```
from matplotlib import pyplot as plt
from sklearn.svm import SVC
from scipy.io import loadmat
import numpy as np
from process_email import *
import codecs
from get_vocab_dict import getVocabDict
from os import listdir
from os.path import isdir
from os.path import isfile, join

porcentaje_entrenamiento = 0.6
porcentaje_validacion = 0.2
porcentaje_test = 0.2

def draw_decisition_boundary(X, y, svm):
    x1 = np.linspace(X[:, 0].min(), X[:, 0].max(), 100)
    x2 = np.linspace(X[:, 1].min(), X[:, 1].max(), 100)
    x1, x2 = np.meshgrid(x1, x2)
    yp = svm.predict(np.array([x1.ravel(),x2.ravel()]).T).reshape(x1.shape)
    pos = (y == 1).ravel()
    neg = (y == 0).ravel()
    plt.scatter(X[pos, 0], X[pos, 1], color='black', marker='+')
    plt.scatter(X[neg, 0], X[neg, 1], color='yellow',
    edgecolors='black', marker='o')
    plt.contour(x1, x2, yp)

def kernel_lineal():
    data = loadmat('ex6data1.mat')
    X, y = data['X'], data['y']

    svm = SVC(kernel='linear', C=1.0)
    svm.fit(X, y.ravel())

    plt.figure()
    draw_decisition_boundary(X, y, svm)
    plt.show()

    svm = SVC(kernel='linear', C=100)
    svm.fit(X, y.ravel())
```

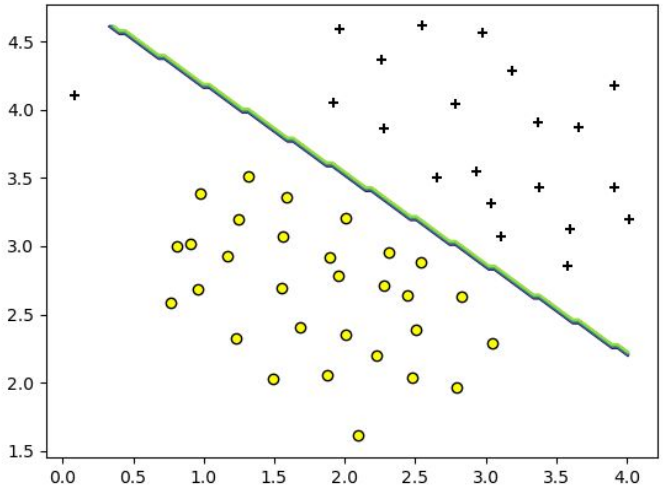
```
    plt.figure()
    draw_decisition_boundary(X, y, svm)
    plt.show()

def kernel_gaussiano():
    data = loadmat('ex6data2.mat')
    X, y = data['X'], data['y']
    sigma = 0.1

    svm = SVC(kernel = 'rbf' , C= 1, gamma= (1 / ( 2 * sigma ** 2)) )
    svm.fit(X, y.ravel())

    plt.figure()
    draw_decisition_boundary(X, y, svm)
    plt.show()

def eleccion_parametros_C_y_Sigma():
    data = loadmat('ex6data3.mat')
    X, y, Xval, yval = data['X'], data['y'], data['Xval'], data['yval']
    possible_values = [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]
    C_value, sigma = 0, 0
    max_score = 0
    best_svm = None

    for i in range(len(possible_values)):
        C_value = possible_values[i]
        for j in range(len(possible_values)):
            sigma = possible_values[j]
            svm = SVC(kernel = 'rbf' , C = C_value, gamma= (1 / ( 2 * sigma **
2)))
            svm.fit(X, y.ravel())
            current_score = svm.score(Xval, yval) #calcula el score con los
ejemplos de validacion (mayor score, mejor es el svm)
            if current_score > max_score:
                max_score = current_score
                best_svm = svm

    plt.figure()
    draw_decisition_boundary(X, y, best_svm)
```

```
    plt.show()

def procesar_mail(path, words_dictionary):
    email_contents = codecs.open(path, 'r', encoding='utf-8',
errors='ignore').read()
    email = email2TokenList(email_contents)
    #email = palabras clave del email

    email_representation = np.zeros((1, len(words_dictionary)))
    for i in range(len(email)):
        if email[i] in words_dictionary:
            index = words_dictionary[email[i]] - 1
            email_representation[0, index] = 1

    return email_representation

def getEmailsPathFromDirectory(directoryPath):
        """
        Iterates through the files of directoryPath and returns them
        """
        filesPath = [directoryPath + '/' + f for f in listdir(directoryPath)
if isfile(join(directoryPath, f))]
        return filesPath

def getSets(spamMailsPath, hardHamMailsPath, easyHamMailsPath,
words_dictionary):
    total_size = len(spamMailsPath) + len(hardHamMailsPath) +
len(easyHamMailsPath)
    training_size = total_size * porcentaje_entrenamiento
    val_size = total_size * porcentaje_validacion
    test_size = total_size * porcentaje_test

    #X = np.array(procesar_mail(spamMailsPath[0], words_dictionary),
dtype=float)
    X = np.empty((0, len(words_dictionary)))
    y = np.array([[]], dtype=float)
    Xval = np.empty((0, len(words_dictionary)))
    yval = np.array([[]], dtype=float)
    Xtest = np.empty((0, len(words_dictionary)))
    ytest = np.array([[]], dtype=float)
```

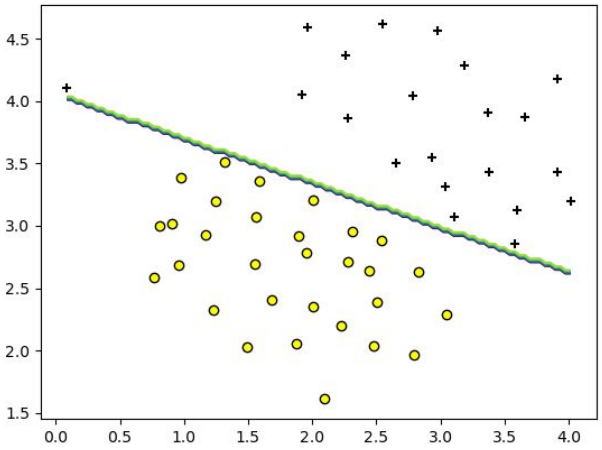
```
    for i in range(len(spamMailsPath)):
        if i < len(spamMailsPath) * porcentaje_entrenamiento:
            #todo cambiar todos los bucles y la X esta mal, se appendean todos
juntos :P
            X = np.vstack((X, procesar_mail(spamMailsPath[i],
words_dictionary)))
            y = np.append(y, [[1]], axis = 1)
        elif i < len(spamMailsPath) * porcentaje_entrenamiento +
len(spamMailsPath) * porcentaje_validacion:
            Xval = np.vstack((Xval, procesar_mail(spamMailsPath[i],
words_dictionary)))
            yval = np.append(yval, [[1]], axis = 1)
        else:
            Xtest = np.vstack((Xtest, procesar_mail(spamMailsPath[i],
words_dictionary)))
            ytest = np.append(ytest, [[1]], axis = 1)

    for i in range(len(easyHamMailsPath)):
        if i < len(easyHamMailsPath) * porcentaje_entrenamiento:
            X = np.vstack((X, procesar_mail(easyHamMailsPath[i],
words_dictionary)))
            y = np.append(y, [[0]], axis=1)
        elif i < len(easyHamMailsPath) * porcentaje_entrenamiento +
len(easyHamMailsPath) * porcentaje_validacion:
            Xval = np.vstack((Xval, procesar_mail(easyHamMailsPath[i],
words_dictionary)))
            yval = np.append(yval, [[0]], axis=1)
        else:
            Xtest = np.vstack((Xtest, procesar_mail(easyHamMailsPath[i],
words_dictionary)))
            ytest = np.append(ytest, [[0]], axis=1)

    for i in range(len(hardHamMailsPath)):
        if i < len(hardHamMailsPath) * porcentaje_entrenamiento:
            X = np.vstack((X, procesar_mail(hardHamMailsPath[i],
words_dictionary)))
            y = np.append(y, [[0]], axis = 1)
        elif i < len(hardHamMailsPath) * porcentaje_entrenamiento +
len(hardHamMailsPath) * porcentaje_validacion:
```

```
            Xval = np.vstack((Xval, procesar_mail(hardHamMailsPath[i],
words_dictionary)))
            yval = np.append(yval, [[0]], axis=1)
        else:
            Xtest = np.vstack((Xtest, procesar_mail(hardHamMailsPath[i],
words_dictionary)))
            ytest = np.append(ytest, [[0]], axis=1)

    return X, y, Xval, yval, Xtest, ytest

def deteccionDeSpam():
    words_dictionary = getVocabDict()
    spamMailsPath = getEmailsPathFromDirectory('./spam')
    hardHamMailsPath = getEmailsPathFromDirectory('./hard_ham')
    easyHamMailsPath = getEmailsPathFromDirectory('./easy_ham')
    X, y, Xval, yval, Xtest, ytest = getSets(spamMailsPath, hardHamMailsPath,
easyHamMailsPath, words_dictionary)

    possible_values = [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]
    C_value, sigma = 0, 0
    max_score = 0
    best_svm = None

    for i in range(len(possible_values)):
        C_value = possible_values[i]
        for j in range(len(possible_values)):
            sigma = possible_values[j]
            svm = SVC(kernel = 'rbf' , C = C_value, gamma= (1 / ( 2 * sigma **
2)))
            svm.fit(X, y.ravel())
            current_score = svm.score(Xval, yval.T) #calcula el score con los
ejemplos de validacion (mayor score, mejor es el svm)
            if current_score > max_score:
                max_score = current_score
                best_svm = svm

    print("Score de clasificar los ejemplos de prueba: " +
str(best_svm.score(Xtest, ytest.T)))
```

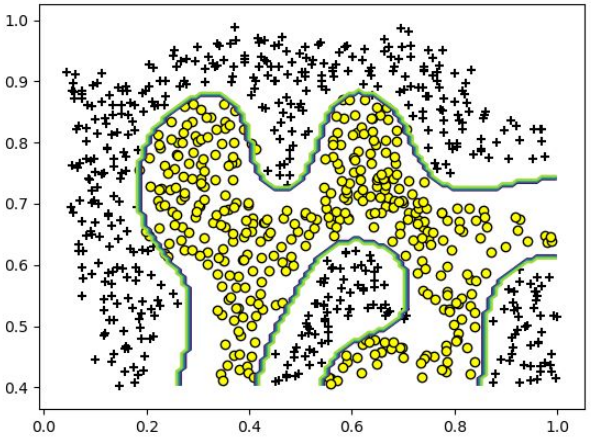
```
kernel_lineal()
kernel_gaussiano()
eleccion_parametros_C_y_Sigma()
deteccionDeSpam()
```



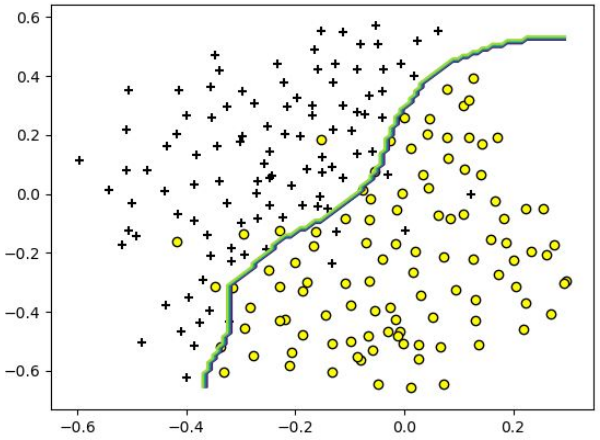
Gráfica obtenida con el kernel lineal (C = 1)



Gráfica obtenida con el kernel lineal (C = 100)



Gráfica obtenida con kernel gaussiano



Gráfica obtenida tras seleccionar los mejores parámetros de C y sigmas

DETECCIÓN DE SPAM

Score obtenido dividiendo el total de los ejemplos en un 60% para los ejemplos de entrenamiento, 20% para los ejemplos de validación y 20% para los ejemplos de prueba: 0.96.

Score obtenido dividiendo el total de los ejemplos en un 90% para los ejemplos de entrenamiento, 5% para los ejemplos de validación y 5% para los ejemplos de prueba: 0.84.

Score obtenido dividiendo el total de los ejemplos en un 60% para los ejemplos de entrenamiento, 30% para los ejemplos de validación y 10% para los ejemplos de prueba: 0.95.

Score obtenido dividiendo el total de los ejemplos en un 40% para los ejemplos de entrenamiento, 30% para los ejemplos de validación y 30% para los ejemplos de prueba: 0.86.