```
# -*- coding: utf-8 -*-
"""
Created on Fri Mar 16 18:18:00 2018

@author: Mingjun
"""

from sklearn import datasets, preprocessing
import numpy as np
import math
import operator
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
from sklearn import svm

rate = 0.8

# This function is to load the toy data, suffule it and split it into training and test
# data set
def get_data():
    # load the data
    data = pd.read_csv('svmdata.csv',header=None).values

    # shuffle data rows before splitting
    np.random.shuffle(data) # shuffle rows
    nos_samples = len(data)

    # training data
    train = data[0:int(round(nos_samples*rate)),:]

    # test data
    test = data[int(round(nos_samples*rate)):,:]
    return train, test

train_sample, test_sample = get_data()

def plot_scatter(data):
    # the scatter plots, red for postive examples, blue for negative
    feature1 = data[:,0]
    feature2 = data[:,1]
    targets = data[:,2]
    colors = ['blue','red']
    plt.figure(1)
    plt.scatter(feature1,feature2,alpha=1,c=targets,cmap=matplotlib.colors.ListedColormap(colors

plot_scatter(train_sample)

X_train = train_sample[:,0:2]
y_train = train_sample[:,2]
X_test = test_sample[:,0:2]
y_test = test_sample[:,2]

# fit the svm model
kernel = 'poly'
clf = svm.SVC(kernel=kernel, gamma=10)
clf.fit(X_train,y_train)

plt.figure(2)
plt.clf()
plt.scatter(train_sample[:, 0], train_sample[:, 1], c=train_sample[:,2], zorder=10, cmap=plt.cm.
```

```
                edgecolor='k', s=20)

# Circle out the test data
plt.scatter(X_test[:, 0], X_test[:, 1], s=80, facecolors='none',
                zorder=10, edgecolor='k')

plt.axis('tight')
x_min = train_sample[:, 0].min()
x_max = train_sample[:, 0].max()
y_min = train_sample[:, 1].min()
y_max = train_sample[:, 1].max()

XX, YY = np.mgrid[x_min:x_max:200j, y_min:y_max:200j]
Z = clf.decision_function(np.c_[XX.ravel(), YY.ravel()])

# Put the result into a color plot
Z = Z.reshape(XX.shape)
plt.pcolormesh(XX, YY, Z > 0, cmap=plt.cm.Paired)
plt.contour(XX, YY, Z, colors=['k', 'k', 'k'],
            linestyles=['--', '-', '--'], levels=[-.5, 0, .5])

# return support vectors
support_vectors = clf.support_vectors_
print(clf.support_vectors_)

# the predicted results
y_predicted = clf.predict(X_test)
print(y_predicted)
```