```
import numpy as np
import random
from sklearn.svm import SVC

SVMbetter = 0
newcounter = 0
avgsupp = 0

for t in range(1000):

    n = 100
    w = np.array([0,0,0])

    p1x = random.uniform(-1,1) #pick 2 random points and find the equation of line going
    through them
    p1y = random.uniform(-1,1)
    p2x = random.uniform(-1,1)
    p2y = random.uniform(-1,1)
    f = np.array([(-(p2x-p1x)+p1x*(p2y-p1y)),(p1y-p2y),(p2x-p1x)]) #correct f in terms of (w0,
    wx, wy)
    points = np.zeros([n, 5])

    for i in range(n):
        points[i,0] = 1 #set all x0 to 1
        points[i,1] = random.uniform(-1,1) #set all x1 to the x-coordinate of point
        points[i,2] = random.uniform(-1,1) #set all x2 to the y-coordinate of point
        points[i,3] = np.sign(np.dot(f,[points[i,0],points[i,1],points[i,2]])) #correct
        classifaction

    if abs(sum(points[:,3])) == 100: #make sure not all points are on one side of line
        continue

    newcounter += 1

    X = points[:,:3]
    y = points[:,3]

    clf = SVC(float('Inf'), 'linear') #find the SVC
    clf.fit(X,y)

    clasf = []
    counter = 0

    while len(clasf) != n: #repeat PLA until all points are classified
        counter += 1
        clasf = []
        misc = []

        for i in range(n):
            points[i,4] = np.sign(np.dot(w,[points[i,0],points[i,1],points[i,2]])) #classify
            according to w
            if points[i,3] == points[i,4]: #set point as classified or misclassified
                clasf += [i]
```

```
            else:
                misc += [i]
        if len(misc) == 0:
            break
        w = w +
        points[misc[random.randint(0,len(misc)-1)],3]*points[misc[random.randint(0,len(misc)-1)],
        0:3]
        #update w with PLA

    SVMcorrect = 0
    PLAcorrect = 0

    for i in range(1000): #use a sample of 1000 points to determine the accuracy of w
        randx = random.uniform(-1,1)
        randy = random.uniform(-1,1)
        if np.sign(np.dot(f,[1,randx,randy])) == np.sign(clf.predict([[1,randx,randy]])[0]):
            SVMcorrect += 1
        if np.sign(np.dot(f,[1,randx,randy])) == np.sign(np.dot(w,[1,randx,randy])):
            PLAcorrect += 1

    if SVMcorrect > PLAcorrect: #compare accuracies of implementing SVM or not
        SVMbetter += 1

    avgsupp += len(clf.support_vectors_) #add up the number of support vectors used

print(SVMbetter/newcounter, avgsupp/newcounter)
```