SVM

Name: Tran Phat Dat

ID Student: B2014971

1) Implement **SVM** using **scikit-learn** library in Python. The program requires 4 parameters:

- file name of trainset
- file name of testset
- *learning rate* /
- maximum number of epochs maxit

Dataset with *m* examples, *n* dimensions (attribute), *2* classes (*-1*, *+1*), is in the format:

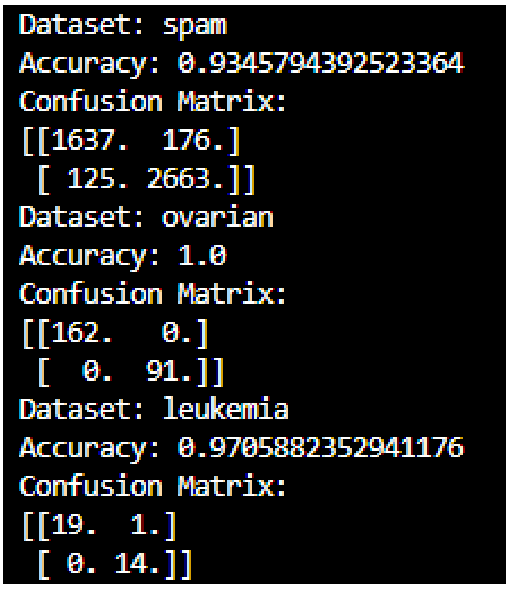
val_i$_1$_a$_1$ val_i$_1$_a$_2$ … val_i$_1$_a$_n$ class_i1
val_i$_2$_a$_1$ val_i$_2$_a$_2$ … val_i$_2$_a$_n$ class_i2
…
val_i$_m$_a$_1$ val_i$_m$_a$_2$ … val_i$_m$_a$_n$ class_im

The program reports the classification results (accuracy, confusion matrix) with different *learning rate* / *and maximum number of epochs maxit for 3 datasets:*

- Spam (**hold-out**)
- Leukemia (**.trn** for trainset, **.tst** for testset)
- Ovarian (**hold-out**)

Answer:

	Spam	**Leukemia**	**Ovarian**
Accuracy	93.46%	97.06%	100%

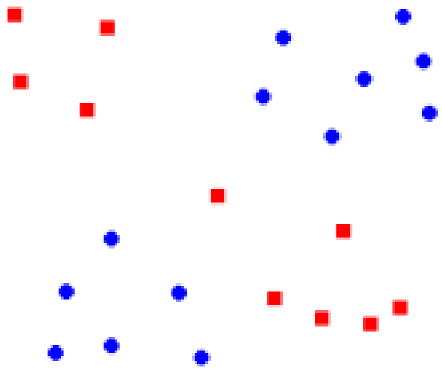


```
Dataset: spam
Accuracy: 0.9345794392523364
Confusion Matrix:
[[1637.  176.]
 [ 125. 2663.]]
Dataset: ovarian
Accuracy: 1.0
Confusion Matrix:
[[162.   0.]
 [  0.  91.]]
Dataset: leukemia
Accuracy: 0.9705882352941176
Confusion Matrix:
[[19.  1.]
 [ 0. 14.]]
```

2) Implement the training program using **SVC** in **scikit-learn** library to classify the dataset

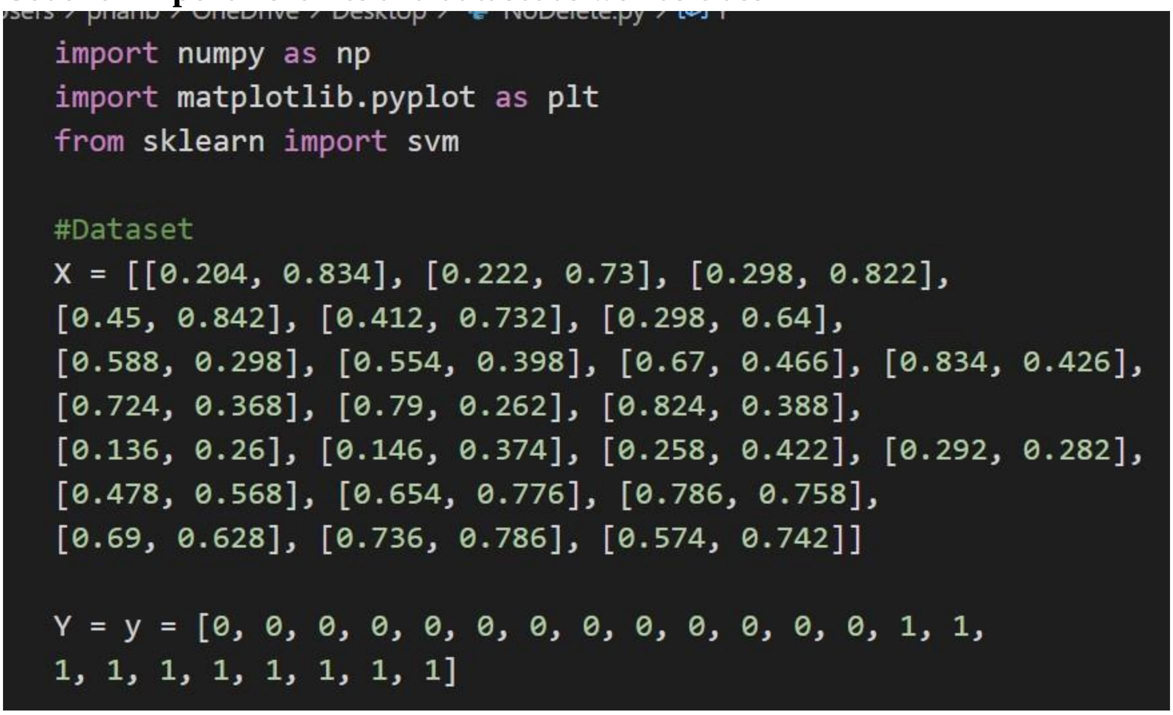
X1	**X2**	**Class**
0.204000	0.834000	0
0.222000	0.730000	0
0.298000	0.822000	0
0.450000	0.842000	0
0.412000	0.732000	0
0.298000	0.640000	0
0.588000	0.298000	0
0.554000	0.398000	0
0.670000	0.466000	0
0.834000	0.426000	0
0.724000	0.368000	0
0.790000	0.262000	0
0.824000	0.338000	0
0.136000	0.260000	1
0.146000	0.374000	1
0.258000	0.422000	1
0.292000	0.282000	1
0.478000	0.568000	1
0.654000	0.776000	1
0.786000	0.758000	1
0.690000	0.628000	1
0.736000	0.786000	1

0.574000	0.742000	1



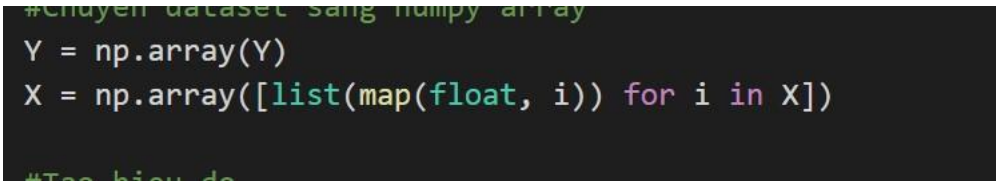
The program visualises datapoints with the separation boundary of the classifier.

Code for **import libraries** and **dataset** as well as **class**



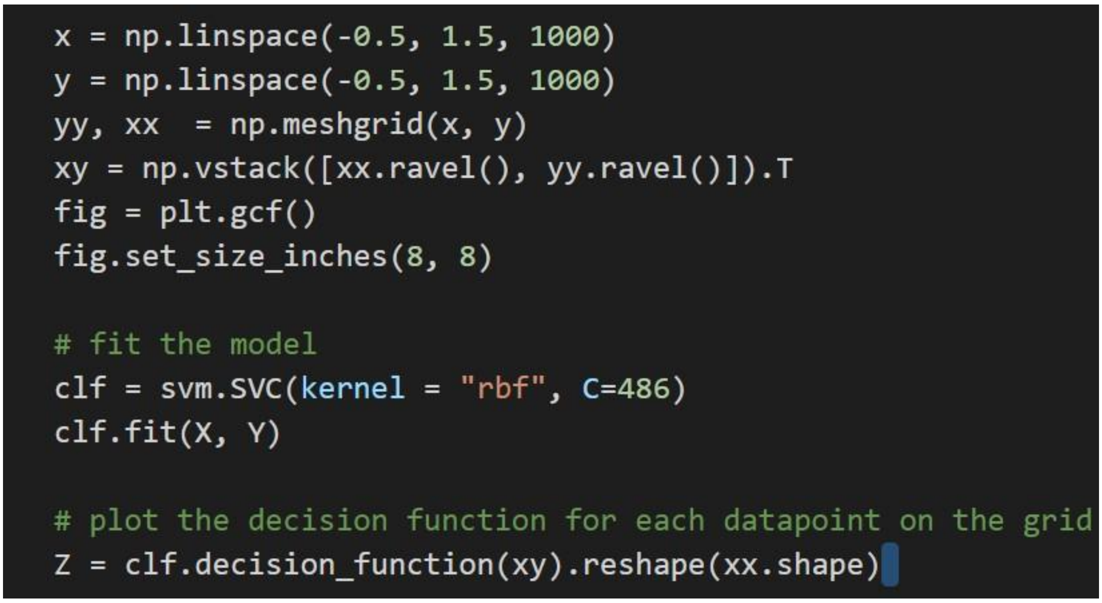
```
sers > phanb > OneDrive > Desktop > NoDelete.py > f
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm

#Dataset
X = [[0.204, 0.834], [0.222, 0.73], [0.298, 0.822],
[0.45, 0.842], [0.412, 0.732], [0.298, 0.64],
[0.588, 0.298], [0.554, 0.398], [0.67, 0.466], [0.834, 0.426],
[0.724, 0.368], [0.79, 0.262], [0.824, 0.388],
[0.136, 0.26], [0.146, 0.374], [0.258, 0.422], [0.292, 0.282],
[0.478, 0.568], [0.654, 0.776], [0.786, 0.758],
[0.69, 0.628], [0.736, 0.786], [0.574, 0.742]]

Y = y = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1]
```

Code for converting **class** and **dataset** to **numpy array**



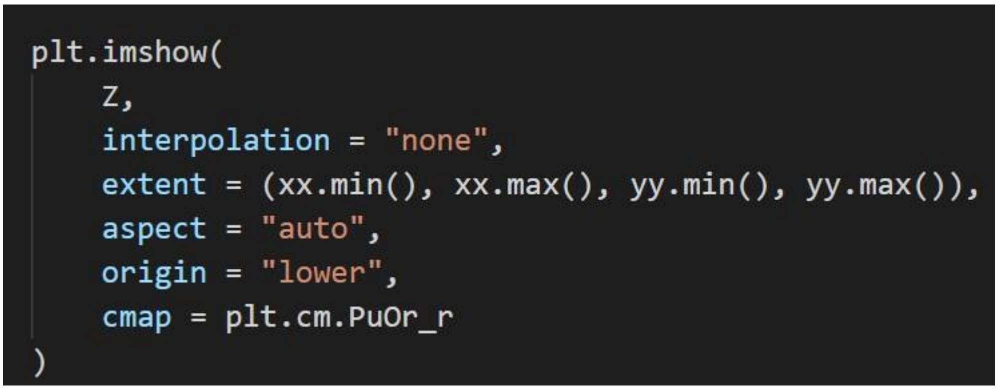
```
#Chuyen dataset sang numpy array
Y = np.array(Y)
X = np.array([list(map(float, i)) for i in X])

#Tao bieu do
```

Code for building the graph



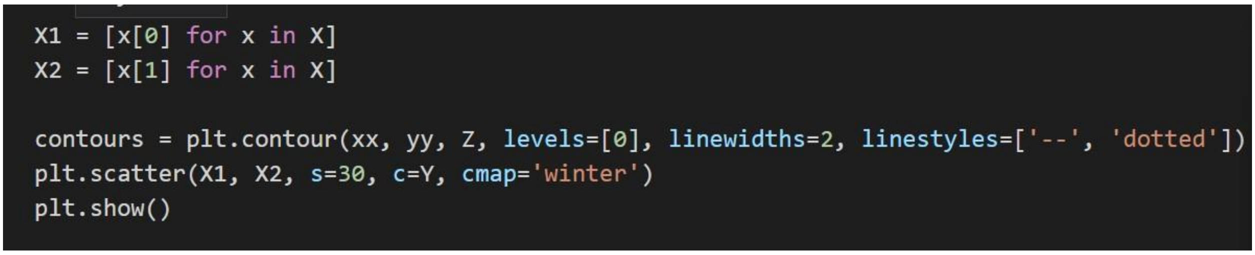
```
x = np.linspace(-0.5, 1.5, 1000)
y = np.linspace(-0.5, 1.5, 1000)
yy, xx  = np.meshgrid(x, y)
xy = np.vstack([xx.ravel(), yy.ravel()]).T
fig = plt.gcf()
fig.set_size_inches(8, 8)

# fit the model
clf = svm.SVC(kernel = "rbf", C=486)
clf.fit(X, Y)

# plot the decision function for each datapoint on the grid
Z = clf.decision_function(xy).reshape(xx.shape)
```

Code for decoration



```
plt.imshow(
    Z,
    interpolation = "none",
    extent = (xx.min(), xx.max(), yy.min(), yy.max()),
    aspect = "auto",
    origin = "lower",
    cmap = plt.cm.PuOr_r
)
```

And this is the code for calling the function



```
X1 = [x[0] for x in X]
X2 = [x[1] for x in X]

contours = plt.contour(xx, yy, Z, levels=[0], linewidths=2, linestyles=['--', 'dotted'])
plt.scatter(X1, X2, s=30, c=Y, cmap='winter')
plt.show()
```

This is my result:

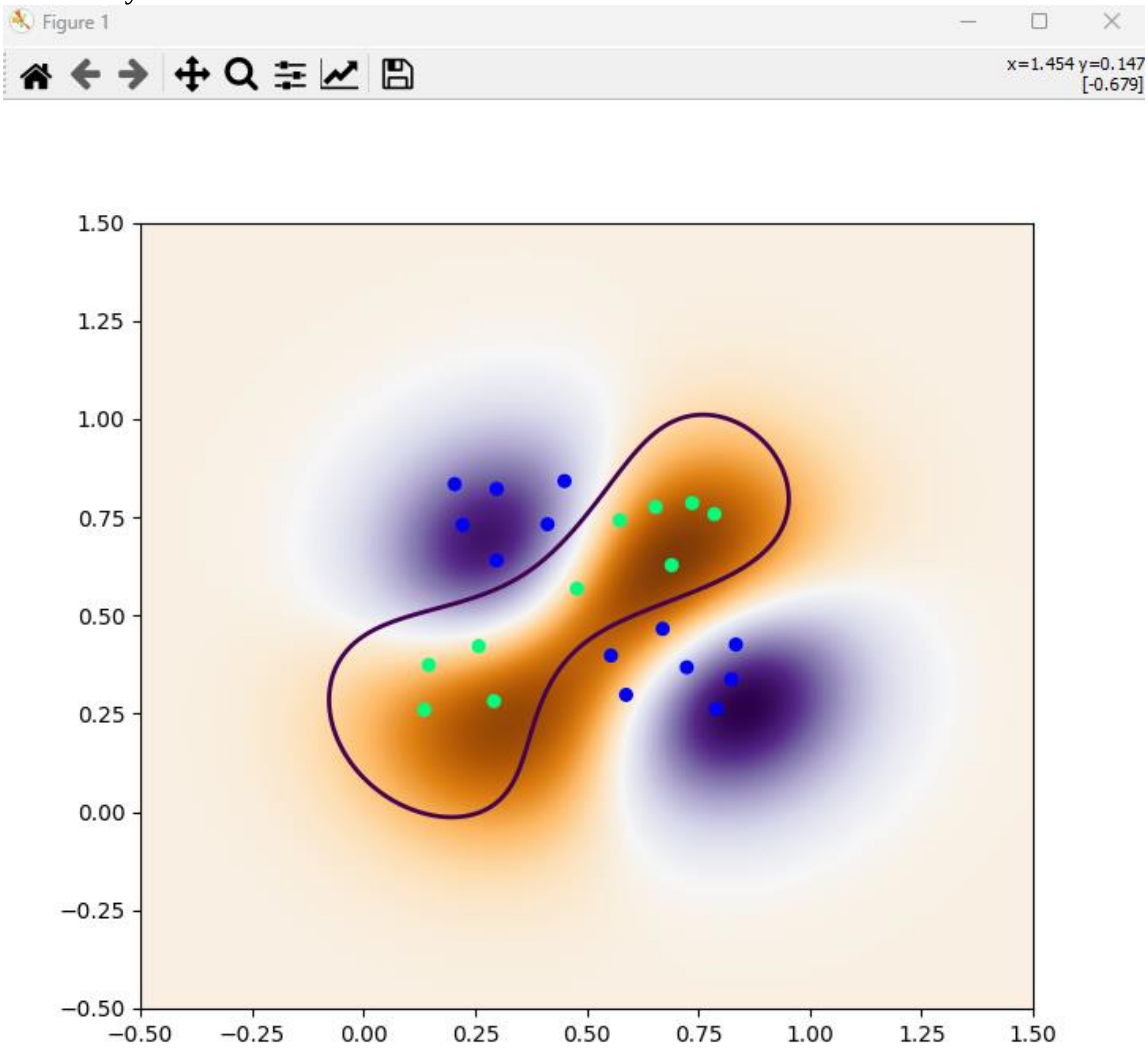


Figure 1
x=1.454 y=0.147
[-0.679]
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50