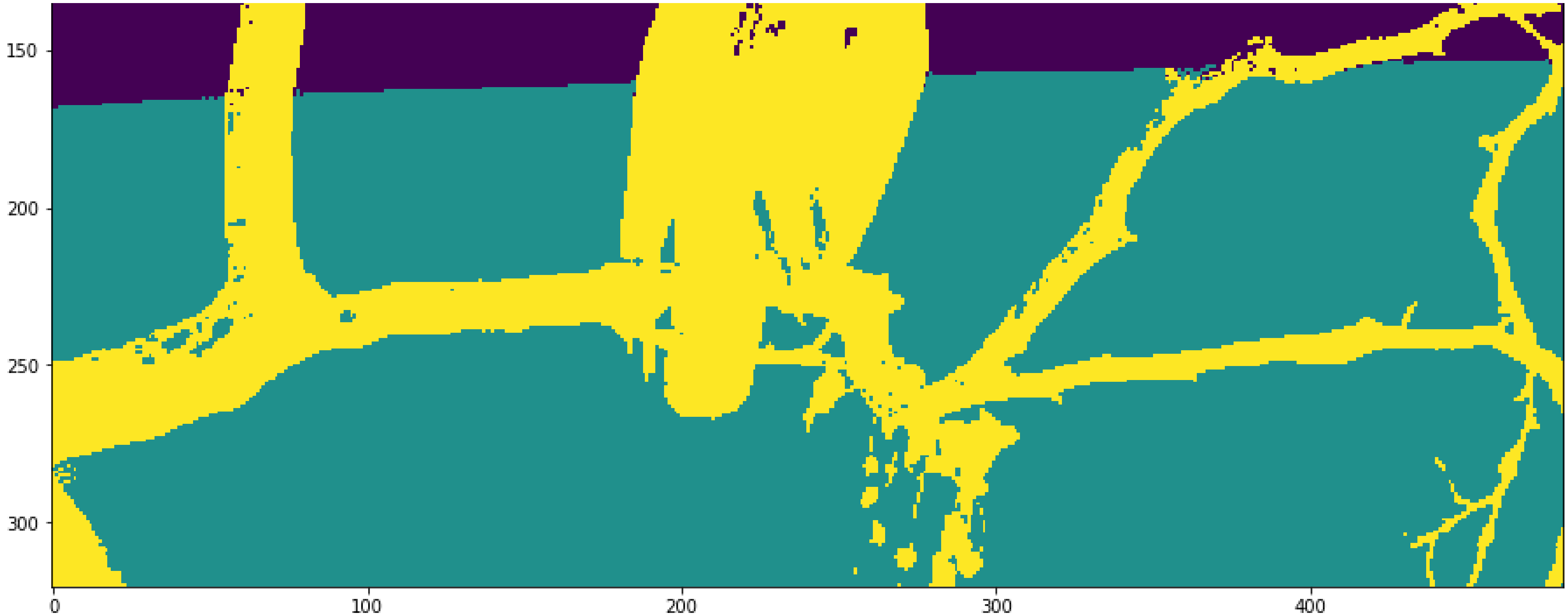
```
k = [2, 3, 4, 5]
plt.axes([1,1,2,2])
plt.imshow(bird)
for i in k:
  cluster = KMeans(i)
  cluster.fit(norm_bird.reshape(bird.shape[0]*bird.shape[1],-1))
  labs = cluster.labels_
  plt.figure()
  plt.axes([1,1,2,2])
  plt.imshow(labs.reshape(bird.shape[0], bird.shape[1]))
```


0
50
100
150
200
250
300
0
100
200
300
400

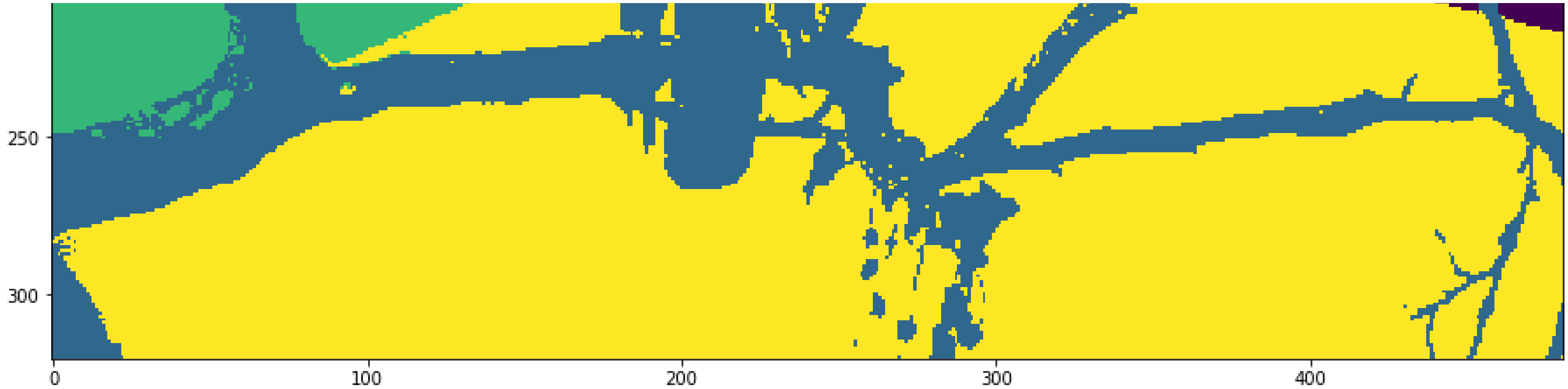

0
50


100
150
200
250
300
0
100
200
300
400


0
50
100


150
200
250
300
0
100
200
300
400


0
50
100
150
200


250
300
0
100
200
300
400


0
50
100
150
200
250

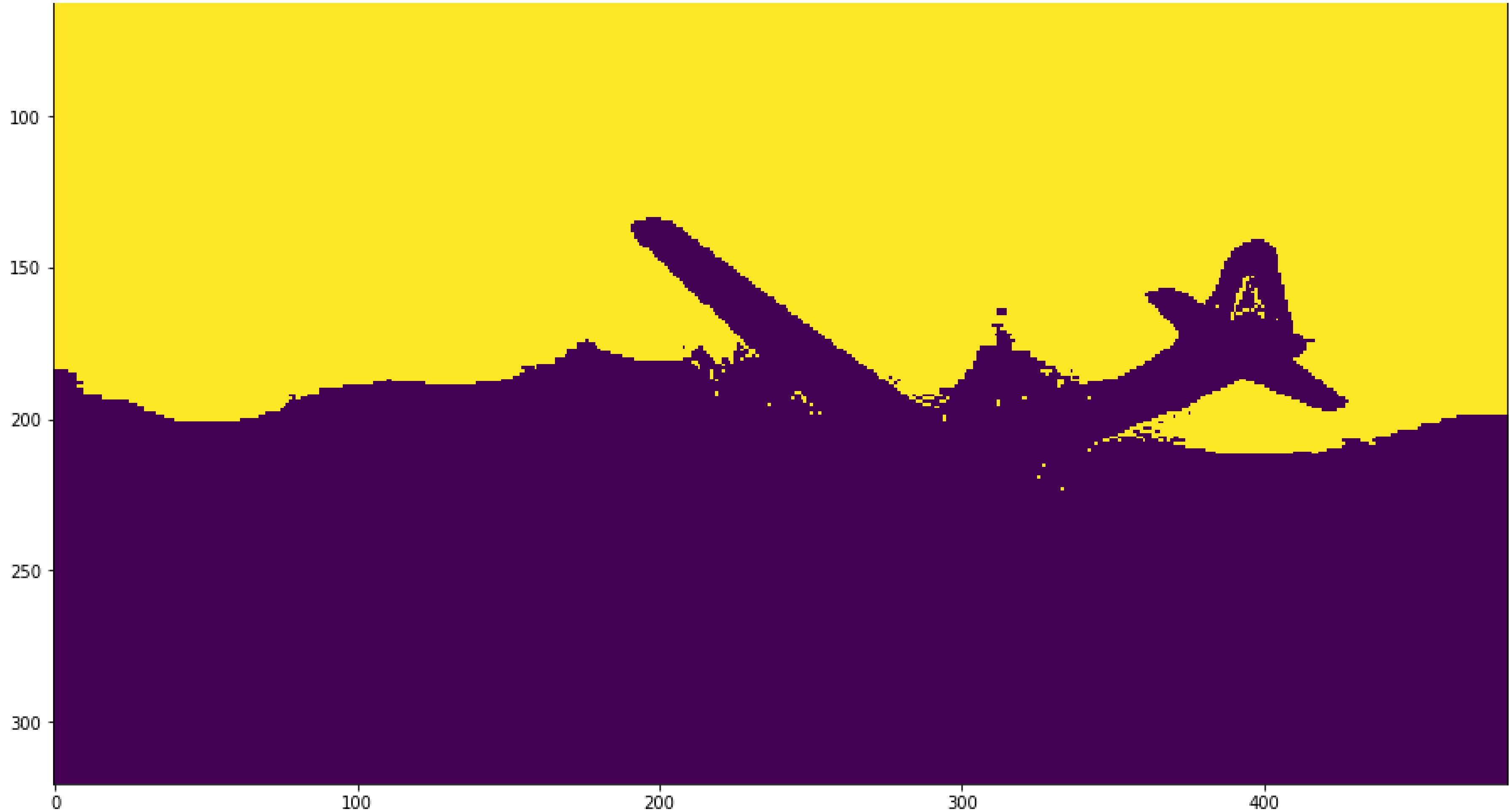

Double-click (or enter) to edit

```
plane = plt.imread('plane.jpg')
norm_plane = normalize(plane, True)
```

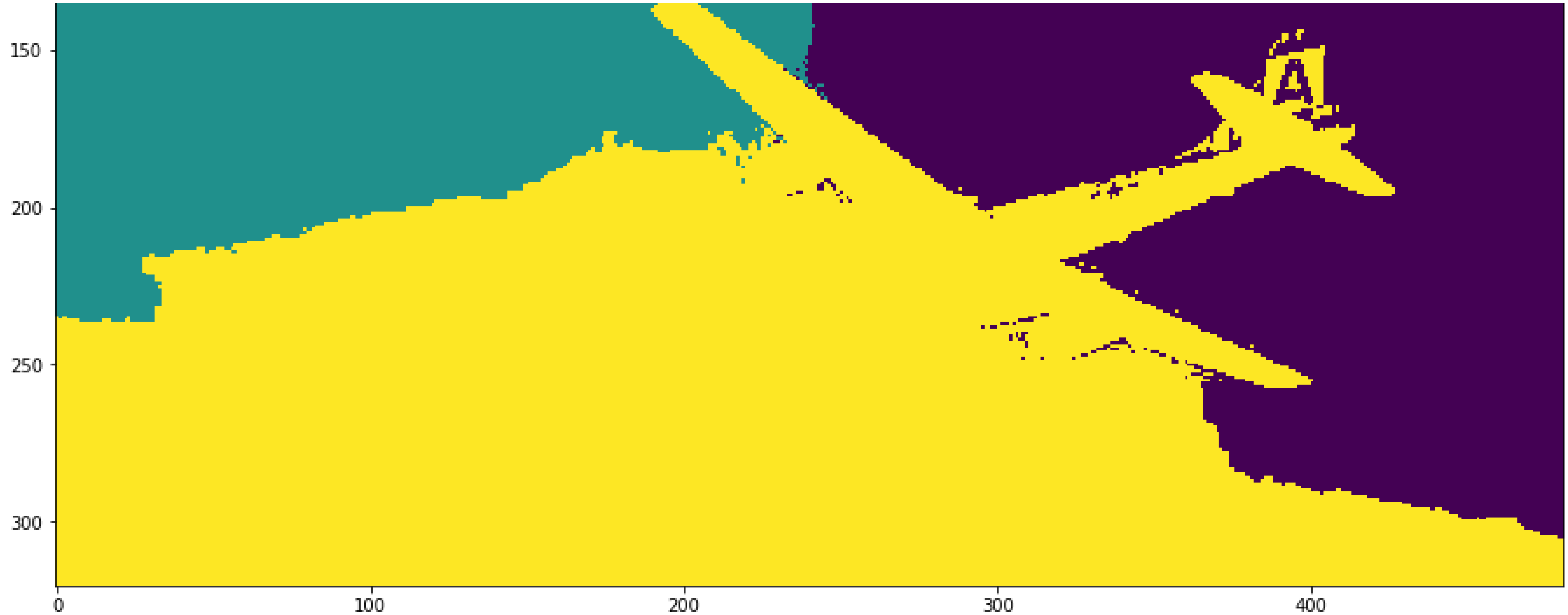
```
k = [2, 3, 4, 5]
plt.axes([1,1,2,2])
plt.imshow(plane)
for i in k:
  cluster = KMeans(i)
  cluster.fit(norm_plane.reshape(plane.shape[0]*plane.shape[1],-1))
  labs = cluster.labels_
  plt.figure()
  plt.axes([1,1,2,2])
  plt.imshow(labs.reshape(plane.shape[0], plane.shape[1]))
```


0
50
100
150
200
250
300
0
100
200
300
400

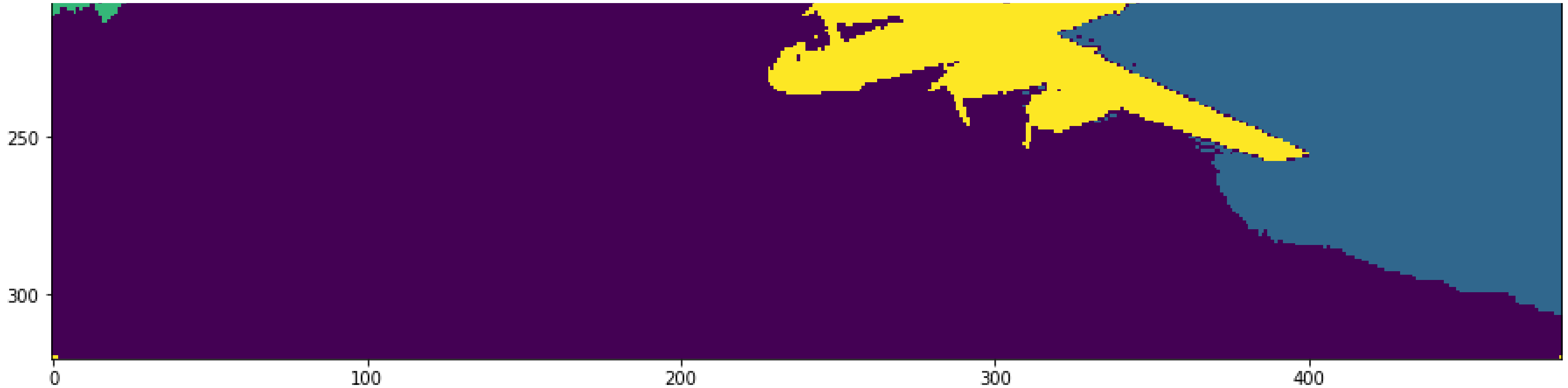

0
50


100
150
200
250
300
0
100
200
300
400


0
50
100


150
200
250
300
0
100
200
300
400


0
50
100
150
200


250
300
0
100
200
300
400

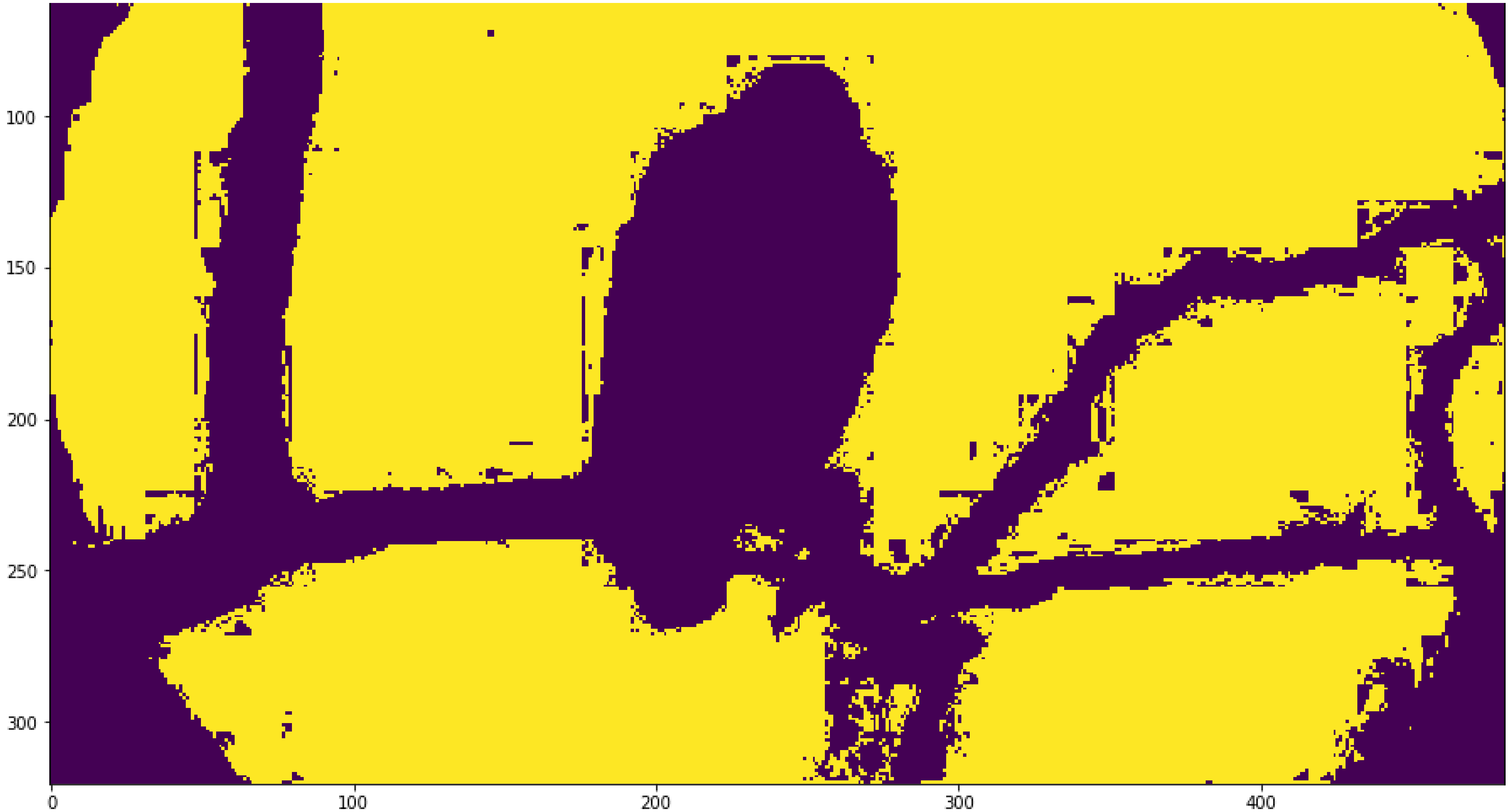

0
50
100
150
200
250


## GMM

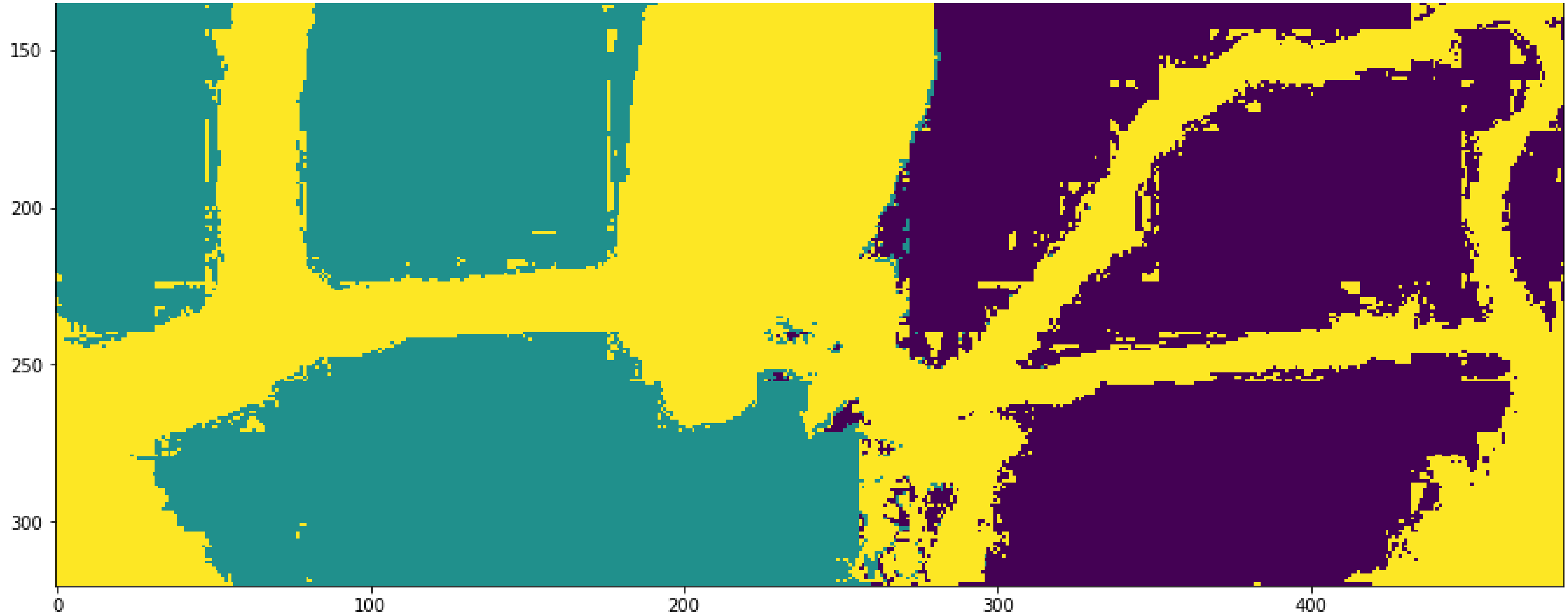
```
plt.axes([1,1,2,2])
plt.imshow(bird)
for i in k:
  gmm = mixture.GaussianMixture(i)
  gmm.fit(norm_bird.reshape(bird.shape[0]*bird.shape[1],-1))
  labs1 = gmm.predict(norm_bird.reshape(bird.shape[0]*bird.shape[1],-1))
  plt.figure()
  plt.axes([1,1,2,2])
  plt.imshow(labs1.reshape(bird.shape[0], bird.shape[1]))
```


0
50
100
150
200
250
300
0
100
200
300
400

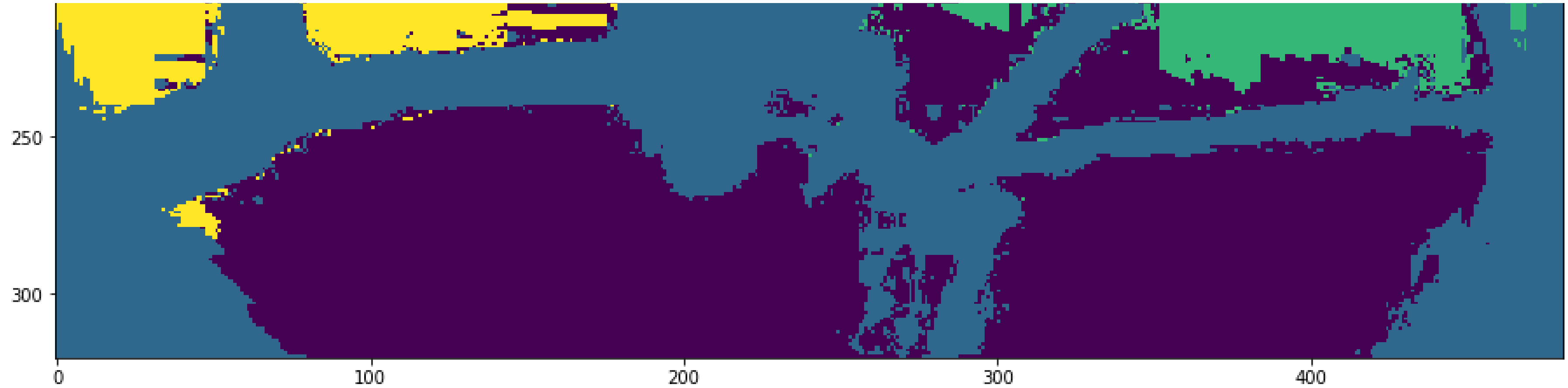

0
50


100
150
200
250
300
0
100
200
300
400


0
50
100

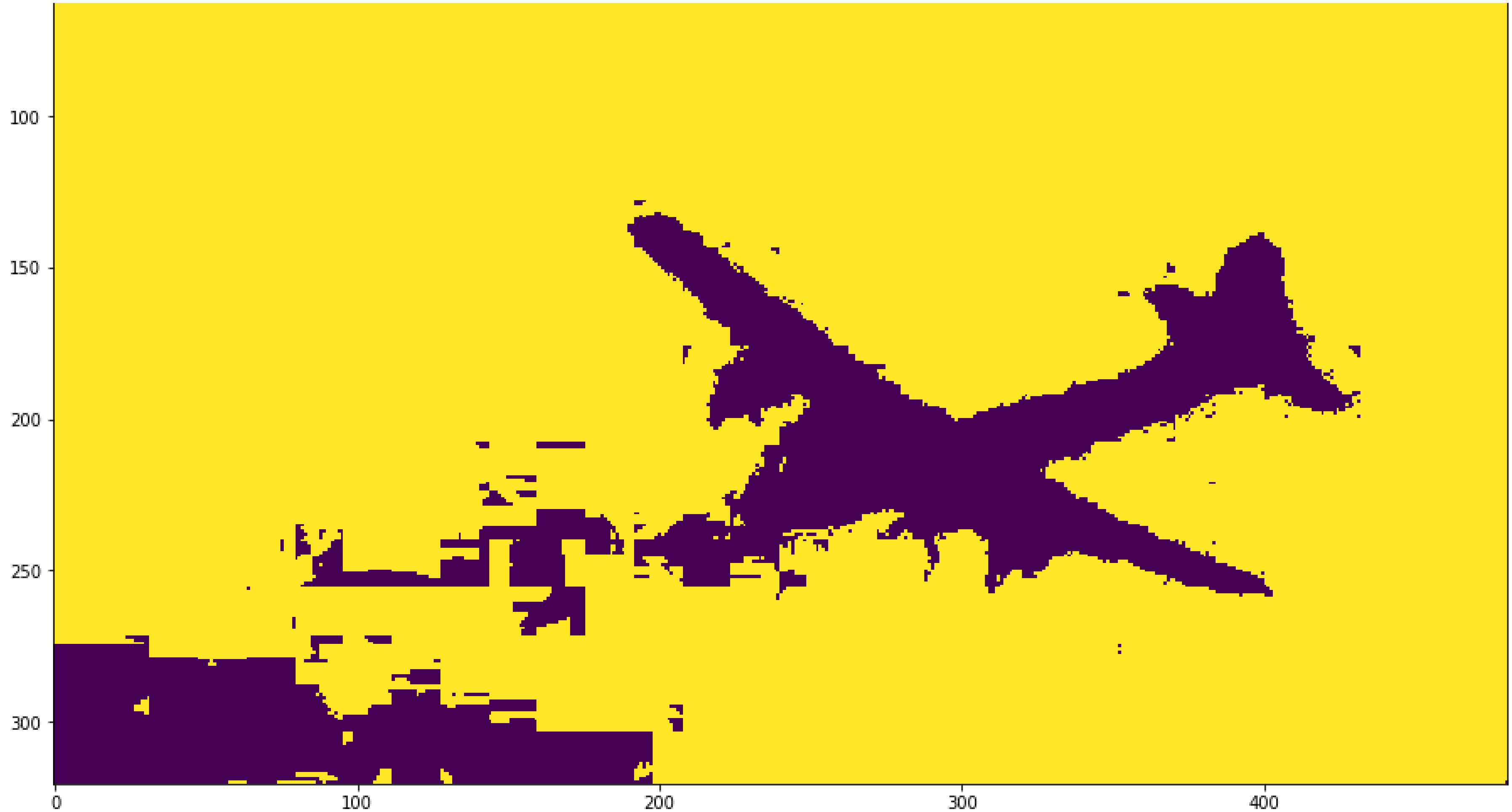

150
200
250
300
0
100
200
300
400


0
50
100
150
200

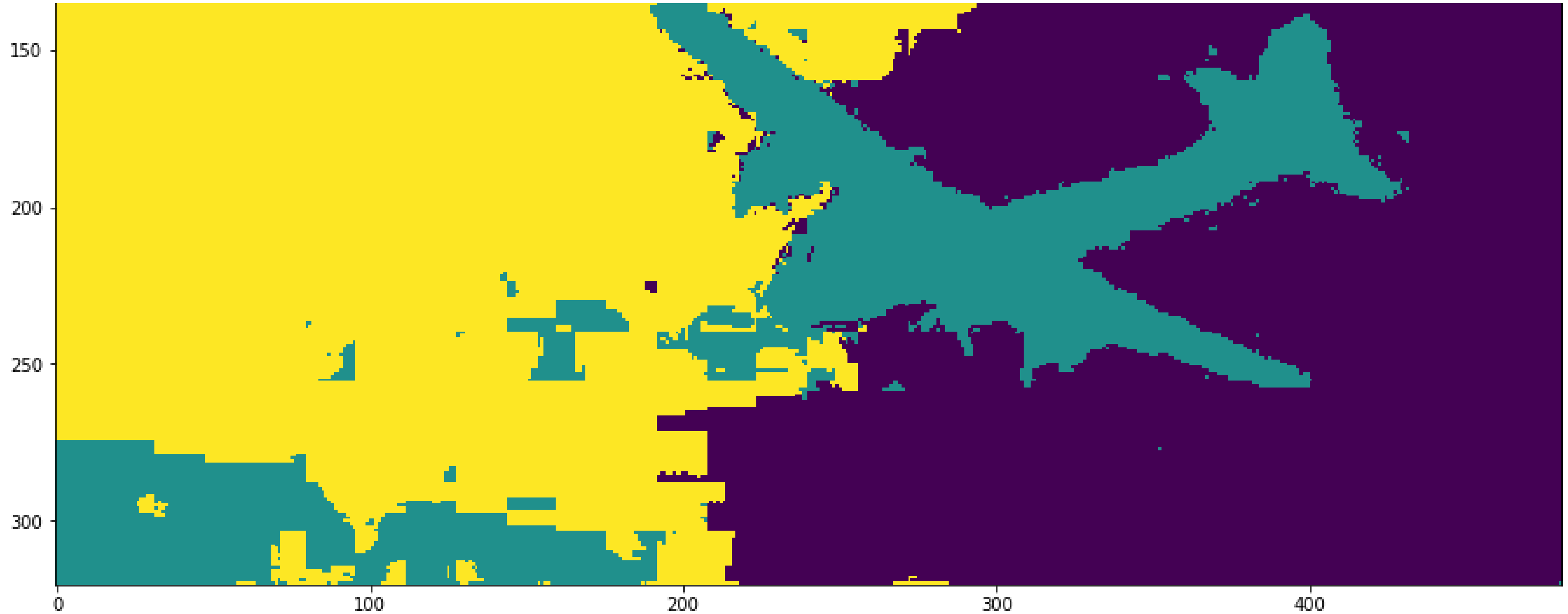

250
300
0
100
200
300
400

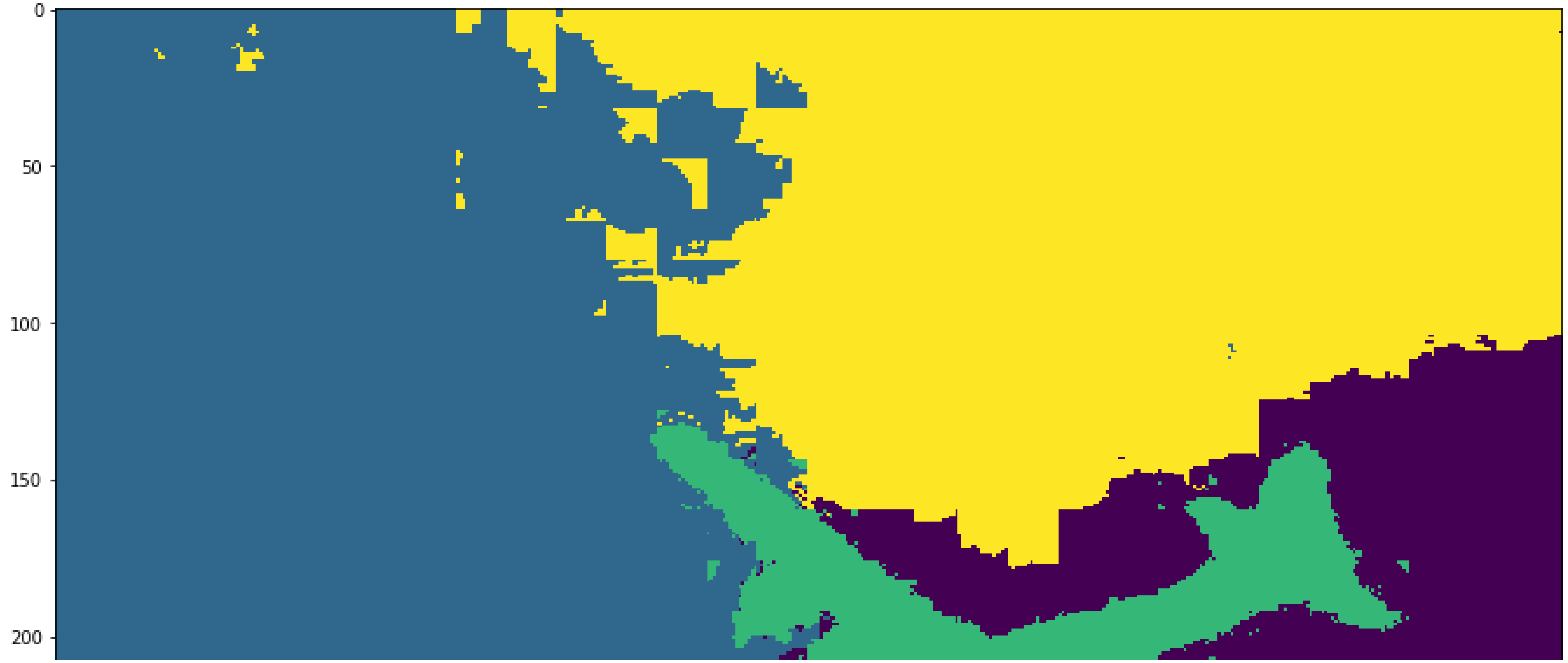

0
50
100
150
200
250

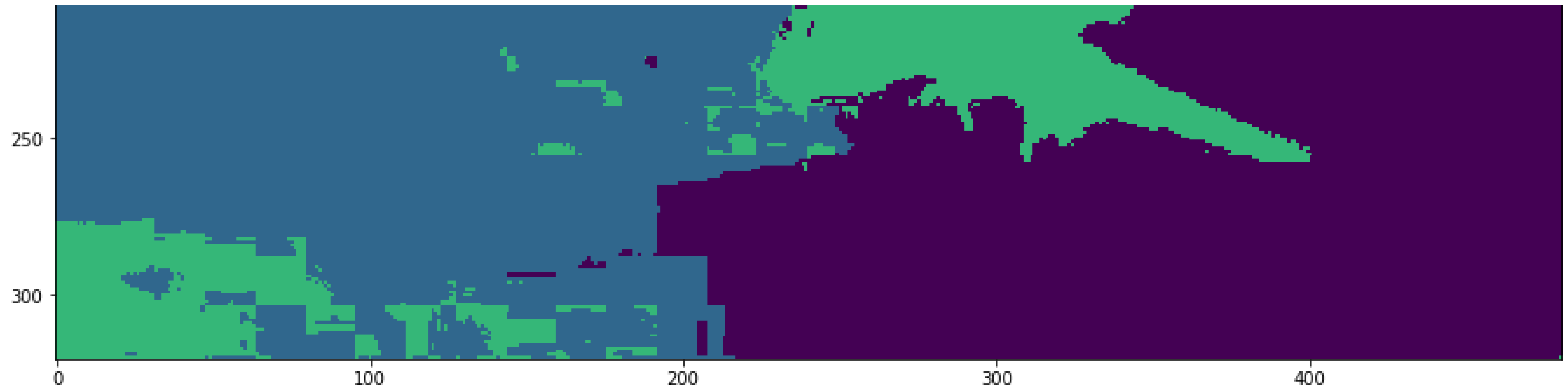

```
plt.axes([1,1,2,2])
plt.imshow(plane)
for i in k:
  gmm = mixture.GaussianMixture(i)
  gmm.fit(norm_plane.reshape(plane.shape[0]*plane.shape[1],-1))
  labs1 = gmm.predict(norm_plane.reshape(plane.shape[0]*plane.shape[1],-1))
  plt.figure()
  plt.axes([1,1,2,2])
  plt.imshow(labs1.reshape(plane.shape[0], plane.shape[1]))
```


100
150
200
250
300
0
100
200
300
400


0
50
100


150
200
250
300
0
100
200
300
400


0
50
100
150
200

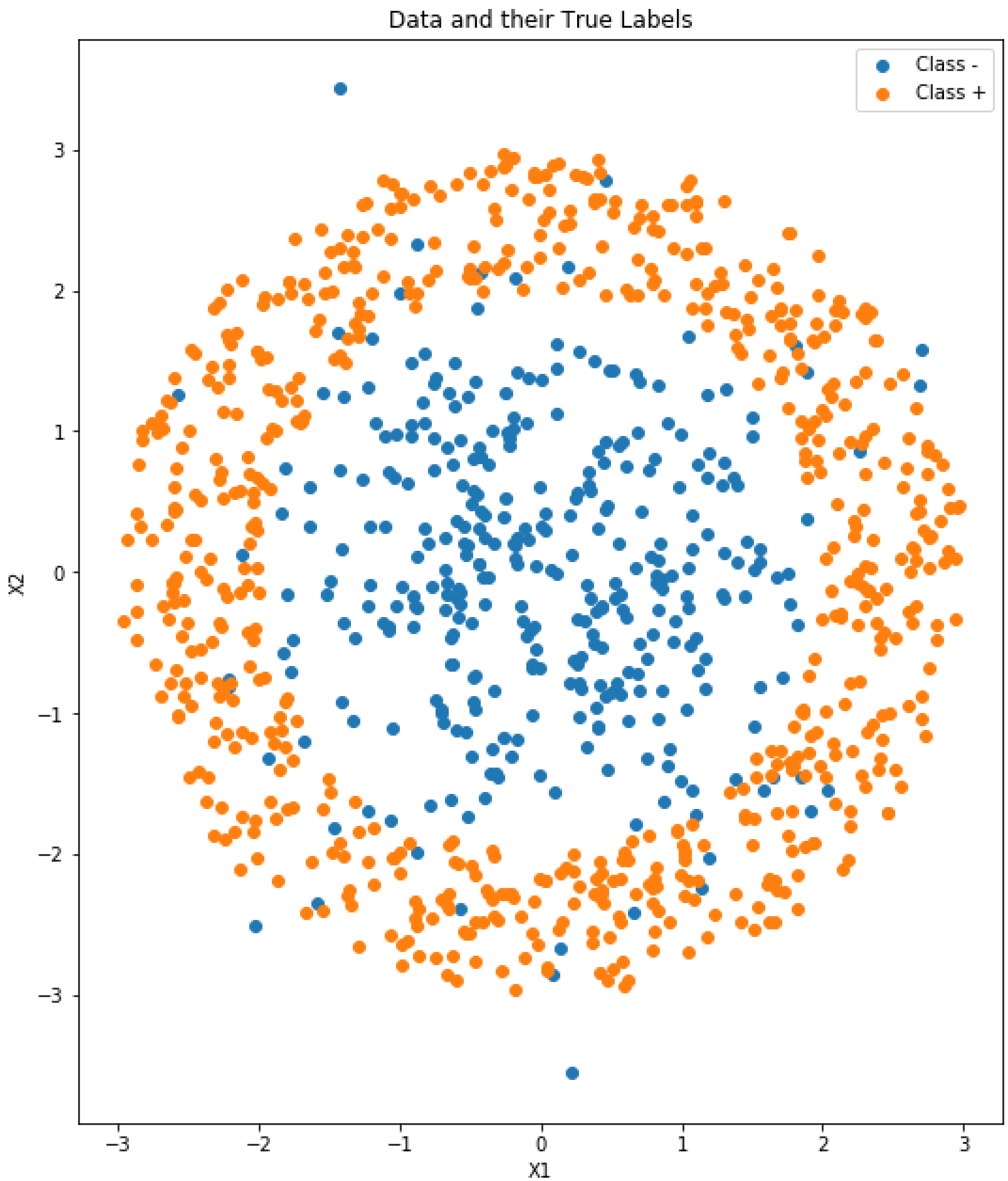

250
300
0
100
200
300
400


0
50
100
150
200
250


## Answer 2

```
N = 1000
mean = np.array([0, 0])
sigma = np.array([1, 0, 0, 1]).reshape(2,2)
prior = [0.35, 0.65]

data = np.zeros([2, 1000])

temp = np.random.rand(1,1000)

idx1 = np.where(temp <= prior[0])[1]
idx2 = np.where(temp > prior[0])[1]

data[:, idx1] = np.random.multivariate_normal(mean, sigma, idx1.shape[0]).T

radius = np.random.uniform(2,3,idx2.shape[0])
angle = np.random.uniform(-np.pi, np.pi, idx2.shape[0])

data[:,idx2] = np.array([radius*np.cos(angle), radius*np.sin(angle)])
Y = np.zeros([1, N])
Y[:, idx1] = np.zeros(idx1.shape[0])
Y[:, idx2] = np.ones(idx2.shape[0])



plt.axes([1,1,2,2])
plt.scatter(data[0,idx1], data[1,idx1])
```

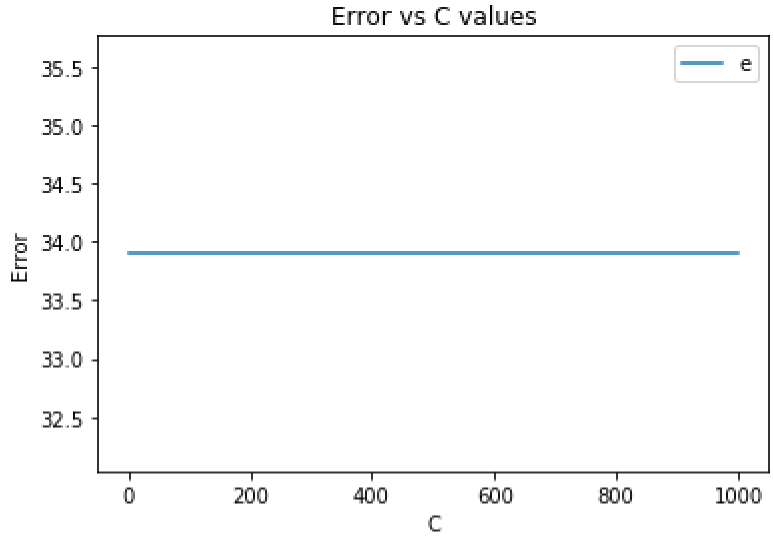
```
plt.scatter(data[0,idx2], data[1,idx2])
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Data and their True Labels')
plt.legend(('Class -', 'Class +'))
plt.gca().set_aspect('equal', 'box')
```



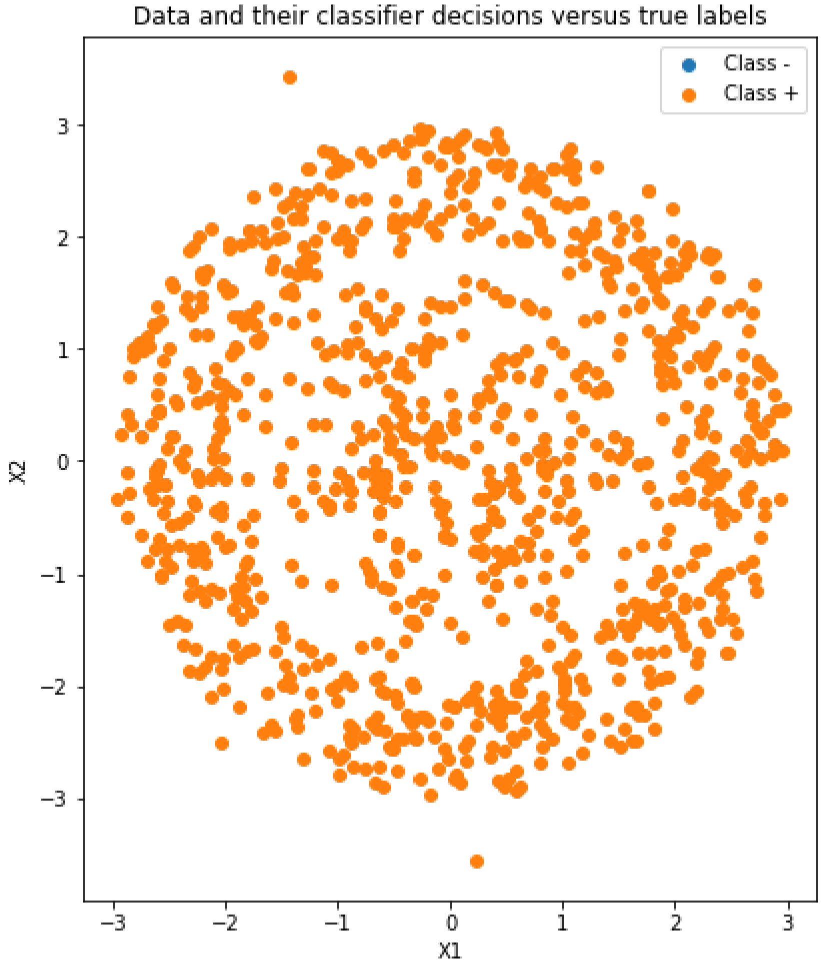

## Linear SVM

```
k = 10
N = 1000

C = [0.001, 0.01, 0.1, 1, 10, 100, 1000]

best = []
for c in C:
  svm_lin = SVC(c, kernel='linear')
  err = []
  for fold in range(k):
    val_data = data[:,fold*(N//k):(fold+1)*(N//k)] # split data into val and train
    y_val = Y[:,fold*(N//k):(fold+1)*(N//k)]
    train_data = np.concatenate((data[:,:fold*(N//k)], data[:,(fold+1)*(N//k):]), axis = 1)
    y_train = np.concatenate((Y[:,:fold*(N//k)], Y[:,(fold+1)*(N//k):]), axis = 1)
    svm_lin.fit(train_data.T, y_train.reshape(y_train.shape[1],))
    label = svm_lin.predict(val_data.T)
    err.append(np.sum(np.abs(label - y_val)))
  best.append(np.mean(err))


plt.plot(C,best)
plt.title('Error vs C values')
plt.xlabel('C')
plt.ylabel('Error')
plt.legend('error')
```

```
<matplotlib.legend.Legend at 0x7f5e30a5d2b0>
```




```
best_c = best.index(min(best))

svm_lin = SVC(C[best_c], kernel='linear')
svm_lin.fit(data.T, Y.reshape(Y.shape[1],))
labs1 = svm_lin.predict(data.T)
boundary1 = svm_lin.decision_function(data.T)

plt.axes([1,1,1.5,1.5])
plt.scatter(data[0,np.where(labs1 == 0)[0]], data[1,np.where(labs1 == 0)[0]])
plt.scatter(data[0,np.where(labs1 == 1)[0]], data[1,np.where(labs1 == 1)[0]])
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Data and their classifier decisions versus true labels')
plt.legend(('Class -', 'Class +'))
plt.gca().set_aspect('equal', 'box')
```




## Gaussian SVM

```
k = 10
N = 1000

C = [0.1, 1, 10, 100]
G = [0.1, 1, 10, 100]

best = []
```

```
for c in C:
  for g in G:
    svm_gaus = SVC(c, kernel='rbf', gamma = g)
    err = []
    for fold in range(k):
      val_data = data[:,fold*(N//k):(fold+1)*(N//k)] # split data into val and train
      y_val = Y[:,fold*(N//k):(fold+1)*(N//k)]
      train_data = np.concatenate((data[:,:fold*(N//k)], data[:,(fold+1)*(N//k):]), axis = 1)
      y_train = np.concatenate((Y[:,:fold*(N//k)], Y[:,(fold+1)*(N//k):]), axis = 1)
      svm_gaus.fit(train_data.T, y_train.reshape(y_train.shape[1],))
      label = svm_gaus.predict(val_data.T)
      err.append(np.sum(np.abs(label - y_val)))
    best.append(np.mean(err))
```

```
best_g = G[best.index(min(best))//4]
best_g
```

```
100
```

```
best_c = C[best.index(min(best))//10]
best_c
```

```
1
```

```
best
```

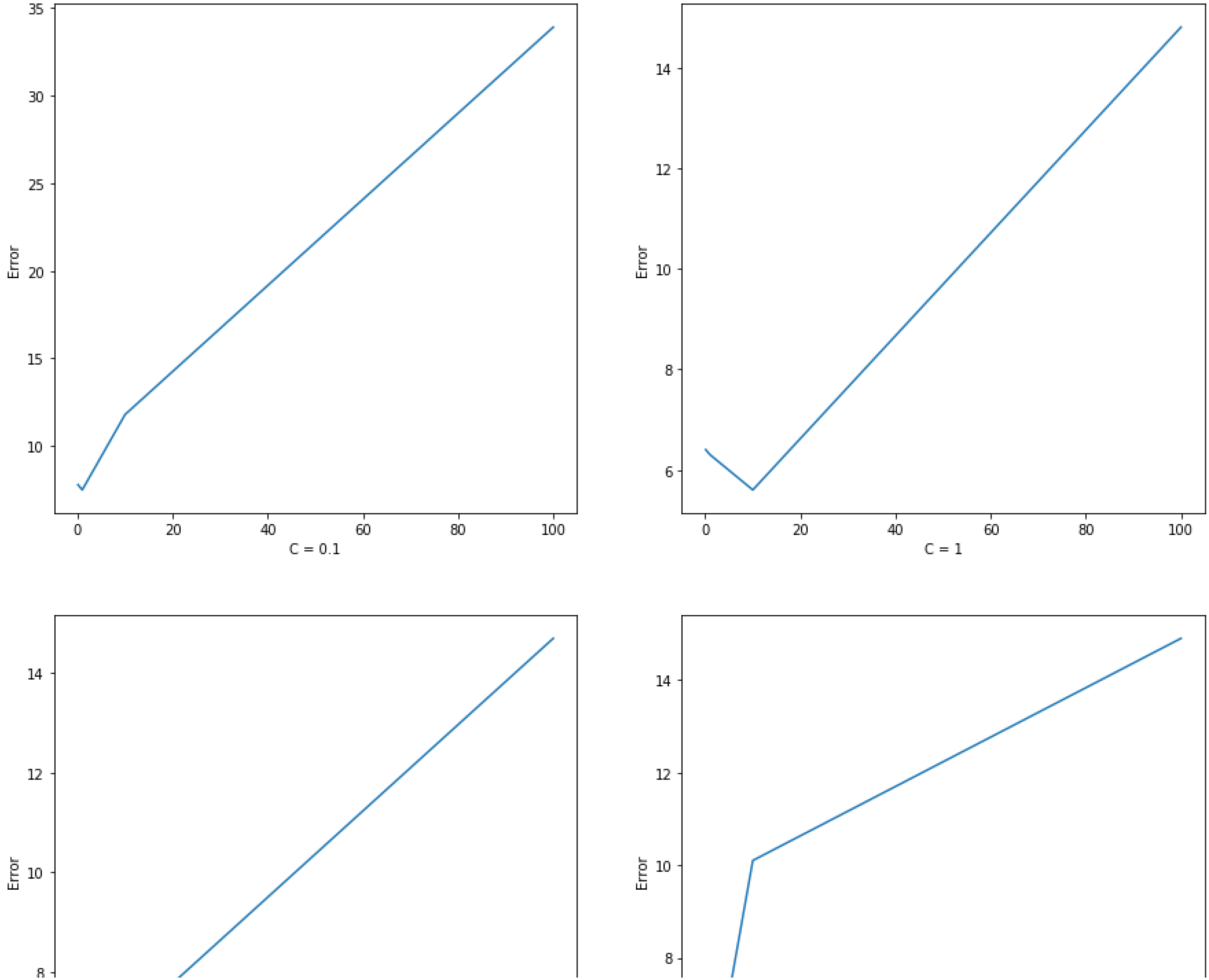
```
[7.8,
 7.5,
 11.8,
 33.9,
 6.4,
 6.3,
 5.6,
 14.8,
 6.2,
 5.4,
 6.9,
 14.7,
 5.8,
 4.9,
 10.1,
 14.9]
```

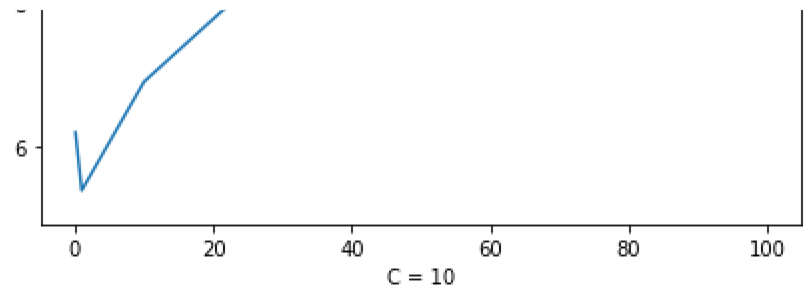
```
np.array(best).reshape(4,4)
```

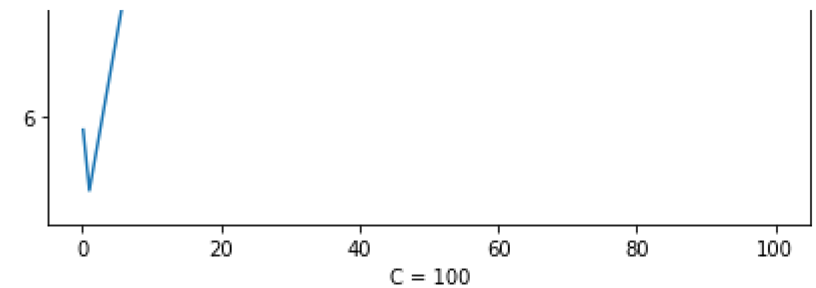
```
array([[ 7.8,  7.5, 11.8, 33.9],
       [ 6.4,  6.3,  5.6, 14.8],
       [ 6.2,  5.4,  6.9, 14.7],
       [ 5.8,  4.9, 10.1, 14.9]])
```

```
fig, ax = plt.subplots(2,2, figsize=(15,15))
ax[0,0].plot([0.1,1,10,100], best[:4])
# ax[0,0].plot(best.index(min(best[:4])), min(best[:4]), 'o')
# ax[0,0].text(i, v+25, "%d" %v, ha="center")
ax[0,1].plot([0.1,1,10,100], best[4:8])
ax[1,0].plot([0.1,1,10,100], best[8:12])
ax[1,1].plot([0.1,1,10,100], best[12:16])
ax[0,0].set(xlabel='C = 0.1', ylabel='Error')
ax[0,1].set(xlabel='C = 1', ylabel='Error')
ax[1,0].set(xlabel='C = 10', ylabel='Error')
ax[1,1].set(xlabel='C = 100', ylabel='Error')
```

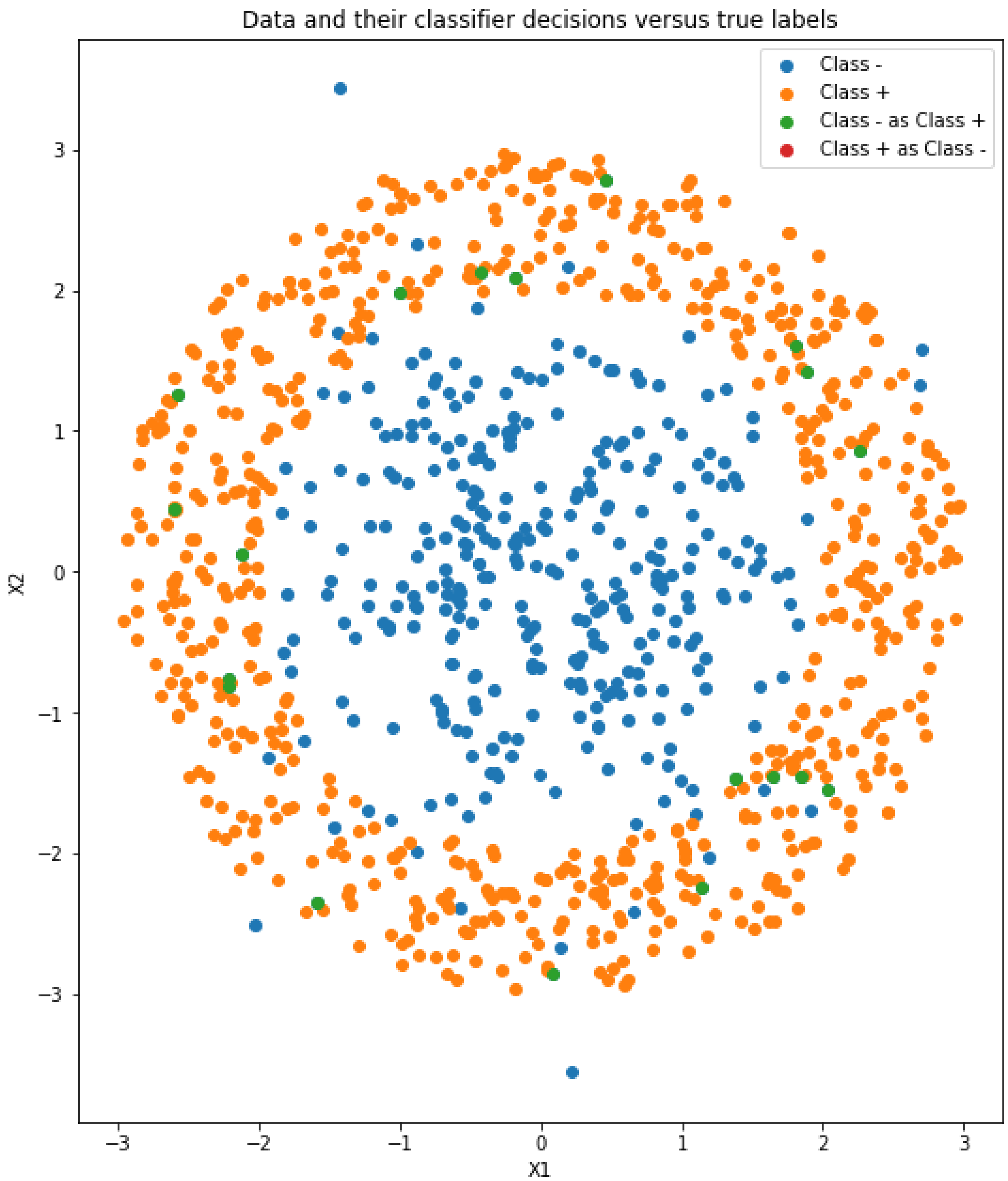
```
[Text(0, 0.5, 'Error'), Text(0.5, 0, 'C = 100')]
```






```
svm_gaus = SVC(best_c, kernel='rbf', gamma = best_g)
svm_gaus.fit(data.T, Y.reshape(Y.shape[1],))
labs2 = svm_gaus.predict(data.T)
boundary2 = svm_gaus.decision_function(data.T)


a = np.where((labs2 == 1) & (Y == 0))[1]
b = np.where((labs2 == 0) & (Y == 1))[1]
plt.axes([1,1,2,2])
plt.scatter(data[0,idx1], data[1,idx1])
plt.scatter(data[0,idx2], data[1,idx2])
plt.scatter(data[0, a], data[1, a])
plt.scatter(data[0, b], data[1, b])
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Data and their classifier decisions versus true labels')
plt.legend(('Class -', 'Class +', 'Class - as Class +', 'Class + as Class -'))
plt.gca().set_aspect('equal', 'box')
```




```
print(idx1.shape, idx2.shape, a.shape, b.shape)
```

```
(339,) (661,) (19,) (0,)
```

## Test Data Set

```
data_tst = np.zeros([2, 1000])

temp_tst = np.random.rand(1,1000)

idx1_tst = np.where(temp_tst <= prior[0])[1]
idx2_tst = np.where(temp_tst > prior[0])[1]

data_tst[:, idx1_tst] = np.random.multivariate_normal(mean, sigma, idx1_tst.shape[0]).T

radius_tst = np.random.uniform(2,3,idx2_tst.shape[0])
angle_tst = np.random.uniform(-np.pi, np.pi, idx2_tst.shape[0])

data_tst[:,idx2_tst] = np.array([radius_tst*np.cos(angle_tst), radius_tst*np.sin(angle_tst)])
Y_tst = np.zeros([1, N])
Y_tst[:, idx1_tst] = np.zeros(idx1_tst.shape[0])
Y_tst[:, idx2_tst] = np.ones(idx2_tst.shape[0])
```

```
print(idx1_tst.shape, idx2_tst.shape)
```

```
(347,) (653,)
```

```
labels_tst_lin = svm_lin.predict(data_tst.T)
```

```
labels_tst_gaus = svm_gaus.predict(data_tst.T)
```

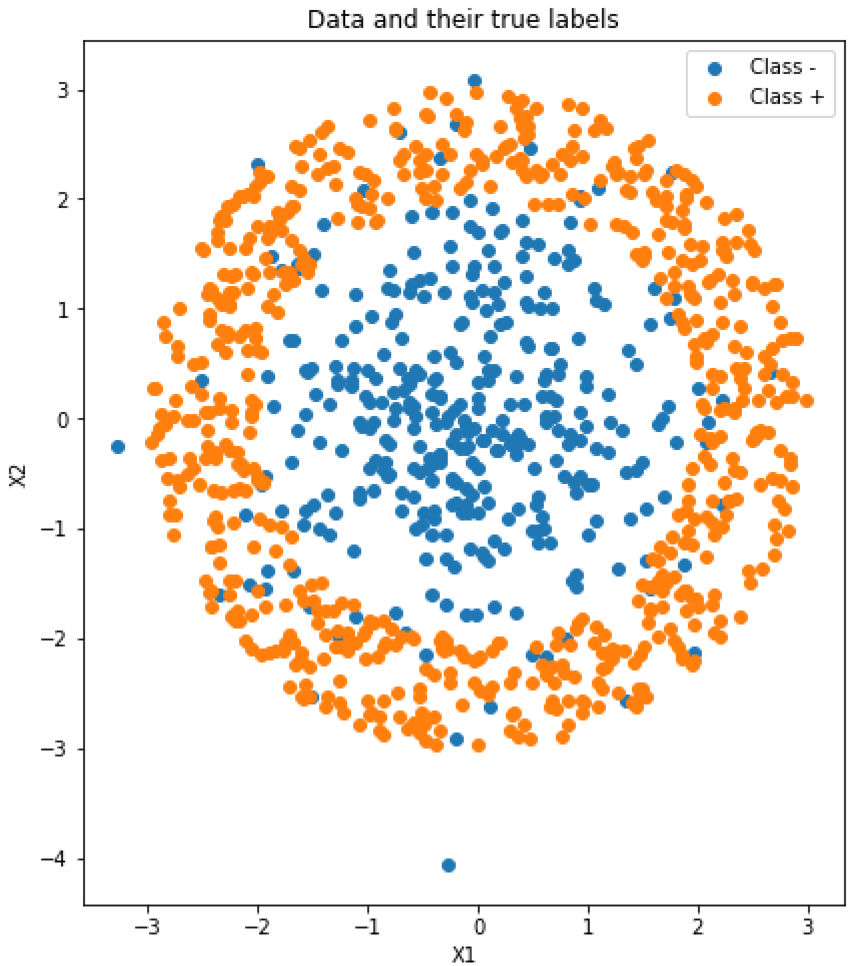
```
np.sum(abs(labels_tst_lin - Y_tst))
```

```
347.0
```

```
np.sum(abs(labels_tst_gaus - Y_tst))
```

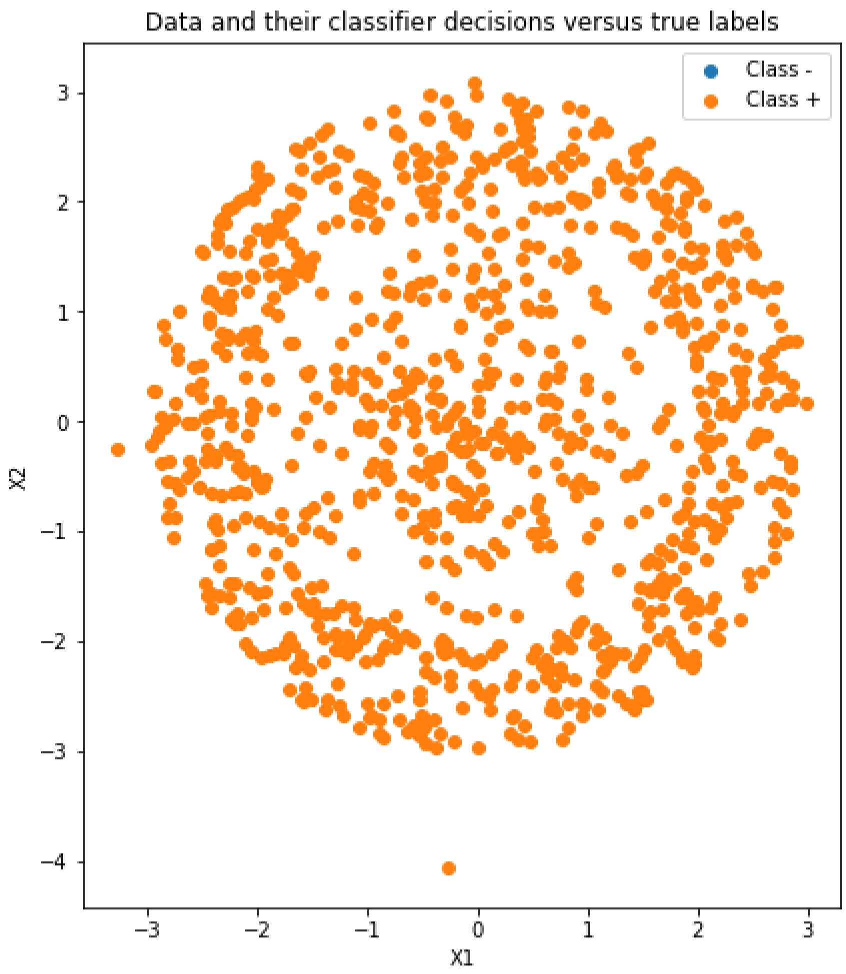
```
150.0
```

```
plt.axes([1,1,1.5,1.5])
plt.scatter(data_tst[0,idx1_tst], data_tst[1,idx1_tst])
plt.scatter(data_tst[0,idx2_tst], data_tst[1,idx2_tst])
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Data and their true labels')
plt.legend(('Class -', 'Class +'))
plt.gca().set_aspect('equal', 'box')
```



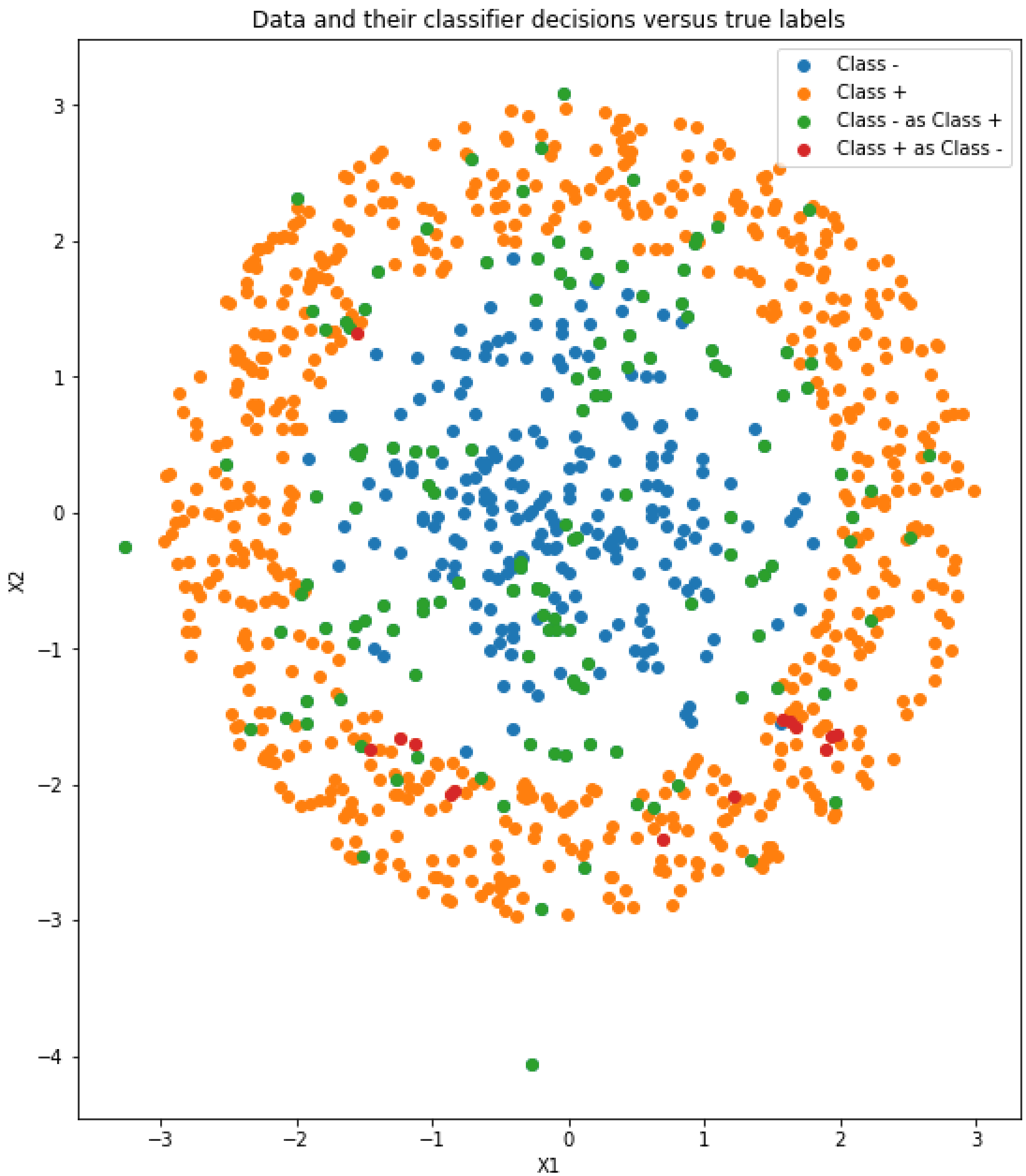

```
plt.axes([1,1,1.5,1.5])
plt.scatter(data_tst[0,np.where(labels_tst_lin == 0)[0]], data_tst[1,np.where(labels_tst_lin == 0)[0]])
```

```
plt.scatter(data_tst[0,np.where(labels_tst_lin == 1)[0]], data_tst[1,np.where(labels_tst_lin == 1)[0]])
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Data and their classifier decisions versus true labels')
plt.legend(('Class -', 'Class +'))
plt.gca().set_aspect('equal', 'box')
```




```
a1 = np.where((labels_tst_gaus == 1) & (Y_tst == 0))[1]
b1 = np.where((labels_tst_gaus == 0) & (Y_tst == 1))[1]
plt.axes([1,1,2,2])
plt.scatter(data_tst[0,idx1_tst], data_tst[1,idx1_tst])
```

```
plt.scatter(data_tst[0,idx2_tst], data_tst[1,idx2_tst])
plt.scatter(data_tst[0, a1], data_tst[1, a1])
plt.scatter(data_tst[0, b1], data_tst[1, b1])
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Data and their classifier decisions versus true labels')
plt.legend(('Class -', 'Class +', 'Class - as Class +', 'Class + as Class -'))
plt.gca().set_aspect('equal', 'box')
```




```
print(idx1_tst.shape[0]- a1.shape[0],idx2_tst.shape[0] -b1.shape[0])
```

```
211 639
```