```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Example 6.3 SVM polynomial kernel

@author: Austin R.J. Downey
"""

import IPython as IP
IP.get_ipython().run_line_magic('reset', '-sf')

import numpy as np
import matplotlib.pyplot as plt
import sklearn as sk
from sklearn import datasets
from sklearn import pipeline
from sklearn import svm

plt.close('all')


#%% Build and plot the data

X, y = sk.datasets.make_moons(n_samples=100, noise=0.25, random_state=2)

plt.figure()
plt.plot(X[:,0][y==0],X[:,1][y==0],'s')
plt.plot(X[:,0][y==1],X[:,1][y==1],'d')
plt.xlabel("$x_1$")
plt.ylabel("$x_2$")
plt.grid(True)


#%% SVM polynominal features
svm_clf = sk.pipeline.Pipeline([
        ("scaler", sk.preprocessing.StandardScaler()),
        ("svm_clf", sk.svm.SVC(kernel="poly", degree=3, coef0=1, C=100))
    ])
svm_clf.fit(X, y)


#%% Prepare fro plotting
# make the 2d space for the color
x1 = np.linspace(-2, 3, 200)
x2 = np.linspace(-2, 2, 100)
x1_grid, x2_grid = np.meshgrid(x1, x2)

# calculate the binary decions and predection values
X2 = np.vstack((x1_grid.ravel(), x2_grid.ravel())).T
y_decision = svm_clf.decision_function(X2).reshape(x1_grid.shape)

# plot the figure (the show up in the figure enviorment defined above
con_lines = [-30,-20,-10,-5,0,5,10,20,30]
contour = plt.contour(x1_grid, x2_grid, y_decision, con_lines, cmap=plt.cm.brg)
plt.clabel(contour, inline=1, fontsize=12)


```