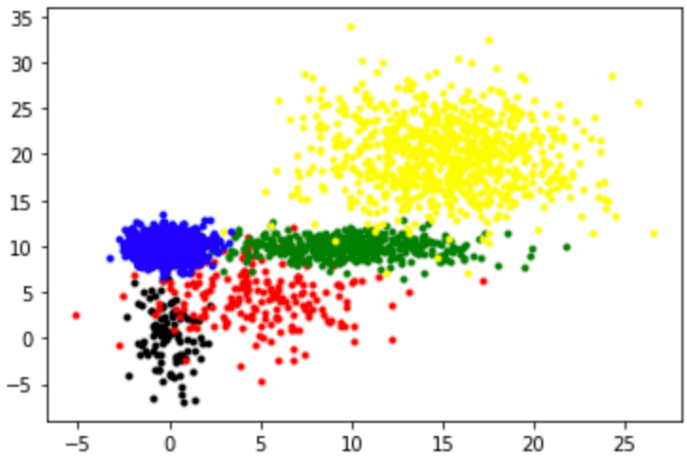
Preparing dataset

Define 5 overlapping Gaussian distributions based on mean and covariance:

```
import numpy as np
import matplotlib.pyplot as plt

mean1 = [0, 0]
cov1 = [[1, 0], [0, 10]]

mean2 = [5, 5]
cov2 = [[10, 0], [0, 10]]

mean3 = [10, 10]
cov3 = [[10, 0], [0, 1]]

mean4 = [15, 20]
cov4 = [[15, 0], [0,15]]

mean5 = [0, 10]
cov5 = [[1, 0], [0, 1]]
```

Sample 100, 200, 400, 800, 1000 data points from the distributions and plot the data points:

```
x11, x12 = np.random.multivariate_normal(mean1, cov1, 100).T
x21, x22 = np.random.multivariate_normal(mean2, cov2, 200).T
x31, x32 = np.random.multivariate_normal(mean3, cov3, 400).T
x41, x42 = np.random.multivariate_normal(mean4, cov4, 800).T
x51, x52 = np.random.multivariate_normal(mean5, cov5, 1000).T

fig, ax = plt.subplots()
#draw the train and test data
ax.scatter(x11,x12, marker='.',color='black')
ax.scatter(x21,x22, marker='.',color='red')
ax.scatter(x31,x32, marker='.',color='green')
ax.scatter(x41,x42, marker='.',color='yellow')
ax.scatter(x51,x52, marker='.',color='blue')
plt.show()
```



Combine the data points into a dataset:

```
from sklearn.model_selection import KFold

# create label for the data
y=[]
nums=[100,200,400,800,1000]
for i,num in enumerate(nums):
    for _ in range(num):
        y.append(i)
y=np.array(y)
# combine the data to a dataset with two features
x1=np.concatenate((x11, x21,x31,x41,x51), axis=0)
x2=np.concatenate((x12, x22,x32,x42,x52), axis=0)
X= np.vstack((x1, x2)).T
```

Classify the dataset

Use 10-fold cross validation and multi-class SVM to classify the datasets.

```
from sklearn import svm
from sklearn import metrics
kf = KFold(n_splits=10,shuffle=True)
acc_mean=0
for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    # create an instance of SVM and fit out data.
    C = 1.0  # SVM regularization parameter
    clf = svm.SVC(kernel='rbf', gamma=0.7, C=C)
    models = clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    acc =metrics.accuracy_score(y_test, y_pred)
    print(acc)
    acc_mean+=acc
print("mean accuracy: {}".format(acc_mean/10))
```

```
0.952
0.948
0.964
0.948
0.928
0.968
0.98
0.952
0.968
0.956
mean accuracy: 0.9564
```