

a)  $\mu_1 = [3 \ 2]^T$   $\mu_2 = [5 \ 4]^T$   $P_1 = P_2$

$\Sigma_1 = \begin{bmatrix} 5 & \\ & 5 \end{bmatrix}$   $\Sigma_2 = \begin{bmatrix} 1 & \\ & 1 \end{bmatrix}$

1. ve 2. sınıflar için ayrı edili fonksiyonlar aşağıdaki gibidir.

$g_1(x) = -\frac{1}{2} (x - \mu_1)^T \Sigma_1^{-1} (x - \mu_1) - \frac{1}{2} \log |\Sigma_1| + \log P(w_1)$

$g_1(x) = -\frac{1}{2} [x-3 \ y-2] \begin{bmatrix} 2 & \\ & 2 \end{bmatrix} \begin{bmatrix} x-3 \\ y-2 \end{bmatrix} + \log \frac{1}{2} = -[(x-3)^2 + (y-2)^2] + \log \frac{1}{2}$

$g_2(x) = -\frac{1}{2} [x-5 \ y-4] \begin{bmatrix} 1 & \\ & 1 \end{bmatrix} \begin{bmatrix} x-5 \\ y-4 \end{bmatrix} + \log \frac{1}{2} = -\frac{1}{2} [(x-5)^2 + (y-4)^2] + \log \frac{1}{2}$

b)  $g_1(x) = g_2(x)$  için iki tarafta da bulunan " $\log \frac{1}{2}$ " değeri sadeleşir.

$x^2 - 6x + 9 + y^2 - 4y + 4 \stackrel{w_1}{\geq} 0.5x^2 - 5x + 12.5 + 0.5y^2 - 4y + 8$

$0.5x^2 + 0.5y^2 - x \stackrel{w_2}{\leq} 7.5$

Yeni gelen veri için  
 $\rightarrow 0.5x^2 + 0.5y^2 - x > 7.5$  ise 2. sınıfa  
 $\rightarrow 0.5x^2 + 0.5y^2 - x < 7.5$  ise 1. sınıfa düşer.

c) Soruda verilmiş olan varyans ve ortalama değerlerini bildiğimiz bir veri var. Her iki sınıfın da dağılım ve ortalamasını biliyoruz. Bize verilen ön verileri kullandık. Bundan dolayı bu bir öğretici sınıflandırıcıdır.

d) Lineerliği  $x^2$  ve  $y^2$  terimleri bozmaktadır. Karar sınırı 2 boyutta da farklı davranmaktadır. Lineer değildir.

e) İlk olarak kodumda kullanacağım kütüphaneleri, numpy ve pandas, import ettim.

```
import numpy as np
import pandas as pd
#Gerekli olan kutuphaneleri import ettik
```

Ardından soruda verilmiş olan değerleri değişkenlere atarak derledim.

```
#pandas kutuphanesini kullanarak tek satırda 2 mean değeri de tanımlandı
ortalamlar = pd.DataFrame({"mean1": [3,2], "mean2": [5,4]})

#numpy kutuphanesini kullanarak kovaryans matrislerini tanımladım
cov1 = np.array([[0.5,0],[0,0.5]])
cov2 = np.array([[1,0],[0,1]])
```

Index	mean1	mean2
0	3	5
1	2	4

cov1 - NumPy object array      cov2 - NumPy object array

	0	1
0	0.5	0
1	0	0.5

	0	1
0	1	0
1	0	1

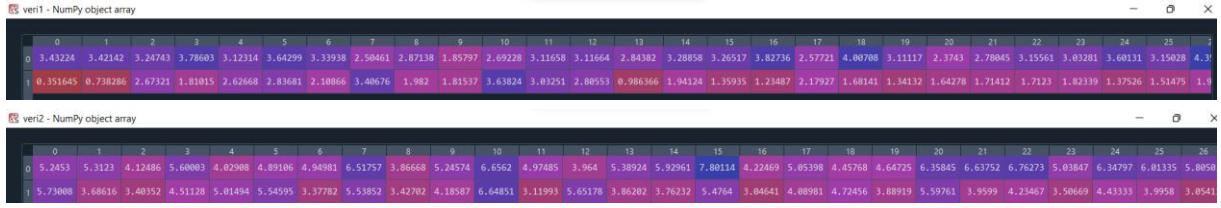
Elle hesapladığım karar eğrisi denklemi için bir fonksiyon tanımladım. Ardından if ve else komutlarıyla 1.sınıf ve 2.sınıf için karar durumları belirledim.

```
#Yazılı olarak bulduğum karar egrisi denklemini koda gecirdim
def karar_egrisi(x,y):
    denklem = 0.5*x**2 + 0.5*y**2 -x #Denklemin kendisi
    if(denklem > 7.5):
        return 2 #Bu durumda 2.sınıfa duser
    else:
        return 1 #Bu durumda 1.sınıfa duser
```

Ardından e şıkında bizden istenen şekilde 100er adet 2B öznitelik vektörü üretip b'de bulduğum karar eğrisini kullanarak hangi sınıftan olduğu belirledim. Ardından bu sonuçlara göre sınıflandırma performansını 2\*2'lik karıştırma matrisine aktardım. Gerekli olan işlemlerin fotoğrafları ve kodları aşağıda mevcuttur.

2B gauss dağılımını kullanarak 100er adet 2B öznitelik vektörü üretilmesi ve çıktısı

```
#Ortalamasi ve cov degeri belli olan normal dagilim seklinde 100er adet 2B oznitelik vektor uretildi.
veri1 = np.random.multivariate_normal(ortalamlar["mean1"], cov1,100).T
veri2 = np.random.multivariate_normal(ortalamlar["mean2"], cov2,100).T
```



veri1 - NumPy object array

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	3.43224	3.42142	3.24743	3.78603	3.12314	3.64299	3.33938	2.50461	2.87138	1.85797	2.69228	3.11658	3.11664	2.84382	3.28858	3.26517	3.82736	2.57721	4.80788	3.11117	2.3743	2.78845	3.15561	3.03281	3.68131	3.15828
1	0.351645	0.738286	2.67321	1.81015	2.62668	2.83681	2.10866	3.40676	1.982	1.81537	3.63824	3.03251	2.80553	0.986366	1.94124	1.35935	1.23487	2.17927	1.68141	1.34132	1.64278	1.71412	1.7123	1.82339	1.37526	1.51475

veri2 - NumPy object array


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	5.2453	5.3123	4.12486	5.68003	4.02988	4.89106	4.94981	6.51757	3.86668	5.24574	6.6562	4.97485	3.964	5.38924	5.92961	7.88114	4.22469	5.05398	4.45768	4.64725	6.35845	6.63752	6.76273	5.03847	6.34797	6.01335
1	5.73888	3.68616	3.40352	4.51128	5.01494	5.54595	3.37782	5.53952	3.42702	4.18587	6.64851	3.11993	5.65178	3.06202	3.76232	5.4764	3.04641	4.68981	4.72456	3.88919	5.59761	3.9599	4.23467	3.50669	4.43333	3.9958

Son olarak karıştırma matrisinin algoritmasını ve satırlarını ile sütunlarının isimlerini tanımlama

```
a,b,c,d = 0,0,0,0 #Ninovaya yuklenen kaynaktaki confusion matrisindeki a,b,c,d degerlerini tanımladım.
#Karıştırma matrisinin algoritmasını for dongusuyle kurdum
for i in range(100): #100 denemede sonuc1 ve sonuc2 için karar egrisi kontrol ediliyor. Toplamda 200 denem
    sonuc1 = karar_egrisi(veri1[0][i], veri1[1][i]) #a,b,c,d seklinde verilen degerler confusion matrisine
    if(sonuc1 == 1):
        a+=1
    else:
        b+=1
    sonuc2=karar_egrisi(veri2[0][i], veri2[1][i])
    if(sonuc2 == 2):
        d+=1
    else:
        c+=1
#Son olarak confusion matris indexlerine isim vererek özelleştiriyoruz.
conf_mat = pd.DataFrame(data={'Positive': [a,c], 'Negative': [b,d]},index=["True","False"])
```

Çıktı:

**KARIŞTIRMA MATRİSİ**

 conf\_mat - DataFrame

Index	Positive	Negative
True	94	6
False	4	96