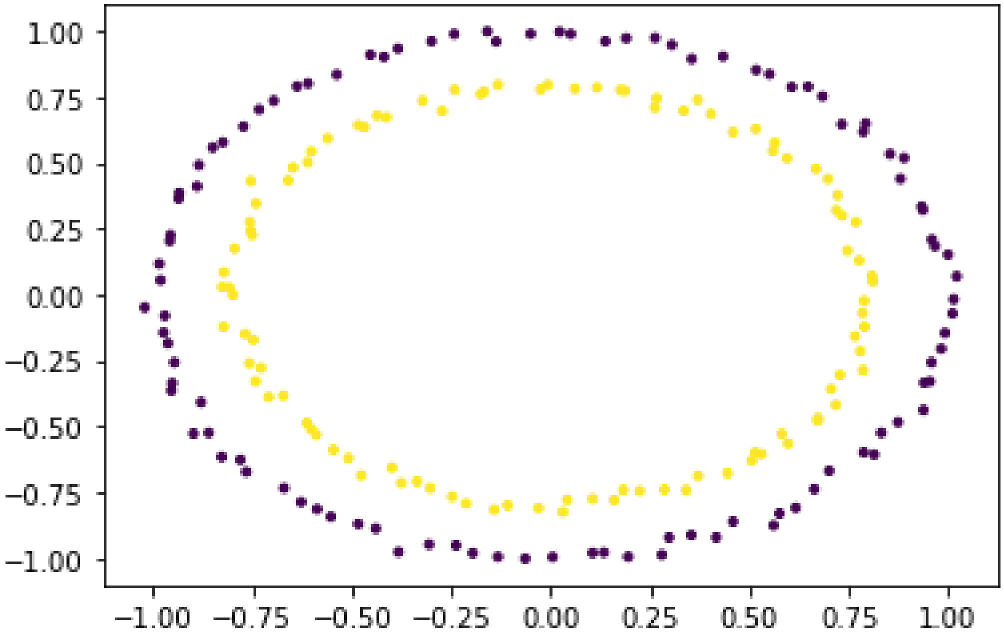
Kernel_SVM

April 26, 2022

[67]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from matplotlib.backends.backend_pdf import PdfPages
from sklearn.datasets import make_circles
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
```

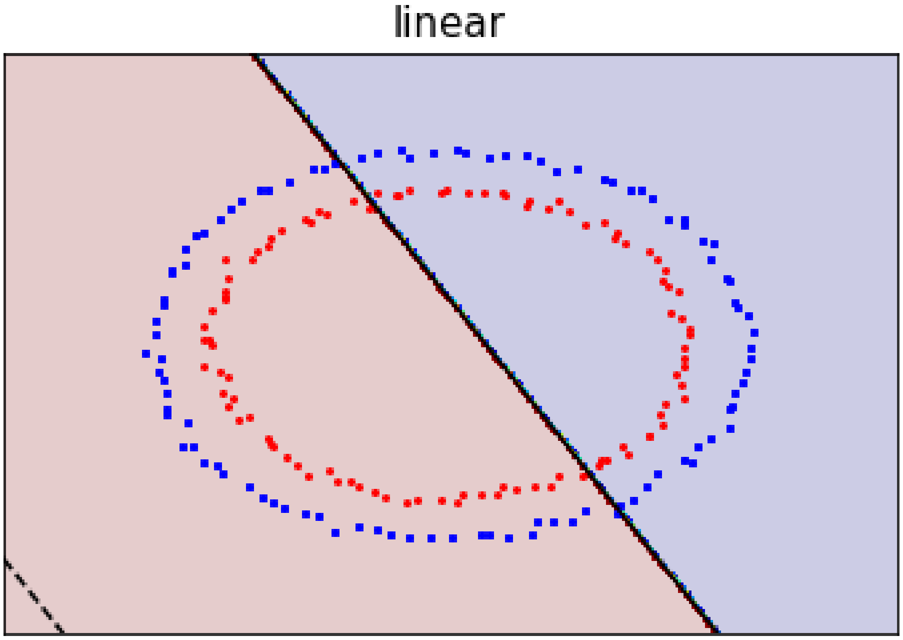
1 Example 1

[2]:
```
# Generate dataset and targets
X, Y = make_circles(n_samples = 200, noise = 0.02)

# visualizing data
plt.scatter(X[:, 0], X[:, 1], c = Y, marker = '.')
plt.show()
```

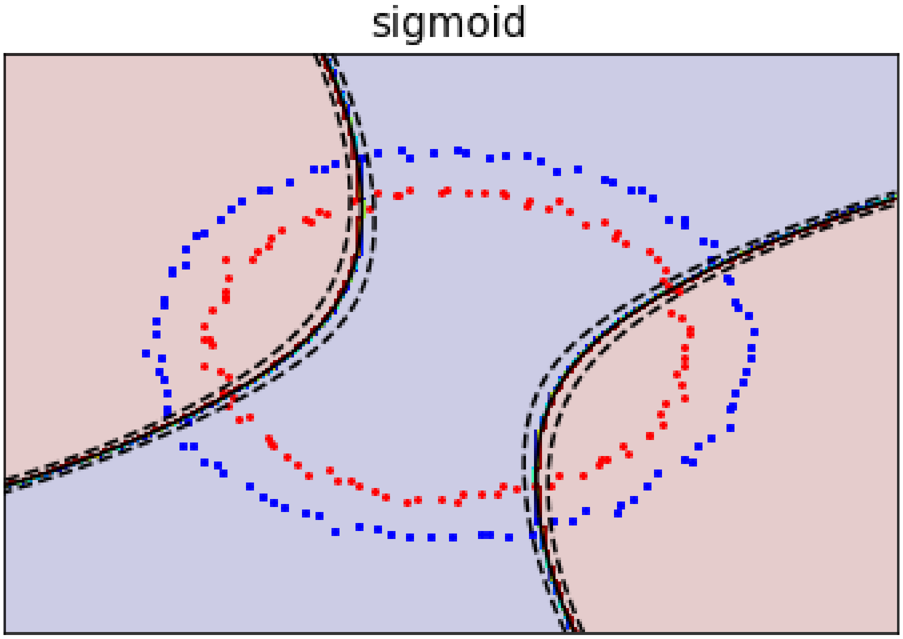


[3]:

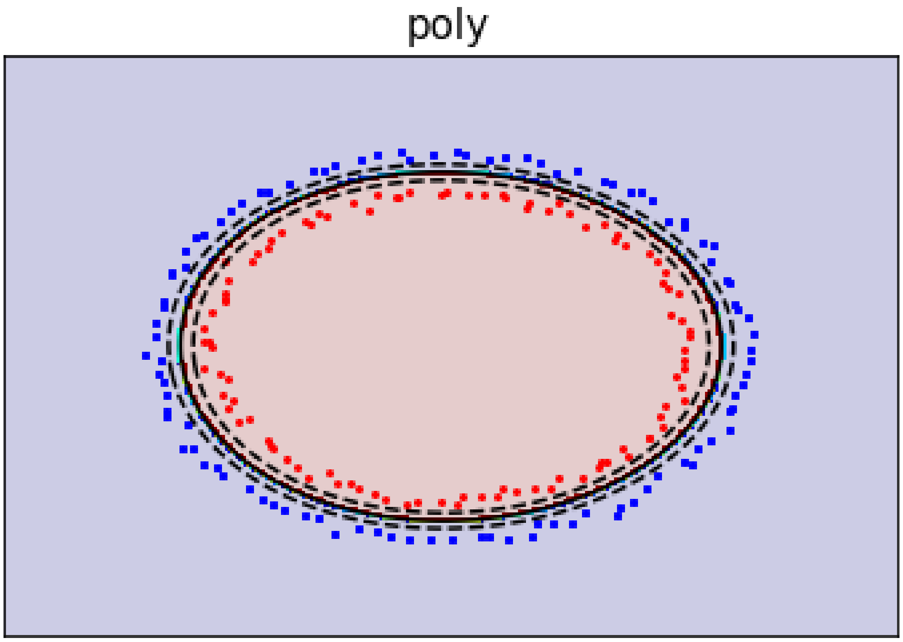
```
fignum = 1

# fit the model
for kernel in ('linear','sigmoid', 'poly', 'rbf'):
    clf = svm.SVC(kernel=kernel, gamma=1, coef0 = 1)
    clf.fit(X, Y)
    with PdfPages(kernel + '3.pdf') as pdf:
        # plot the line, the points, and the nearest vectors to the plane
        fig, ax = plt.subplots()
        plt.figure(fignum, figsize=(5, 5))
        plt.clf()

        plt.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[:, 1],␣
 ↪s=80,
                    facecolors='None')
        plt.plot(X[Y==0, 0], X[Y==0, 1], 'bs', markersize = 2)
        plt.plot(X[Y==1, 0], X[Y==1, 1], 'ro', markersize = 2)

        plt.axis('tight')
        x_min = -1.5
        x_max = 1.5
        y_min = -1.5
        y_max = 1.5

        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.figure(fignum, figsize=(5, 5))
        CS = plt.contourf(XX, YY, np.sign(Z), 200, cmap='jet', alpha = .2)
        plt.contour(XX, YY, Z, colors=['k', 'k', 'k'], linestyles=['--', '-',␣
 ↪'--'],
                    levels=[-.5, 0, .5])
        plt.title(kernel, fontsize = 15)
        plt.xlim(x_min, x_max)
        plt.ylim(y_min, y_max)

        plt.xticks(())
        plt.yticks(())
        fignum = fignum + 1
        pdf.savefig()
plt.show()
```



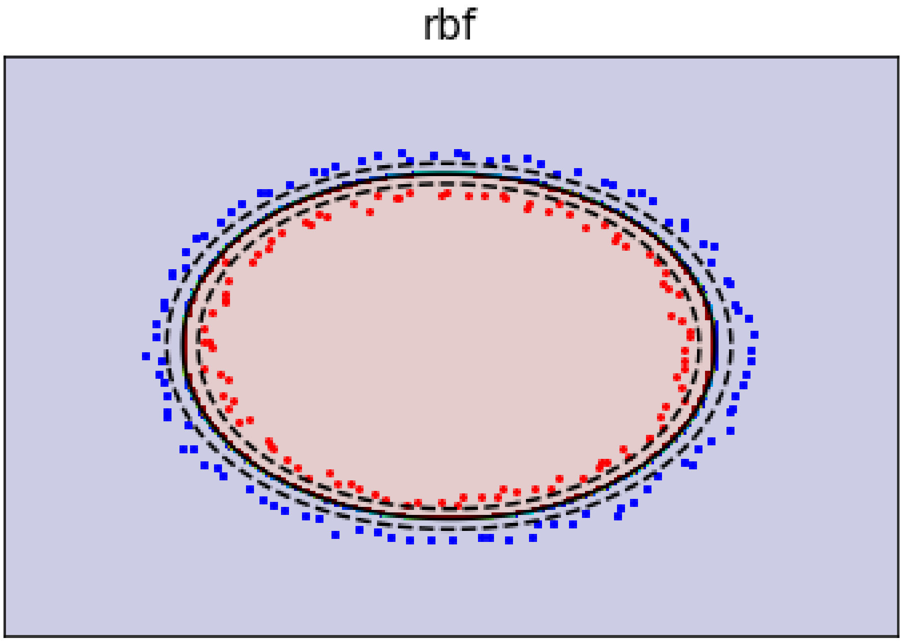
linear



sigmoid



poly



rbf

[2]:
```
import os
os.chdir('/home/nhanta/ML/data/kernel')
```

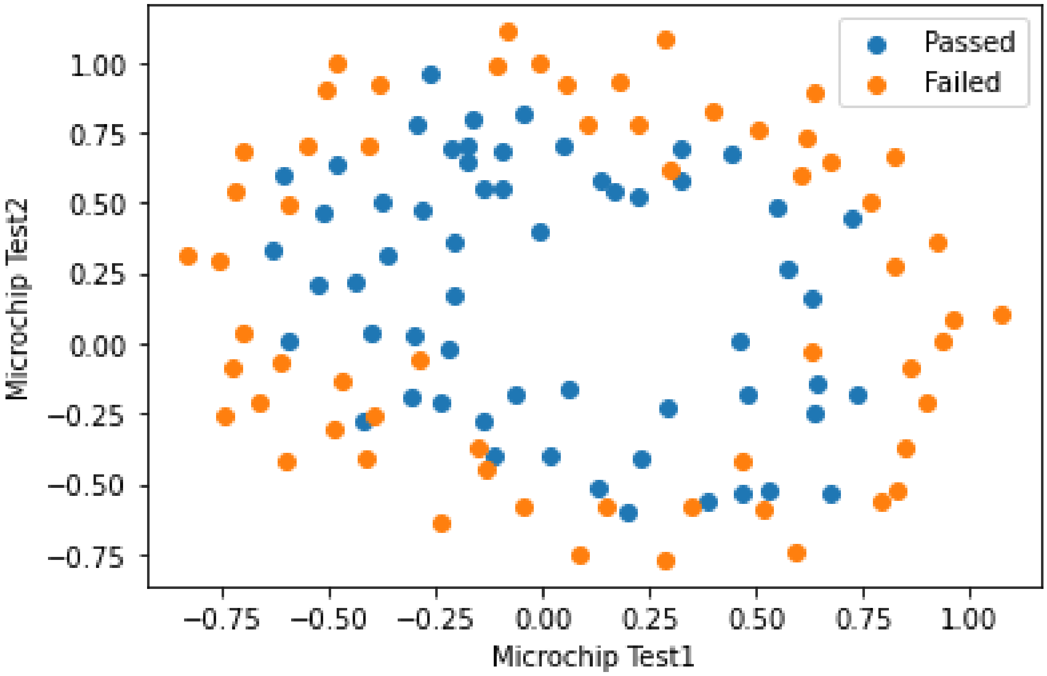
[3]:
```
import pandas as pd
```

[7]:
```
t = pd.read_csv('ex2data2.txt', header = None)
```

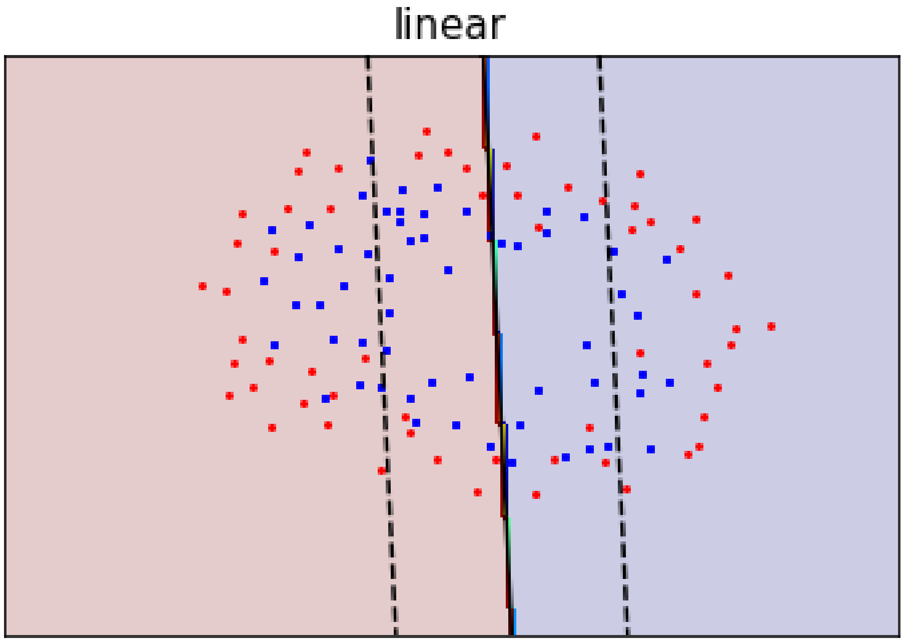
[10]:
```
X = t.iloc[:, [0,1]]
```

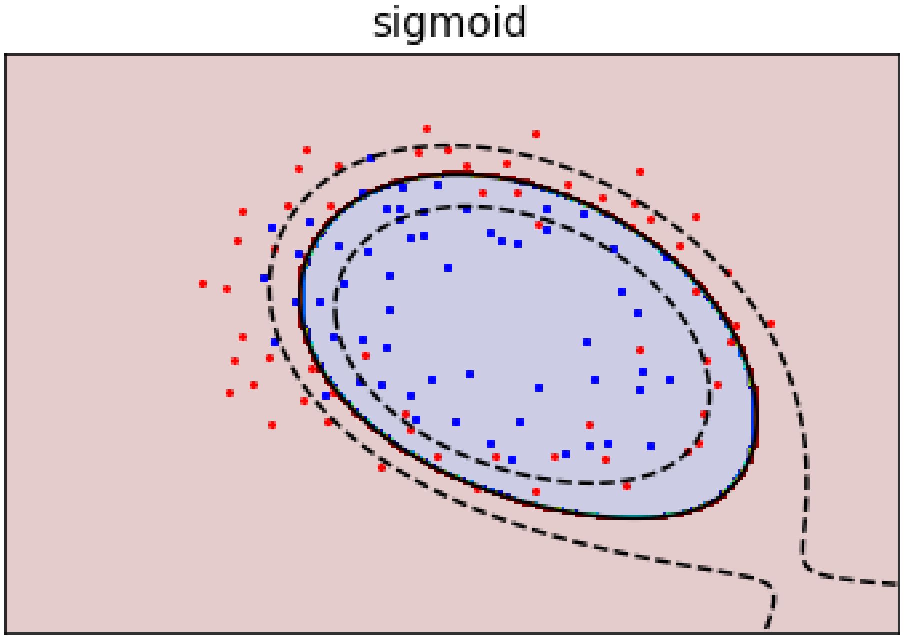
[14]:
```
Y = t.iloc[:, 2]
```

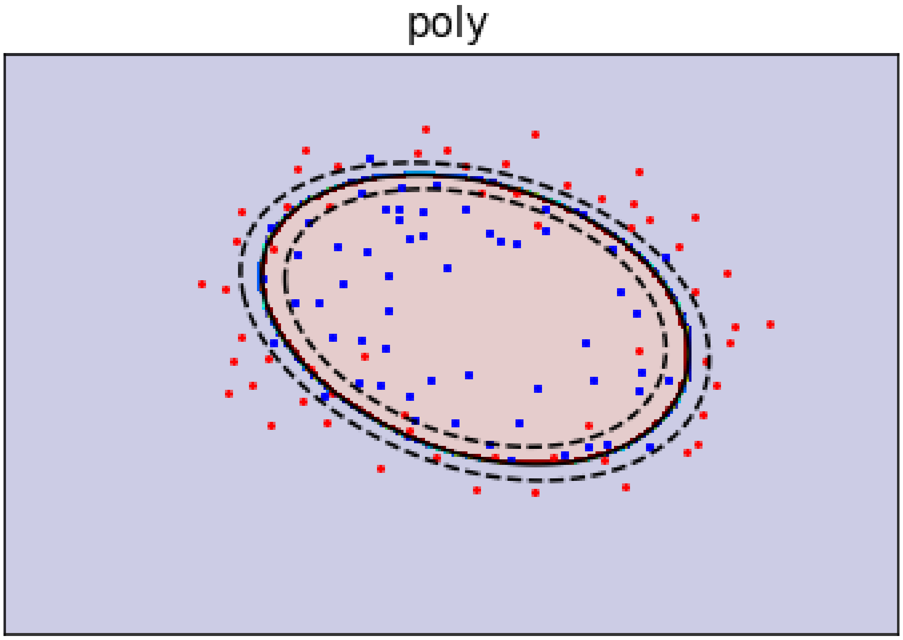
[18]:
```
# visualizing data
mask = Y == 1
passed = plt.scatter(X[mask][0].values, X[mask][1].values)
failed = plt.scatter(X[~mask][0].values, X[~mask][1].values)
plt.xlabel('Microchip Test1')
plt.ylabel('Microchip Test2')
plt.legend((passed, failed), ('Passed', 'Failed'))
plt.show()
```



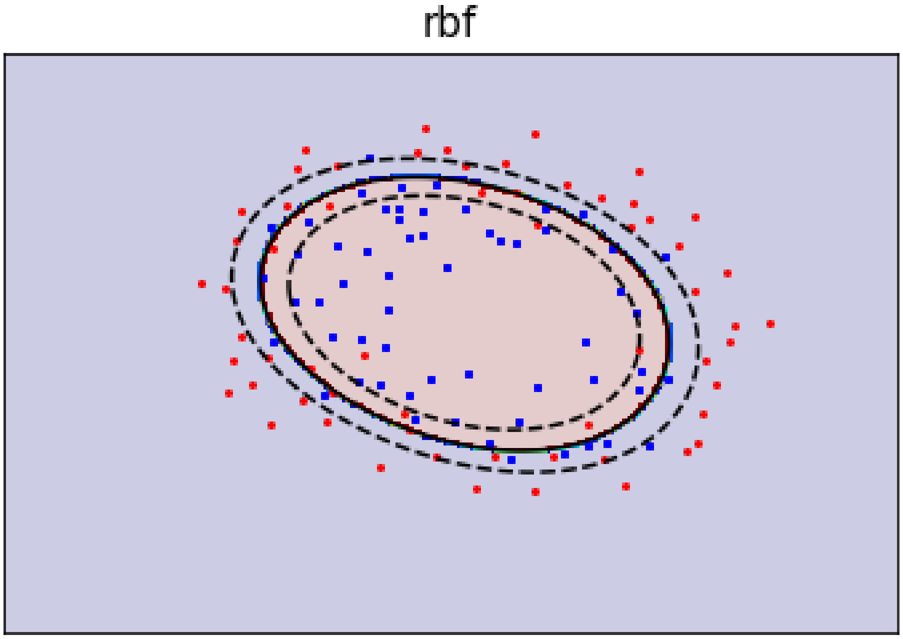
[20]:
```
fignum = 1

# fit the model
for kernel in ('linear','sigmoid', 'poly', 'rbf'):
    clf = svm.SVC(kernel=kernel, gamma=1, coef0 = 1)
    clf.fit(X, Y)
    with PdfPages(kernel + '3.pdf') as pdf:
        # plot the line, the points, and the nearest vectors to the plane
        fig, ax = plt.subplots()
        plt.figure(fignum, figsize=(5, 5))
        plt.clf()

        plt.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[:, 1],␣
 ↪s=80,
                    facecolors='None')
        plt.plot(X[mask][0].values, X[mask][1].values, 'bs', markersize = 2)
        plt.plot(X[~mask][0].values, X[~mask][1].values, 'ro', markersize = 2)

        plt.axis('tight')
        x_min = -1.5
        x_max = 1.5
        y_min = -1.5
        y_max = 1.5

        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.figure(fignum, figsize=(5, 5))
        CS = plt.contourf(XX, YY, np.sign(Z), 200, cmap='jet', alpha = .2)
        plt.contour(XX, YY, Z, colors=['k', 'k', 'k'], linestyles=['--', '-',␣
 ↪'--'], levels=[-.5, 0, .5])
        plt.title(kernel, fontsize = 15)
        plt.xlim(x_min, x_max)
        plt.ylim(y_min, y_max)

        plt.xticks(())
        plt.yticks(())
        fignum = fignum + 1
        pdf.savefig()
plt.show()
```


linear


sigmoid



poly



rbf

2 Example 2

2.0.1 Task 1

[21]:

```
import scipy.io as sio
```

[28]:

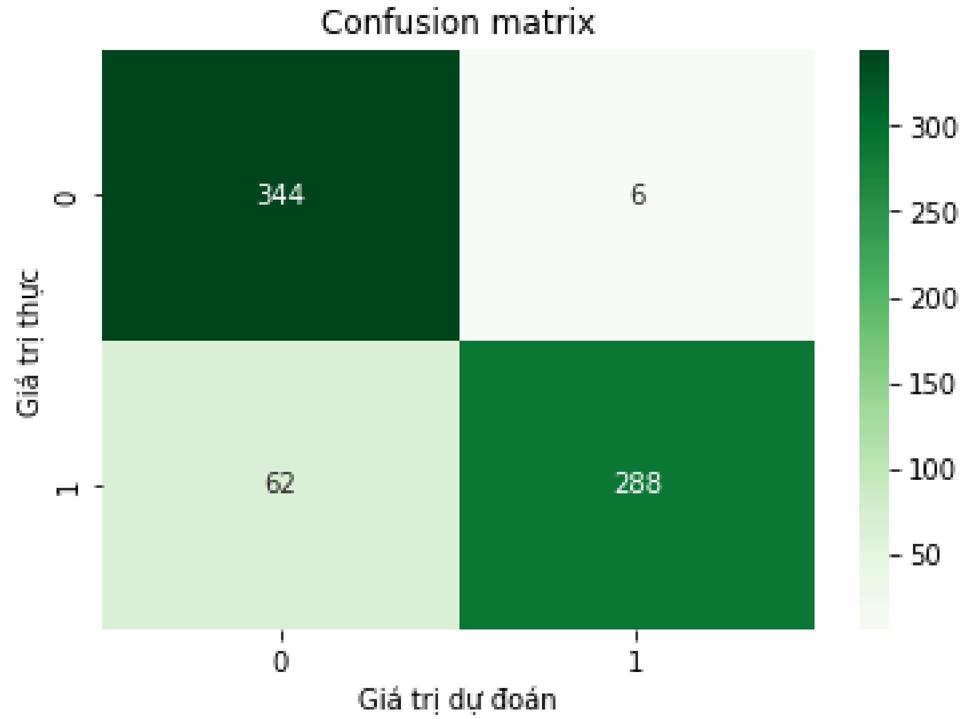
```
A = sio.loadmat('ARgender.mat')
X_train = A['Y_train'].T
X_test = A['Y_test'].T
print(X_train.shape)
N = 700
y_train = A['label_train'].reshape(N)
y_test = A['label_test'].reshape(N)
```

```
(700, 300)
```

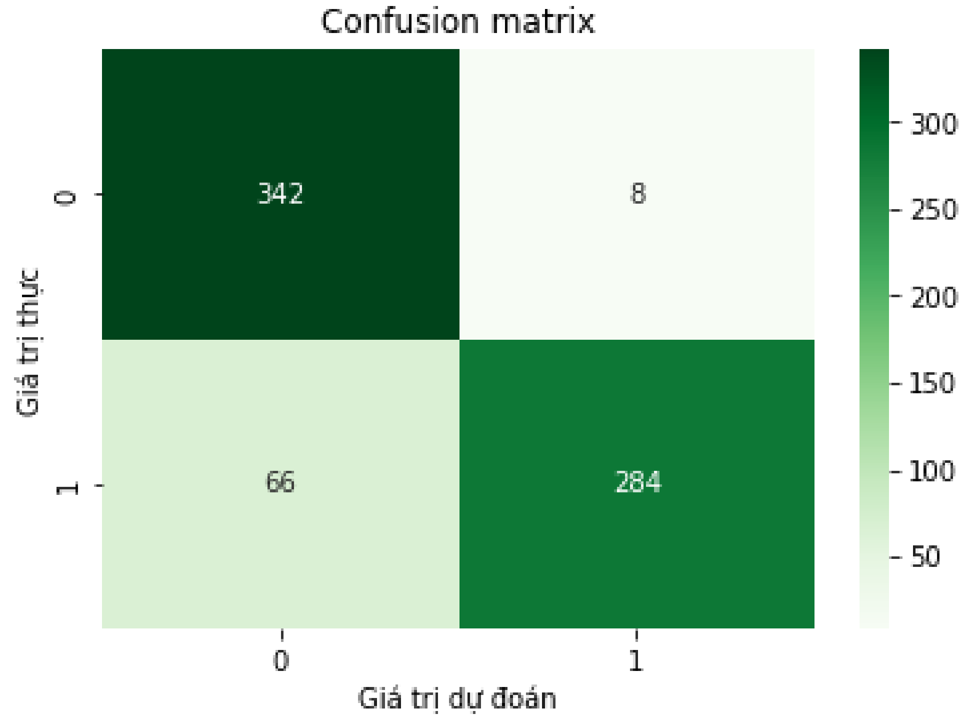
[48]:

```
fignum = 1
# fit the model
for kernel in ('linear','sigmoid', 'poly', 'rbf'):
    print ("Kernel is: ", kernel)
    clf = svm.SVC(kernel=kernel, gamma=1, coef0 = 1)
    clf.fit(X_train, y_train)
    svm_pred = clf.predict(X_test)
    cm_svm = confusion_matrix(y_test, svm_pred)
    ax = plt.subplot()
    sns.heatmap(cm_svm,annot=True, ax = ax, fmt='g', cmap='Greens')
    # labels, title and ticks
    ax.set_xlabel('Giá trị dự đoán')
    ax.set_ylabel('Giá trị thực')
    ax.set_title('Confusion matrix')
    plt.show()
    print ("Accuracy of " + kernel,  acr)
```

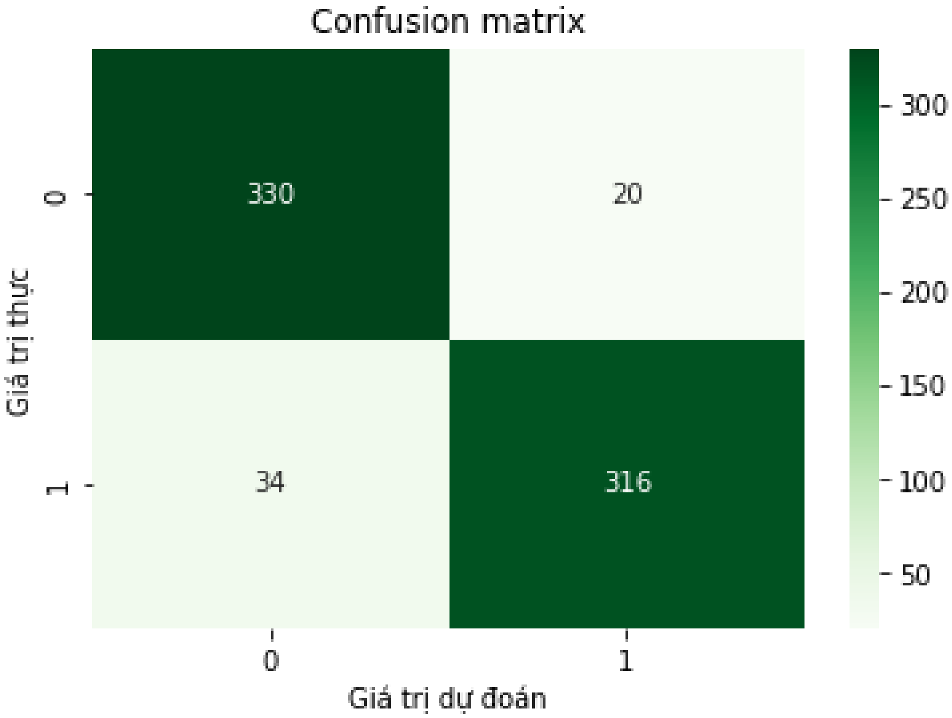
```
Kernel is:  linear
```



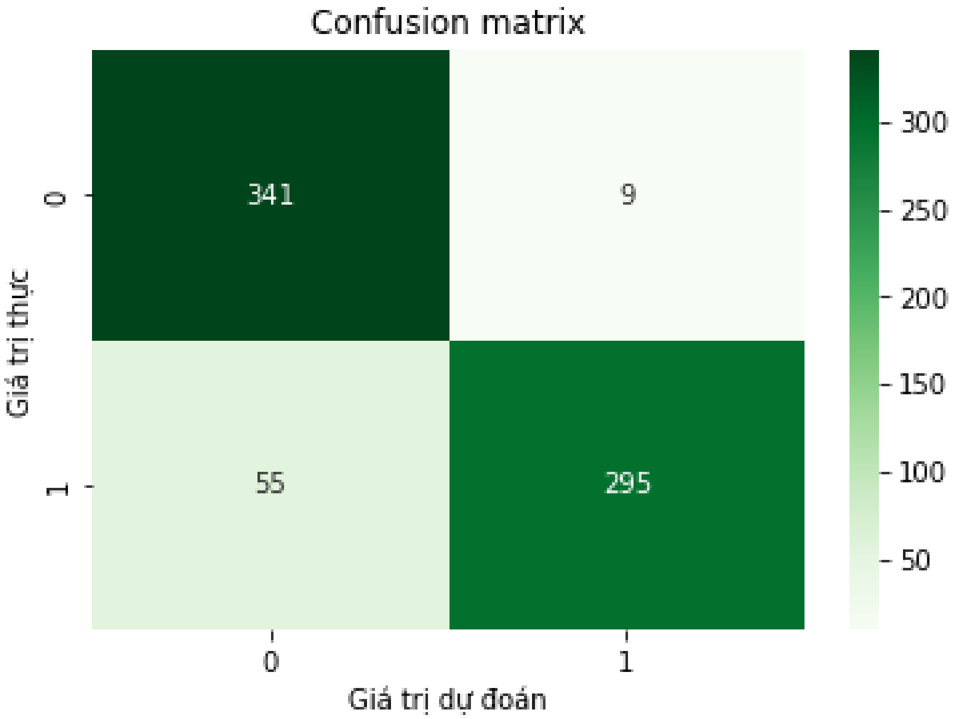
```
Accuracy of linear 0.9028571428571428
Kernel is:  sigmoid
```



```
Accuracy of sigmoid 0.8942857142857142
Kernel is:  poly
```



```
Accuracy of poly 0.9228571428571428
Kernel is:  rbf
```



```
Accuracy of rbf 0.9085714285714286
```

2.1 Task 2

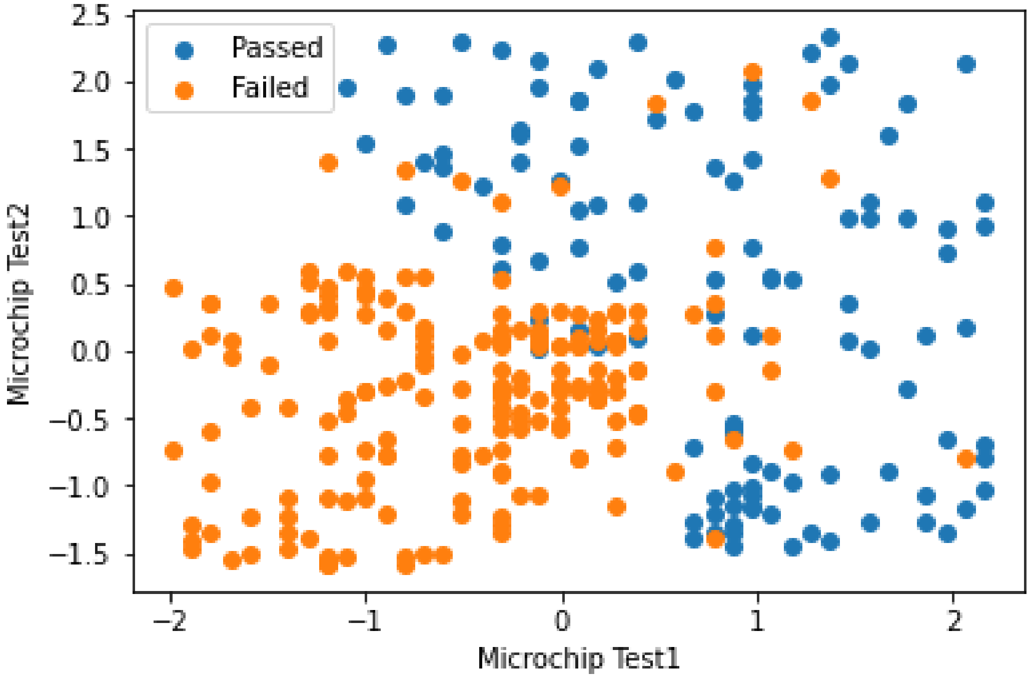
```
[49]: import numpy as np
      from sklearn.preprocessing import StandardScaler
```

```
[50]: dataset = pd.read_csv('dataset.csv')
      X = dataset.iloc[:, :-1].values
      y = dataset.iloc[:, -1].values
```

```
[52]: XTrain, XTest, yTrain, yTest = train_test_split(X, y, test_size = 0.25,␣
       ↪random_state = 0)
```

```
[53]: sc = StandardScaler()
      XTrain = sc.fit_transform(XTrain)
      XTest = sc.transform(XTest)
```

```
[62]: # visualizing data
      mask_train = yTrain == 1
      passed = plt.scatter(XTrain[mask_train][:, 0], XTrain[mask_train][:, 1])
      failed = plt.scatter(XTrain[~mask_train][:, 0], XTrain[~mask_train][:, 1])
      plt.xlabel('Microchip Test1')
      plt.ylabel('Microchip Test2')
      plt.legend((passed, failed), ('Passed', 'Failed'))
      plt.show()
```

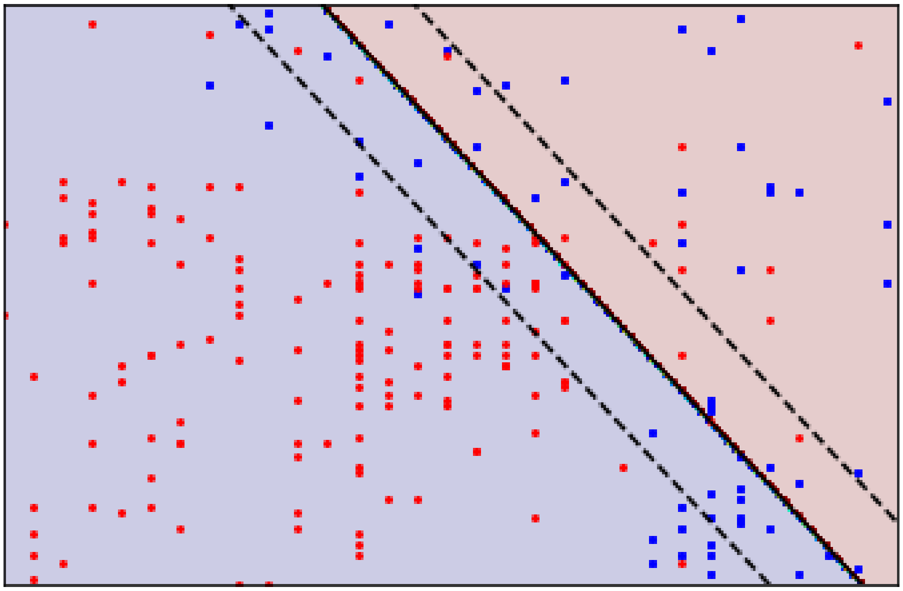


[66]:
```
fignum = 1

# fit the model
for kernel in ('linear','sigmoid', 'poly', 'rbf'):
    clf = svm.SVC(kernel=kernel, gamma=1, coef0 = 1)
    clf.fit(XTrain, yTrain)
    svm_pred = clf.predict(XTest)
    acr = accuracy_score(yTest, svm_pred)
    print ('Kernel: ', kernel)
    print ('acc:',   acr)
    with PdfPages(kernel + '4.pdf') as pdf:
        # plot the line, the points, and the nearest vectors to the plane
        fig, ax = plt.subplots()
        plt.figure(fignum, figsize=(5, 5))
        plt.clf()

        plt.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[:, 1],␣
 ↪s=80,
                    facecolors='None')
        plt.plot(XTrain[mask_train][:, 0], XTrain[mask_train][:, 1], 'bs',␣
 ↪markersize = 2)
```

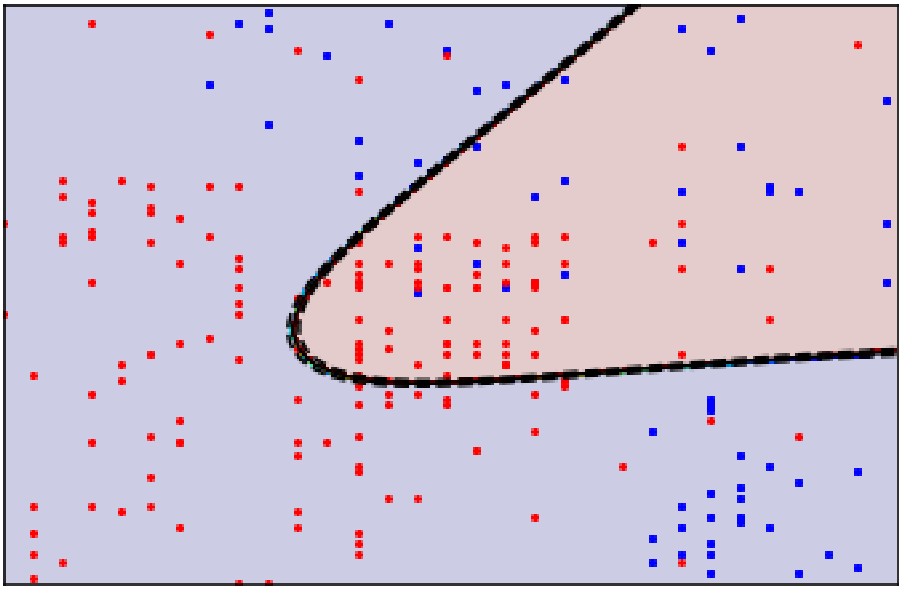
```
        plt.plot(XTrain[~mask_train][:, 0], XTrain[~mask_train][:, 1], 'ro', markersize = 2)

        plt.axis('tight')
        x_min = -1.5
        x_max = 1.5
        y_min = -1.5
        y_max = 1.5

        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.figure(fignum, figsize=(5, 5))
        CS = plt.contourf(XX, YY, np.sign(Z), 200, cmap='jet', alpha = .2)
        plt.contour(XX, YY, Z, colors=['k', 'k', 'k'], linestyles=['--', '-', '--'], levels=[-.5, 0, .5])
        plt.title(kernel, fontsize = 15)
        plt.xlim(x_min, x_max)
        plt.ylim(y_min, y_max)

        plt.xticks(())
        plt.yticks(())
        fignum = fignum + 1
        pdf.savefig()

plt.show()
```

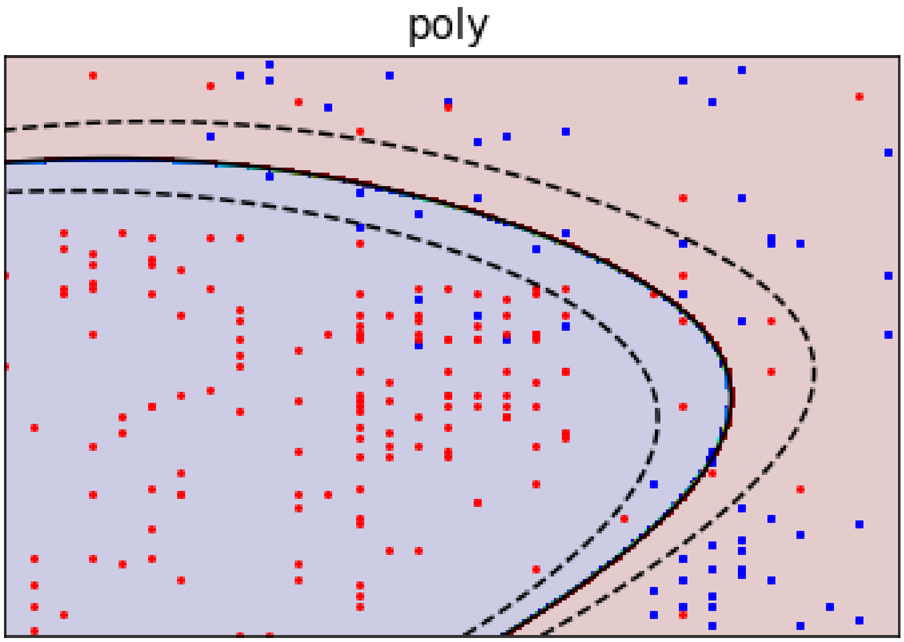
```
Kernel:  linear
acc: 0.9
Kernel:  sigmoid
acc: 0.7
Kernel:  poly
acc: 0.93
Kernel:  rbf
acc: 0.93
```

linear

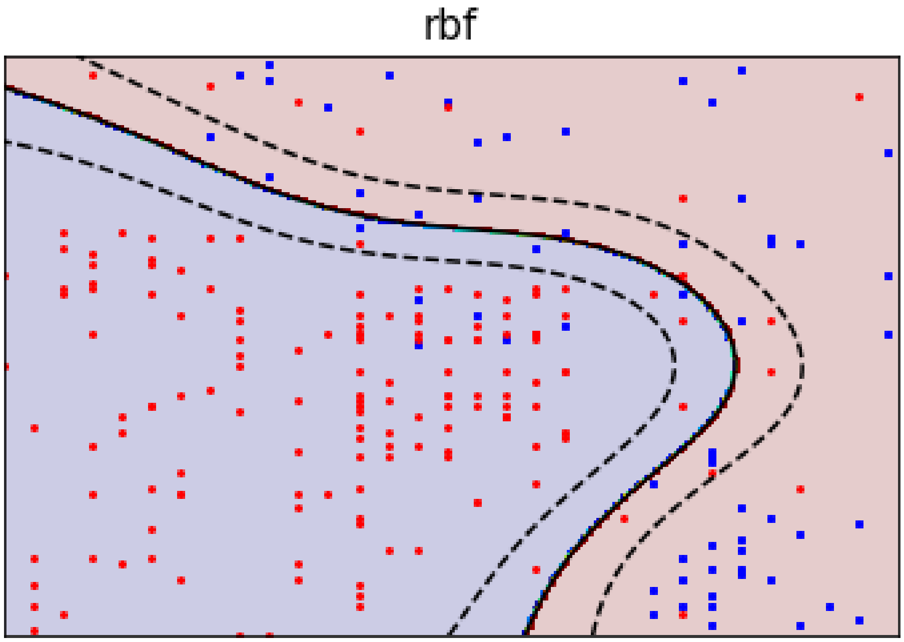


sigmoid





poly



rbf