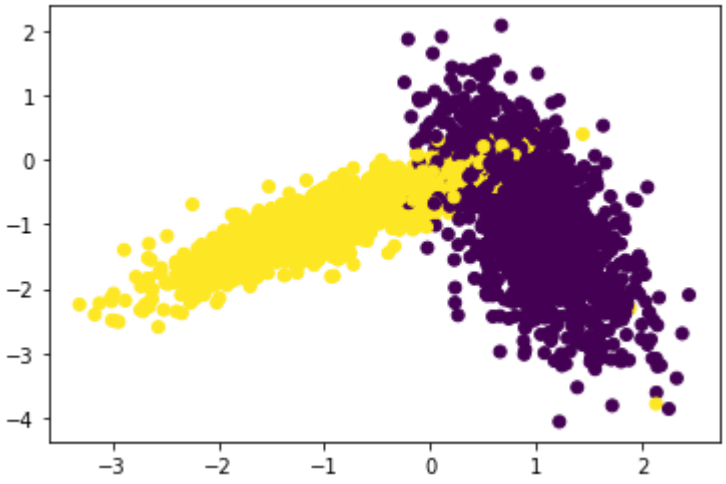
In [150]:

```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import numpy
from tqdm import tqdm
import numpy as np
from sklearn.metrics.pairwise import euclidean_distances
import random


x,y = make_classification(n_samples=10000, n_features=2, n_informative=2, n_redundant=
0, n_clusters_per_class=1, random_state=60)
X_train, X_test, y_train, y_test = train_test_split(x,y,stratify=y,random_state=42)

# del X_train,X_test
```

In [151]:

```
%matplotlib inline
import matplotlib.pyplot as plt
#colors = {0:'orange', 1:'blue'}
plt.scatter(X_test[:,0], X_test[:,1],c=y_test)
plt.show()
```



In [146]:

```
# Reference: https://www.kaggle.com/paulrohan2020/ml-algorithms-from-scratch-with-pure-python/comments
from sklearn.metrics import accuracy_score


def randomized_search_cv_custom(x_train, y_train, classifier, params, folds):


    trainscores = []
    testscores = []


    for k in tqdm(params['n_neighbors']):
        trainscores_folds = []
        testscores_folds = []

        for j in range(0, folds):

            elementsperfold = int(len(x_train) / folds)


            test_indices = list(set(list(range((elementsperfold * j), (elementsperfold*(j+1))))))

            train_indices = list(set(list(range(0, len(x_train)))) - set(test_indices))


            x_train_fold = x_train[train_indices]
            y_train_fold = y_train[train_indices]
            x_test_fold = x_train[test_indices]
            y_test_fold = y_train[test_indices]


            classifier.n_neighbors = k
            classifier.fit(x_train_fold, y_train_fold)

            y_predicted = classifier.predict(x_test_fold)
            testscores_folds.append(accuracy_score(y_test_fold, y_predicted))

            y_predicted = classifier.predict(x_train_fold)
            trainscores_folds.append(accuracy_score(y_train_fold, y_predicted))

        trainscores.append(np.mean(np.array(trainscores_folds)))
        testscores.append(np.mean(np.array(testscores_folds)))

    return trainscores, testscores, params
```

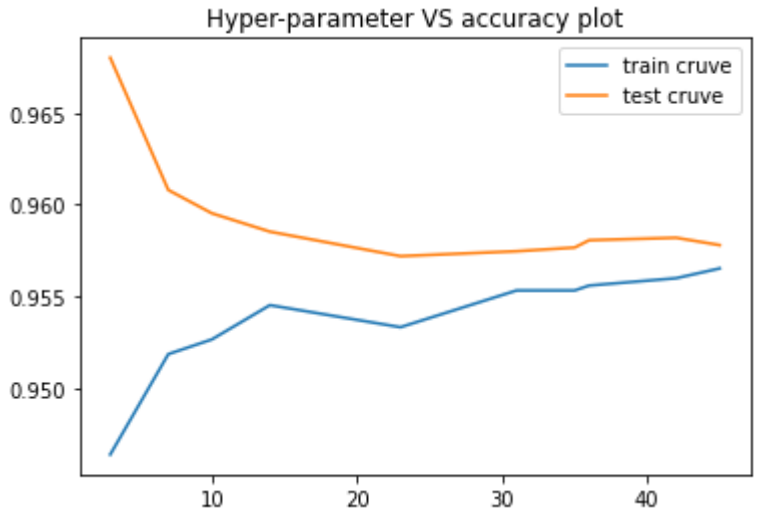
In [147]:

```
from sklearn.neighbors import KNeighborsClassifier
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

# Our Classifier is KNN, hence assign a variable to it.
neigh = KNeighborsClassifier()
params_range = 50
random_values_for_param_range = sorted(random.sample(range(1, param_range), 10))
params = {'n_neighbors': random_values_for_param_range}

folds = 3

# Now invoking our custom function randomized_search_cv_custom(x_train,y_train,classifier, param_range, num_of_total_fold) and store the returned values
testscores, trainscores, params = randomized_search_cv_custom(X_train, y_train, neigh, params, folds)

print('trainscores are: ', trainscores)
print('testscores are: ', testscores)
print('params are:', params)
```

```
100%|██████████| 10/10 [00:15<00:00,  1.55s/it]

trainscores are:  [0.9464, 0.9518666666666666, 0.9526666666666667, 0.95453
33333333333, 0.9533333333333333, 0.9553333333333334, 0.9553333333333334,
0.9556, 0.956, 0.9565333333333333]
testscores are:  [0.968, 0.9608, 0.9595333333333333, 0.9585333333333335,
0.9571999999999999, 0.9574666666666666, 0.9576666666666666, 0.958066666666
6667, 0.9582, 0.9578000000000001]
params are: {'n_neighbors': [3, 7, 10, 14, 23, 31, 35, 36, 42, 45]}
```

In [148]:

```
plt.plot(params['n_neighbors'],trainscores, label='train cruve')
plt.plot(params['n_neighbors'],testscores, label='test cruve')
plt.title('Hyper-parameter VS accuracy plot')
plt.legend()
plt.show()
```



In [138]:

```
# understanding this code line by line is not that importent
def plot_decision_boundary(X1, X2, y, clf):
        # Create color maps
    cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
    cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])

    x_min, x_max = X1.min() - 1, X1.max() + 1
    y_min, y_max = X2.min() - 1, X2.max() + 1

    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02), np.arange(y_min, y_max, 0.02))
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.figure()
    plt.pcolormesh(xx, yy, Z, cmap=cmap_light)
    # Plot also the training points
    plt.scatter(X1, X2, c=y, cmap=cmap_bold)

    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.title("2-Class classification (k = %i)" % (clf.n_neighbors))
    plt.show()
```

In [149]:

```
#Best Hyperparameter = 45
from matplotlib.colors import ListedColormap
neigh = KNeighborsClassifier(n_neighbors = 45)
neigh.fit(X_train, y_train)
plot_decision_boundary(X_train[:, 0], X_train[:, 1], y_train, neigh)
```