|  |  |  |
| --- | --- | --- |
|  | به نام خدا |  |
| **دانشگاه تهران**  **دانشکده‌ مهندسی مکانیک**  **هوش مصنوعی**  **تمرین 3** | | |

|  |  |
| --- | --- |
| محمد مشرقی | نام و نام خانوادگی |
|  | شماره‌ دانشجویی |
|  | تاریخ ارسال گزارش |

Contents

[1-SVM(Linear) 3](#_Toc128010450)

[2- 5](#_Toc128010451)

[3- 9](#_Toc128010452)

[RBF 11](#_Toc128010453)

[Sigmoid 12](#_Toc128010454)

[Poly 13](#_Toc128010455)

# 1-SVM(Linear)

ابتدا با توزیع یکنواخت دو دسته داده درست می کنیم.

برای اولی داریم :

*MAX\_X1 = 1*

*MIN\_X1 = 0*

*MAX\_X2 = 1*

*MIN\_X2 = -1*

*SAMPLE = 100*

*x1 = uniform(MIN\_X1, MAX\_X1 ,SAMPLE)*

*x2 = uniform(MIN\_X2, MAX\_X2 ,SAMPLE)*

برای دومی داریم :

*MAX\_X1 = 3*

*MIN\_X1 = 2*

*MAX\_X2 = 4*

*MIN\_X2 = 3*

*SAMPLE = 100*

*x1 = uniform(MIN\_X1, MAX\_X1 ,SAMPLE)*

*x2 = uniform(MIN\_X2, MAX\_X2 ,SAMPLE)*

حال به دسته دو دسته برچسب می زنیم به دسته اول لیبل صفر و به دسته دوم لیبل یک:

for i in **range**(SAMPLE):

  Y.append(0)

for i in **range**(SAMPLE):

  Y.append(1)

Y =array(Y)

X =concatenate((X1,X2), axis=0)

حال سراغ دسته بندی می رویم:

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y,*

*test\_size = 0.25, random\_state = 0)*

بعد سراغ SVM رفته و svc را import می کنیم.  
و روی خطی تنظیم می کنیم : داریم:

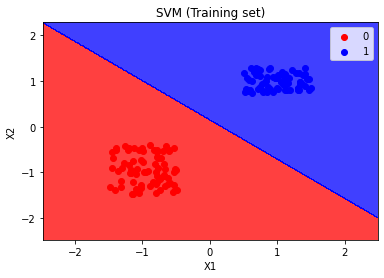
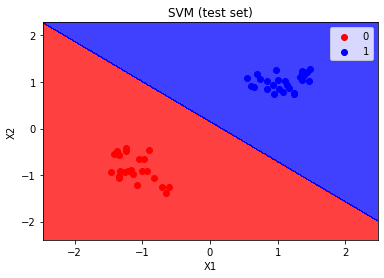
*classifier = SVC(kernel = 'linear', random\_state = 0)*

بعد از اموزش و دیدن نتایج ماتریس در هم ریختگی و ....... :

confusion matrix :

[[23 0]

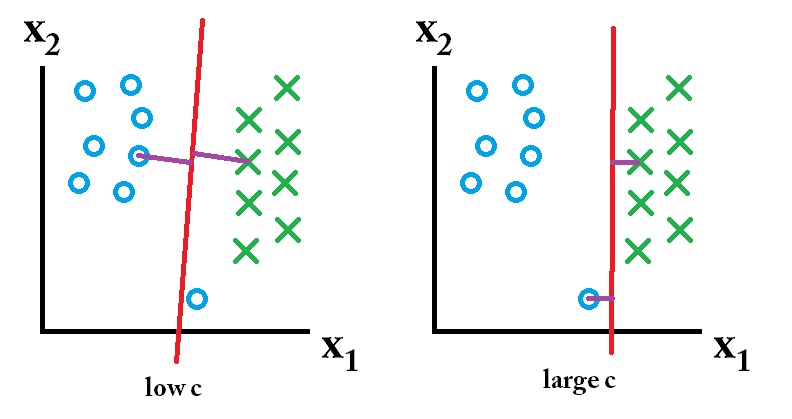
[ 0 27]]

 accuracy\_score = 1.000 precision\_score = 1.000 recall\_score = 1.000  
حال به سراغ نمودار ها می رویم:

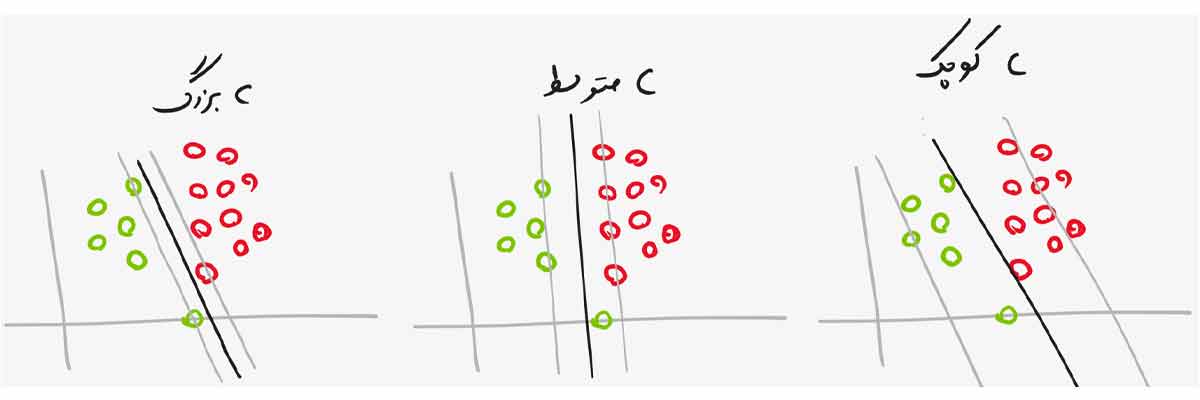
با توجه به نتایج تونستیم با کرنل خطی به دقت صد درصد برسیم (به دلیل فاصله اینکه دو دسته از هم دور بودن.)

# 2-

پارامتر C : یک **ترم ثابت مثبت** است که توسط **کاربر**تعیین می شود وحاشیه امن (Margin) را **کنترل**می کند. و نسبت آن عکس اندازه حاشیه امن است.



* **Cبزرگ**: C اگر مقدار بزرگی داشته باشد در نتیجه آن قیدهای مسئله بهنیه سازی SVM شدید می شوند و اگر خیلی خیلی بزرگ باشد، مسئله بهینه سازی شبیه Hard Margin می شود. یعنی خطای طبقه بندی باید صفر باشد!  
  که منظور همون overfit هستش.
* **C متوسط**: C اگر مقدار متوسط و حد معمولی داشته باشد در نتیجه آن شدت قیدهای مسئله بهنیه سازی SVM کمتر می شود و به مدل اجازه میدهد مقداری خطا داشته باشد (ولی خطای حداقل نه خیلی زیاد) و همین باعث میشود که مرز مناسبی برای مسئله طبقه بندی پیدا کند.
* **C کوچک**: C اگر مقدار کوچکی داشته باشد در نتیجه آن شدت قیدهای مسئله بهنیه سازی SVM خیلی کمتر می شود و به مدل اجازه میدهد خطای زیادی داشته باشد که در نتیجه آن ممکن است مدل به داده های پرت (outlier) حساس شود و در نتیجه مرز مناسبی بدست نیاید. مقدار C نباید خیلی کوچک باشد!



حال برای انتخاب بهترین اندازه C با استفاده از یک for و روش kfold و از بین سه تا معیار accuracy score و precision score و recall score یکی را طبق نیاز انتخاب می کنیم.  
حال یک For از 0.1 تا 50 می زنیم و بهترین مقدار C را با روش kfold و معیار accuracy score پیدا می کنیم.

cv = KFold(n\_splits=10, random\_state=1, shuffle=True)

def save\_data(mean\_scores,highest\_mean\_scores,best\_C):

    if(highest\_mean\_scores < mean\_scores):

      highest\_mean\_scores = mean\_scores

      best\_C = i

while(i < 50):

  classifier = SVC(C=i,kernel = 'linear', random\_state = 0)

  classifier.fit(X\_train, y\_train)

  scores = cross\_val\_score(classifier, X, Y, scoring='accuracy', cv=cv, n\_jobs=-1)

  print("C = %.1f" % i,'Accuracy: %.4f (the standard deviation : %.3f )' % (mean(scores), std(scores)))

  save\_data(mean(scores),highest\_mean\_scores,best\_C)

  i += 0.1

print("best valume of C = ",best\_C,"highest mean scores = ", highest\_mean\_scores)

نتایج:

C = 0.1 Accuracy: 0.9050 (the standard deviation : 0.079)

C = 0.2 Accuracy: 0.9050 (the standard deviation : 0.079)

C = 0.3 Accuracy: 0.9050 (the standard deviation : 0.079)

C = 0.4 Accuracy: 0.9200 (the standard deviation : 0.051)

C = 0.5 Accuracy: 0.9300 (the standard deviation : 0.051)

C = 0.6 Accuracy: 0.9300 (the standard deviation : 0.051)

C = 0.7 Accuracy: 0.9200 (the standard deviation : 0.040)

C = 0.8 Accuracy: 0.9200 (the standard deviation : 0.046)

C = 0.9 Accuracy: 0.9100 (the standard deviation : 0.049)

C = 1.0 Accuracy: 0.9200 (the standard deviation : 0.051)

C = 1.1 Accuracy: 0.9200 (the standard deviation : 0.051)

C = 1.2 Accuracy: 0.9350 (the standard deviation : 0.050)

C = 1.3 Accuracy: 0.9300 (the standard deviation : 0.051)

C = 1.4 Accuracy: 0.9300 (the standard deviation : 0.051)

C = 1.5 Accuracy: 0.9350 (the standard deviation : 0.055)

C = 1.6 Accuracy: 0.9350 (the standard deviation : 0.055)

C = 1.7 Accuracy: 0.9300 (the standard deviation : 0.056)

C = 1.8 Accuracy: 0.9400 (the standard deviation : 0.054)

C = 1.9 Accuracy: 0.9400 (the standard deviation : 0.054)

C = 2.0 Accuracy: 0.9400 (the standard deviation : 0.054)

C = 2.1 Accuracy: 0.9400 (the standard deviation : 0.054)

C = 2.2 Accuracy: 0.9350 (the standard deviation : 0.055)

C = 2.3 Accuracy: 0.9400 (the standard deviation : 0.054)

C = 2.4 Accuracy: 0.9400 (the standard deviation : 0.054)

C = 2.5 Accuracy: 0.9400 (the standard deviation : 0.054)

C = 2.6 Accuracy: 0.9450 (the standard deviation : 0.057)

C = 2.7 Accuracy: 0.9450 (the standard deviation : 0.057)

C = 2.8 Accuracy: 0.9450 (the standard deviation : 0.057)

C = 2.9 Accuracy: 0.9400 (the standard deviation : 0.054)

C = 3.0 Accuracy: 0.9400 (the standard deviation : 0.054)

C = 3.1 Accuracy: 0.9400 (the standard deviation : 0.054)

C = 3.2 Accuracy: 0.9400 (the standard deviation : 0.054)

C = 3.3 Accuracy: 0.9300 (the standard deviation : 0.060)

C = 3.4 Accuracy: 0.9300 (the standard deviation : 0.060)

C = 3.5 Accuracy: 0.9250 (the standard deviation : 0.060)

C = 3.6 Accuracy: 0.9250 (the standard deviation : 0.060)

C = 3.7 Accuracy: 0.9250 (the standard deviation : 0.060)

C = 3.8 Accuracy: 0.9250 (the standard deviation : 0.060)

C = 3.9 Accuracy: 0.9250 (the standard deviation : 0.060)

C = 4.0 Accuracy: 0.9250 (the standard deviation : 0.060)

C = 4.1 Accuracy: 0.9250 (the standard deviation : 0.060)

C = 4.2 Accuracy: 0.9300 (the standard deviation : 0.051)

C = 4.3 Accuracy: 0.9300 (the standard deviation : 0.051)

C = 4.4 Accuracy: 0.9300 (the standard deviation : 0.051)

C = 4.5 Accuracy: 0.9300 (the standard deviation : 0.051)

C = 4.6 Accuracy: 0.9300 (the standard deviation : 0.051)

C = 4.7 Accuracy: 0.9300 (the standard deviation : 0.051)

C = 4.8 Accuracy: 0.9300 (the standard deviation : 0.051)

C = 4.9 Accuracy: 0.9250 (the standard deviation : 0.051)

C = 5.0 Accuracy: 0.9250 (the standard deviation : 0.051)

بهترین مقدار C (با این داده های درست شده):

best valume of C = 2.6000000

highest mean scores = 0.9450000000000001

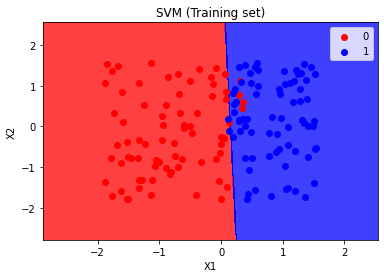
نمونه ای از train با c =2.6 (طبیعتا در اینجا نباید جواب accuracy score با highest mean score یکی شود چون دومی میانگینی از تعدادی accuracy score است.)

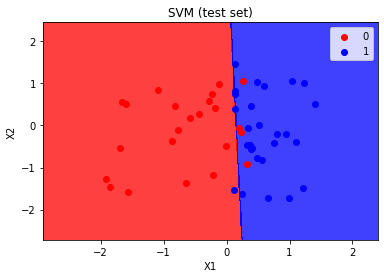
confusion matrix :

[[19 4]

[ 2 25]]

accuracy\_score = 0.880 precision\_score = 0.862 recall\_score = 0.926





# 3-

در اینجا ابتدا باید داده سازی کنیم

در این قسمت برای داده سازی اول ، محدوده داده سازی را بیشترین مقدار گفته شده در نظر می گیریم که در اینجا می شود بعد با یک while داده سازی می کنیم و تک تک داده ها را چک می کنیم که ایا در شرط صدق می کند یا نه اگر نکرد که هیچ ، اما اگر داده در شرط درست بود آن را ذخیره می کنیم.

برای دسته اول :

samples = 100

lower\_bound = -2\*\*0.5

upper\_bound = 2\*\*0.5

while count < samples:

    x1, x2 = np.random.uniform(lower\_bound, upper\_bound, 2)

    if 1 < x1\*\*2 + x2\*\*2 < 2:

        X1[count] = [x1, x2]

        count += 1

برای دسته دوم :

lower\_bound = -5\*\*0.5

upper\_bound = 5\*\*0.5

while count < samples:

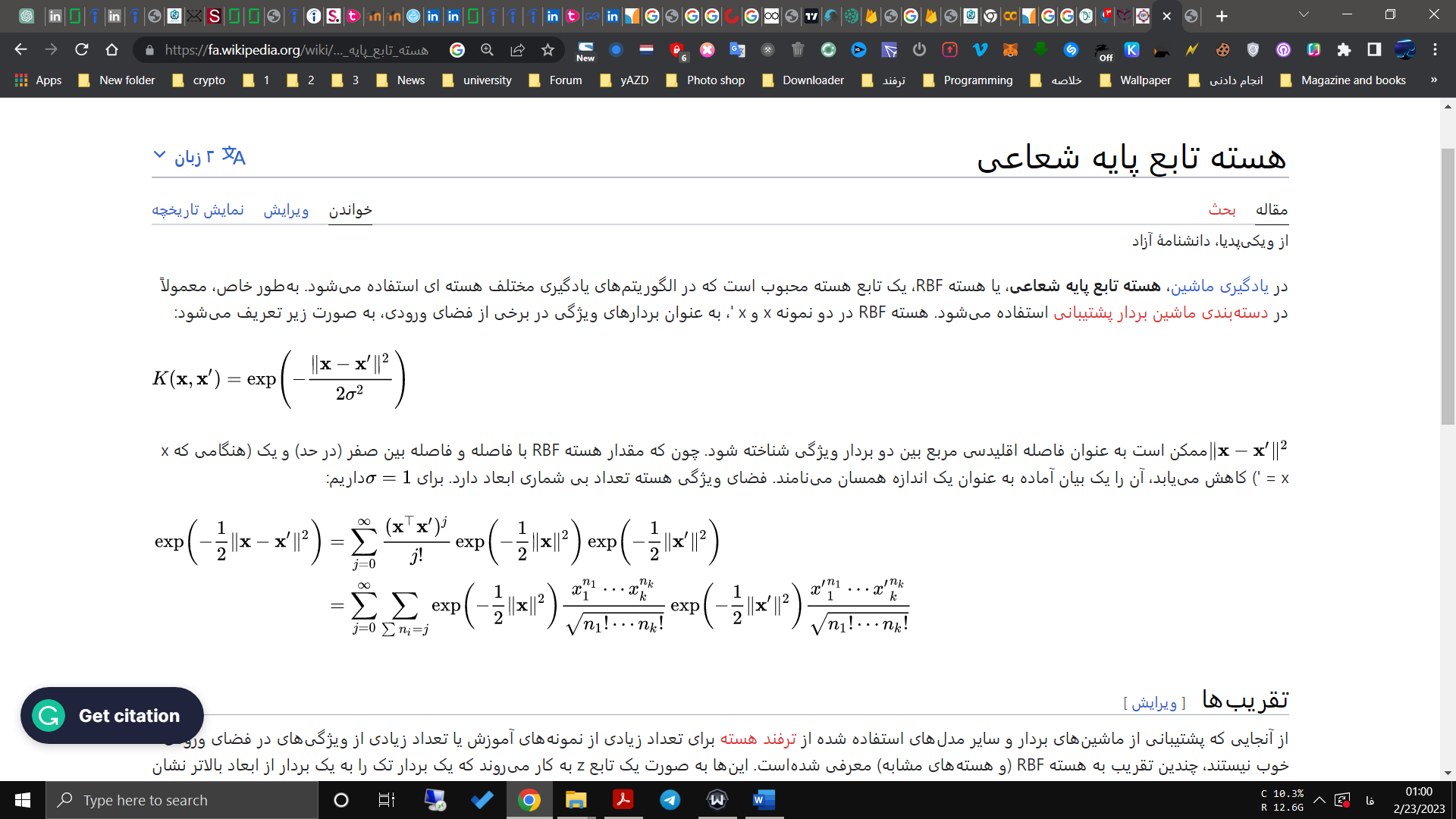
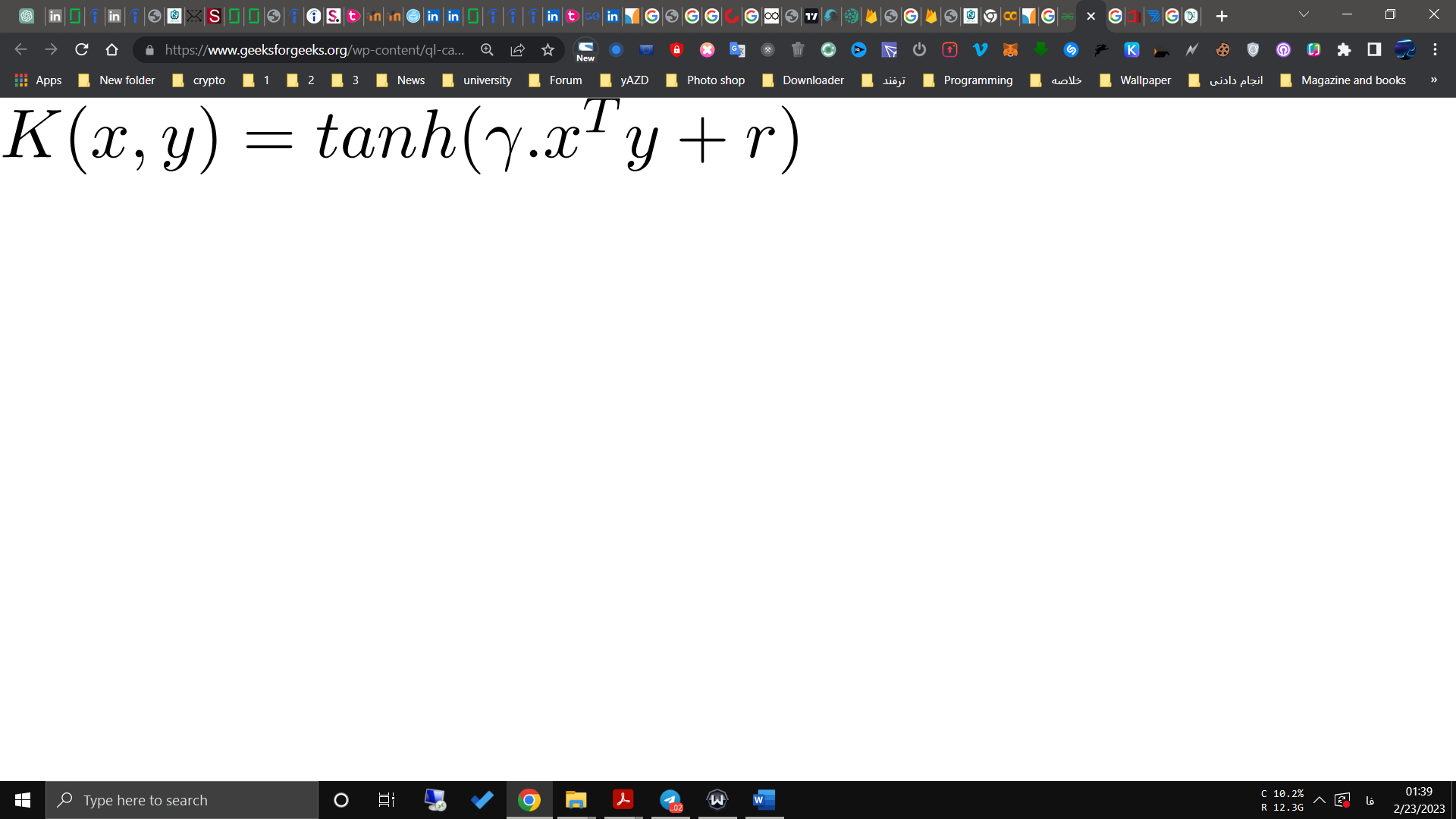
    x1, x2 = np.random.uniform(lower\_bound, upper\_bound, 2)

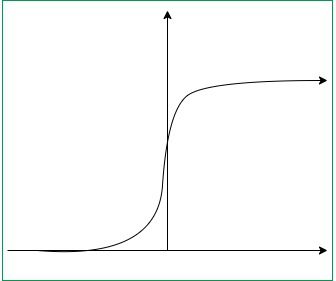
    if 4 < x1\*\*2 + x2\*\*2 < 5:

        X2[count] = [x1, x2]

        count += 1

حال به سراغ کرنل ها می رویم:

* RBF : محبوب ترین کرنل – با اکثر الگوریتم های ماشین لرنینگ سازگار – هزینه محاسباتیش زیاد است.
* Sigmoid : مثل صفر و یک عمل می کند و به عنوان فعال ساز در artificial neurons   
  استفاده می شود.



حال تست می کنیم در اینجا هم مثل قبل برا مقدار C یک فور می زنیم:

## RBF

C = 0.1 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 0.2 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 0.3 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 0.4 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 0.5 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 0.6 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 0.7 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 0.8 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 0.9 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 1.0 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 1.1 Accuracy: 1.0000 (the standard deviation : 0.000)

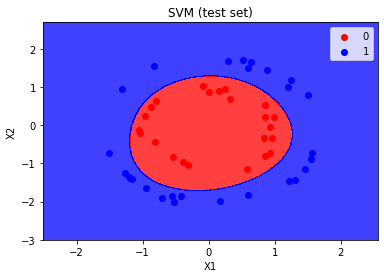
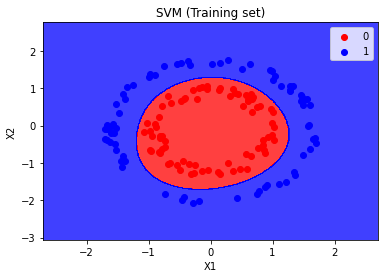
best valume of C = 0.1 highest mean scores = 1.0

با توجه به پراکندگی داده ها همیشه accuracy در این for برابر یک است.

confusion matrix :

[[23 0]

[ 0 27]]

 accuracy\_score = 1.000 precision\_score = 1.000 recall\_score = 1.000

## Sigmoid

با توجه به نتایج به درد این مادل کردن این داده نمی خورد و خطای زیادی دارد.

C = 0.1 Accuracy: 0.4100 (the standard deviation : 0.099)

C = 0.2 Accuracy: 0.4300 (the standard deviation : 0.147)

C = 0.3 Accuracy: 0.4350 (the standard deviation : 0.155)

C = 0.4 Accuracy: 0.4650 (the standard deviation : 0.110)

C = 0.5 Accuracy: 0.4600 (the standard deviation : 0.094)

C = 0.6 Accuracy: 0.5150 (the standard deviation : 0.071)

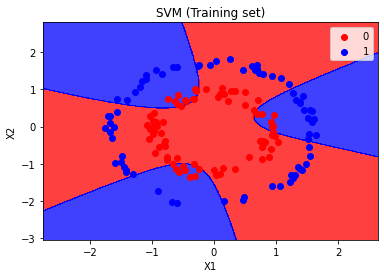
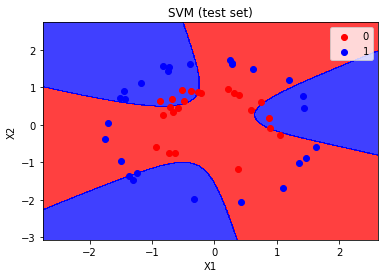
best valume of C = 0.7999999999999999 highest mean scores = 0.5650000000000001

برای بهترین مقدار یافت شده C :

confusion matrix :

[[15 8]

[13 14]]

 accuracy\_score = 0.580 precision\_score = 0.636 recall\_score = 0.519

## Poly

در اینجا C را بین 0.1 تا 10 و همزمان درجه poly را بین 2 تا 5 چک می کنیم . بهترین رو انتخاب می کنیم.

درجه دو :

C = 1.2 D = 2 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 1.3 D = 2 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 1.4 D = 2 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 1.5 D = 2 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 1.6 D = 2 Accuracy: 1.0000 (the standard deviation : 0.000)

درجه سه:

C = 1.5 D = 3 Accuracy: 0.7050 (the standard deviation : 0.115)

C = 1.6 D = 3 Accuracy: 0.7050 (the standard deviation : 0.115)

C = 1.7 D = 3 Accuracy: 0.7050 (the standard deviation : 0.115)

C = 1.8 D = 3 Accuracy: 0.7050 (the standard deviation : 0.115)

C = 1.9 D = 3 Accuracy: 0.7100 (the standard deviation : 0.114)

C = 2.0 D = 3 Accuracy: 0.7100 (the standard deviation : 0.114)

C = 2.1 D = 3 Accuracy: 0.7100 (the standard deviation : 0.114)

درجه چهار :

C = 0.6 D = 4 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 0.7 D = 4 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 0.8 D = 4 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 0.9 D = 4 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 1.0 D = 4 Accuracy: 1.0000 (the standard deviation : 0.000)

C = 1.1 D = 4 Accuracy: 1.0000 (the standard deviation : 0.000)

درجه پنج:

C = 0.1 D = 5 Accuracy: 0.6350 (the standard deviation : 0.180)

C = 0.2 D = 5 Accuracy: 0.7050 (the standard deviation : 0.096)

C = 0.3 D = 5 Accuracy: 0.7100 (the standard deviation : 0.102)

C = 0.4 D = 5 Accuracy: 0.6900 (the standard deviation : 0.116)

C = 0.5 D = 5 Accuracy: 0.6900 (the standard deviation : 0.116)

C = 0.6 D = 5 Accuracy: 0.6850 (the standard deviation : 0.125)

C = 0.7 D = 5 Accuracy: 0.6850 (the standard deviation : 0.125)

C = 0.8 D = 5 Accuracy: 0.6850 (the standard deviation : 0.125)

با توجه به نتایج برای درجه دو و چهار accuracy برابر یک می شود و چون نمی خواهیم محاسبات سنگین شود از درجه دو استفاده می کنیم:

best valume of C = 0.1 best valume of D = 2 highest mean scores = 1.0

داریم:

confusion matrix :

[[23 0]

[ 0 27]]

accuracy\_score = 1.000 precision\_score = 1.000 recall\_score = 1.000

