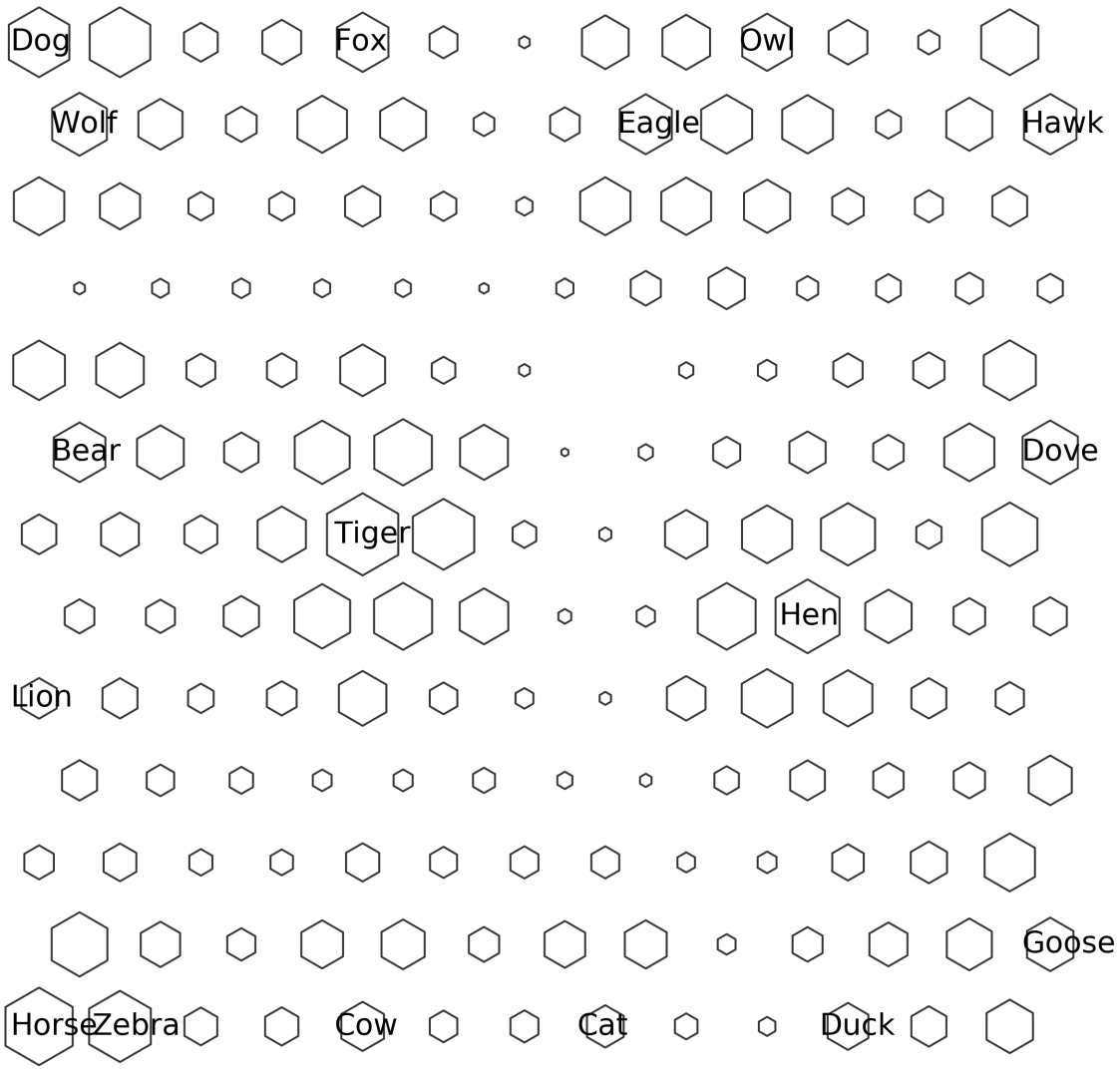
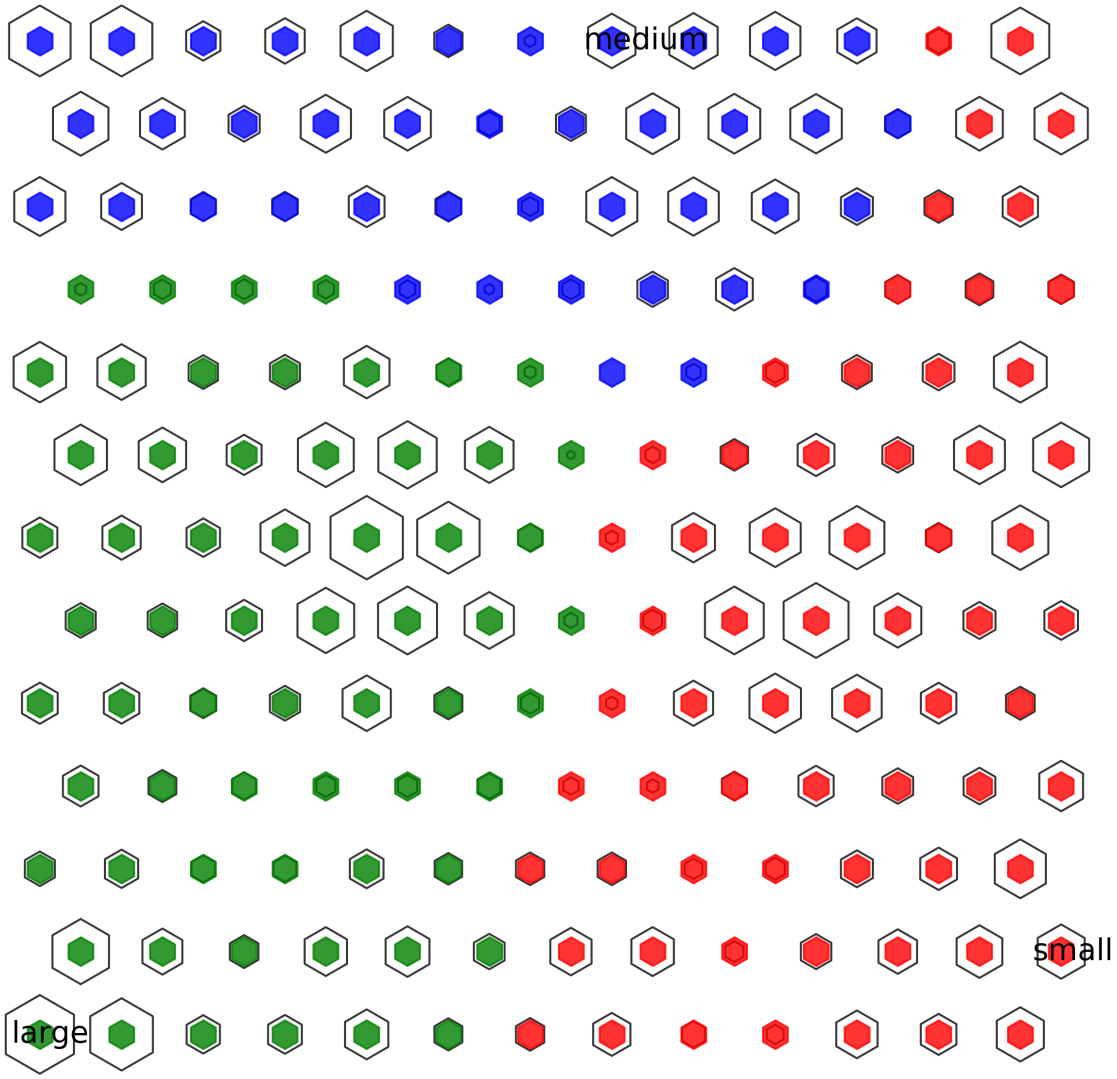
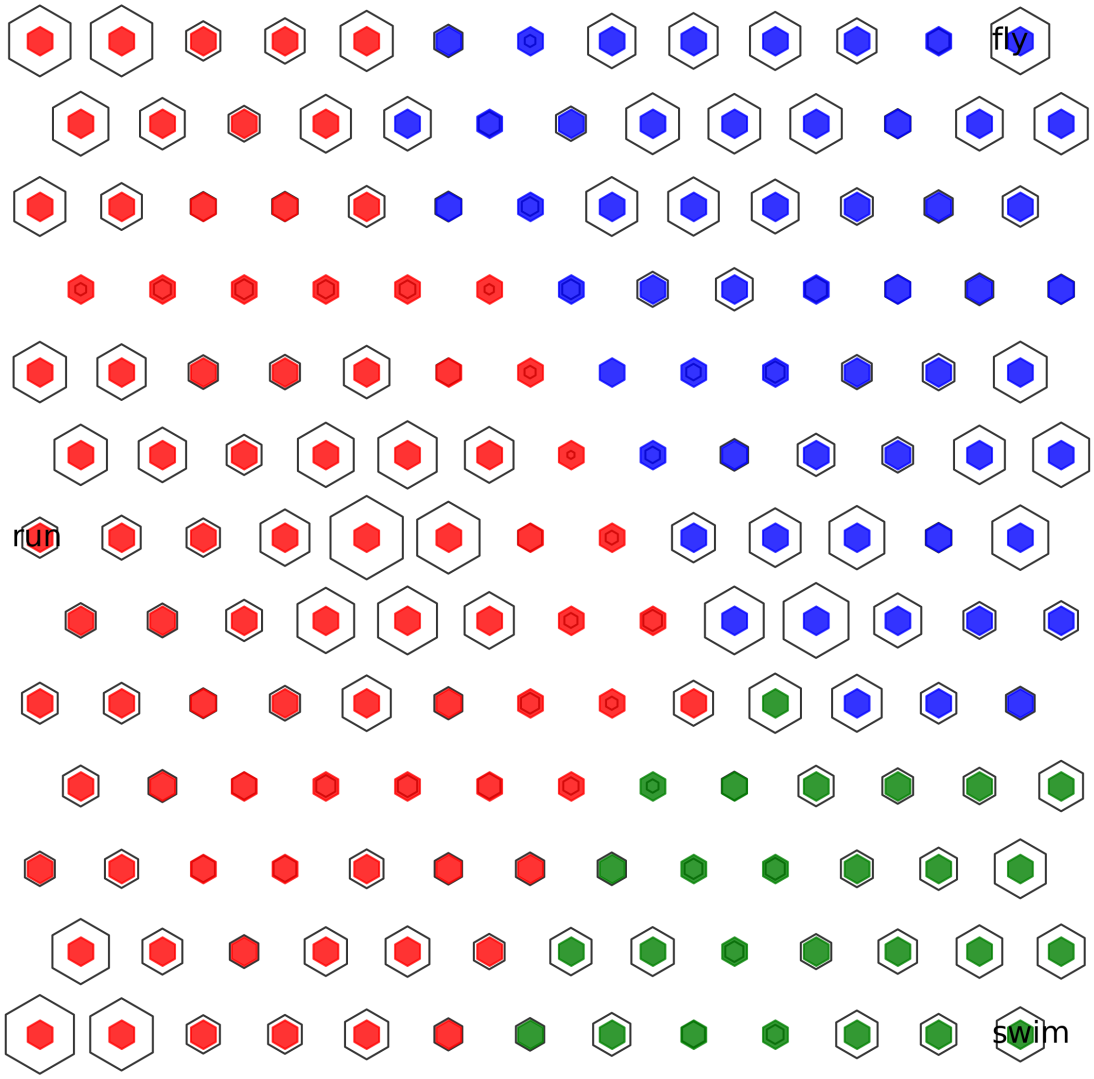
In [2]:
```
#P1
for name in dir():
    del globals()[name]

import numpy as np
import matplotlib.pyplot as plt
import os

#import from data folder
ad=os.getcwd()
ad=ad+'\data\\'
X=np.loadtxt(open(ad+"out.csv", "rb"), delimiter=",")

labels=np.array(['Dove', 'Hen', 'Duck', 'Goose', 'Owl', 'Hawk', 'Eagle',
                 'Fox', 'Dog', 'Wolf', 'Cat', 'Tiger', 'Lion', 'Horse',
                 'Zebra', 'Cow', 'Bear'])
Traits=np.array(['small', 'medium', 'large', '2 legs', '4 legs', 'hair',
                  'hooves','mane', 'feathers', 'hunt', 'run', 'fly', 'swim'])

MapDim=int(np.sqrt(X.shape[0]))

DMap=np.reshape(X[:,0],(MapDim,MapDim),order='F')
DMap=-DMap
DMap=DMap-np.min(DMap)
DMap=DMap/np.max(DMap)
DMap=20**DMap
DMap=DMap-np.min(DMap)
DMap=DMap/np.max(DMap)

X=X[:,1:]
k=np.argmax(X,axis=0)

#making the grid
nodes=np.linspace(0,MapDim-1,MapDim)
x1, y1 = np.meshgrid(nodes, nodes)
i=np.arange(1,MapDim,2)
x1[i,:]=x1[i,:]+0.5

#plotting
fig1=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(x1, y1, alpha=0.8,marker='h', facecolor='none', edgecolors='k', s=
1700*DMap)
for j in np.arange(len(labels)):
    c=np.array([k[j]//MapDim,k[j]%MapDim]) # getting the winner
    c=c*2/2
    if c[1]%2==1:
        c[0]=c[0]+ 0.5
    plt.text(c[0]-0.35,c[1]+0.1,labels[j], fontsize=15)
plt.gca().invert_yaxis()
plt.axis('off')
plt.show(fig1)
fig1.savefig('P1_1.svg',format='svg')

#_______
X=X[:,17:]
k=np.argmax(X,axis=0)
```

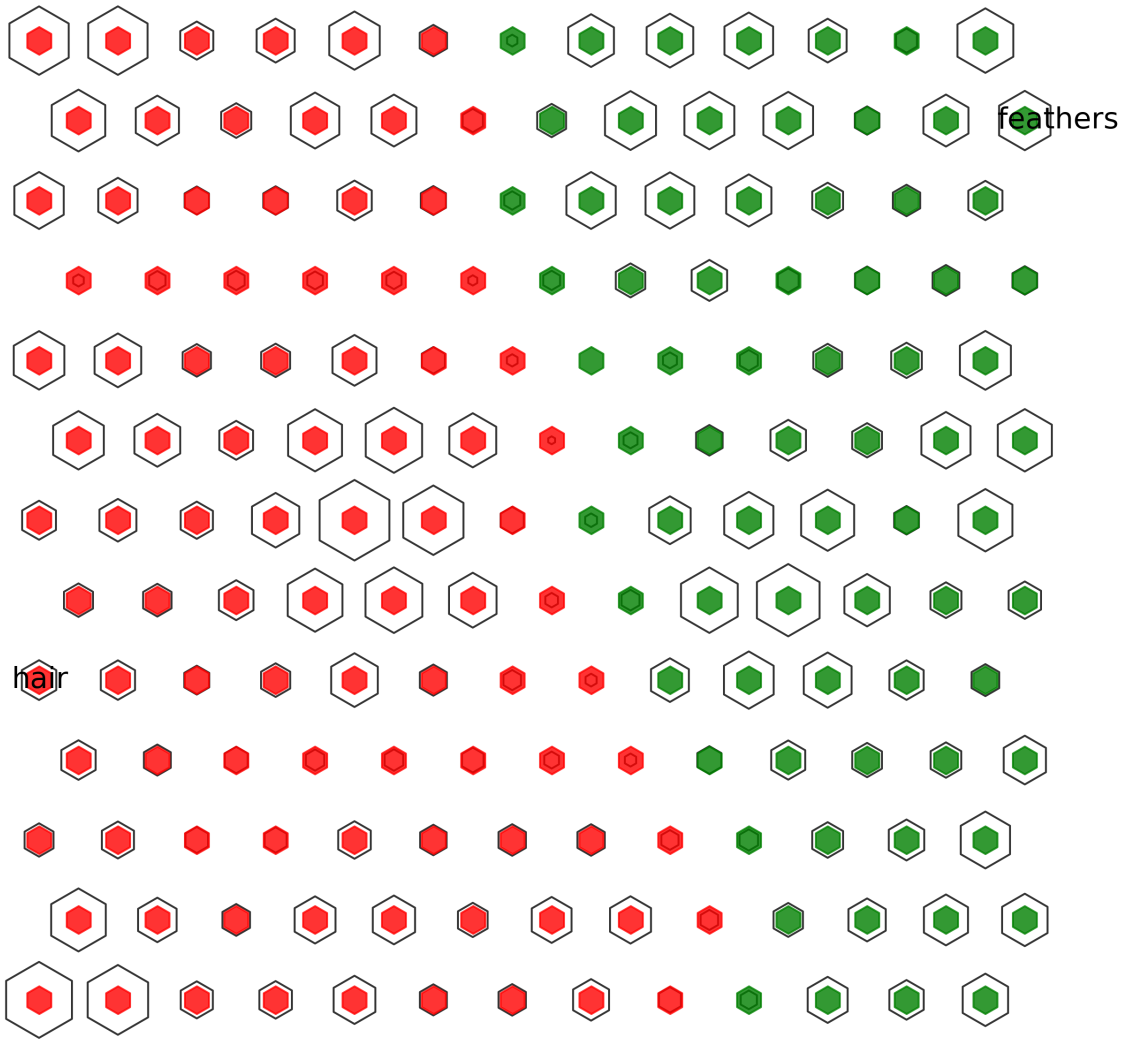
```
Map1=np.reshape(X[:,0],(MapDim,MapDim),order='F')-np.reshape(X[:,1],(MapDim,Ma
pDim),order='F')
Map2=np.reshape(X[:,0],(MapDim,MapDim),order='F')-np.reshape(X[:,2],(MapDim,Ma
pDim),order='F')
Map3=np.reshape(X[:,1],(MapDim,MapDim),order='F')-np.reshape(X[:,2],(MapDim,Ma
pDim),order='F')

fig2=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(x1, y1, alpha=0.8,marker='h', facecolor='none', edgecolors='k', s=
1700*DMap)
plt.scatter(x1[(Map1>=0) * (Map2>=0)], y1[(Map1>=0) * (Map2>=0)], alpha=0.8,ma
rker='h', c='r', s=200)
plt.scatter(x1[(Map2<=0) * (Map3<=0)], y1[(Map2<=0) * (Map3<=0)], alpha=0.8,ma
rker='h', c='g', s=200)
plt.scatter(x1[(Map1<0) * (Map3>0)], y1[(Map1<0) * (Map3>0)], alpha=0.8,marker
='h', c='b', s=200)
for j in np.arange(3):
    c=np.array([k[j]//MapDim,k[j]%MapDim])
    c=c*2/2
    if c[1]%2==1:
        c[0]=c[0]+ 0.5
    plt.text(c[0]-0.35,c[1]+0.1,Traits[j], fontsize=15)
plt.gca().invert_yaxis()
plt.axis('off')
plt.show(fig2)
# fig2.savefig('P1_2.svg',format='svg')

#_______

Map1=np.reshape(X[:,10],(MapDim,MapDim),order='F')-np.reshape(X[:,11],(MapDim,
MapDim),order='F')
Map2=np.reshape(X[:,10],(MapDim,MapDim),order='F')-np.reshape(X[:,12],(MapDim,
MapDim),order='F')
Map3=np.reshape(X[:,11],(MapDim,MapDim),order='F')-np.reshape(X[:,12],(MapDim,
MapDim),order='F')

fig3=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(x1, y1, alpha=0.8,marker='h', facecolor='none', edgecolors='k', s=
1700*DMap)
plt.scatter(x1[(Map1>=0) * (Map2>=0)], y1[(Map1>=0) * (Map2>=0)], alpha=0.8,ma
rker='h', c='r', s=200)
plt.scatter(x1[(Map2<=0) * (Map3<=0)], y1[(Map2<=0) * (Map3<=0)], alpha=0.8,ma
rker='h', c='g', s=200)
plt.scatter(x1[(Map1<0) * (Map3>0)], y1[(Map1<0) * (Map3>0)], alpha=0.8,marker
='h', c='b', s=200)
for j in np.array([10,11,12]):
    c=np.array([k[j]//MapDim,k[j]%MapDim])
    c=c*2/2
    if c[1]%2==1:
        c[0]=c[0]+ 0.5
    plt.text(c[0]-0.35,c[1]+0.1,Traits[j], fontsize=15)
plt.gca().invert_yaxis()
plt.axis('off')
plt.show(fig3)
# fig3.savefig('P1_3.svg',format='svg')
```

```
#_______

Map1=np.reshape(X[:,5],(MapDim,MapDim),order='F')-np.reshape(X[:,8],(MapDim,Ma
pDim),order='F')

fig4=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(x1, y1, alpha=0.8,marker='h', facecolor='none', edgecolors='k', s=
1700*DMap)
plt.scatter(x1[Map1>=0], y1[Map1>=0], alpha=0.8,marker='h', c='r', s=200)
plt.scatter(x1[Map1<0], y1[Map1<0], alpha=0.8,marker='h', c='g', s=200)
for j in np.array([5,8]):
    c=np.array([k[j]//MapDim,k[j]%MapDim])
    c=c*2/2
    if c[1]%2==1:
        c[0]=c[0]+ 0.5
    plt.text(c[0]-0.35,c[1]+0.1,Traits[j], fontsize=15)
plt.gca().invert_yaxis()
plt.axis('off')
plt.show(fig4)
# fig4.savefig('P1_4.svg',format='svg')
```


Dog
Fox
Owl
Wolf
Eagle
Hawk
Bear
Dove
Tiger
Hen
Lion
Goose
Horse
Zebra
Cow
Cat
Duck

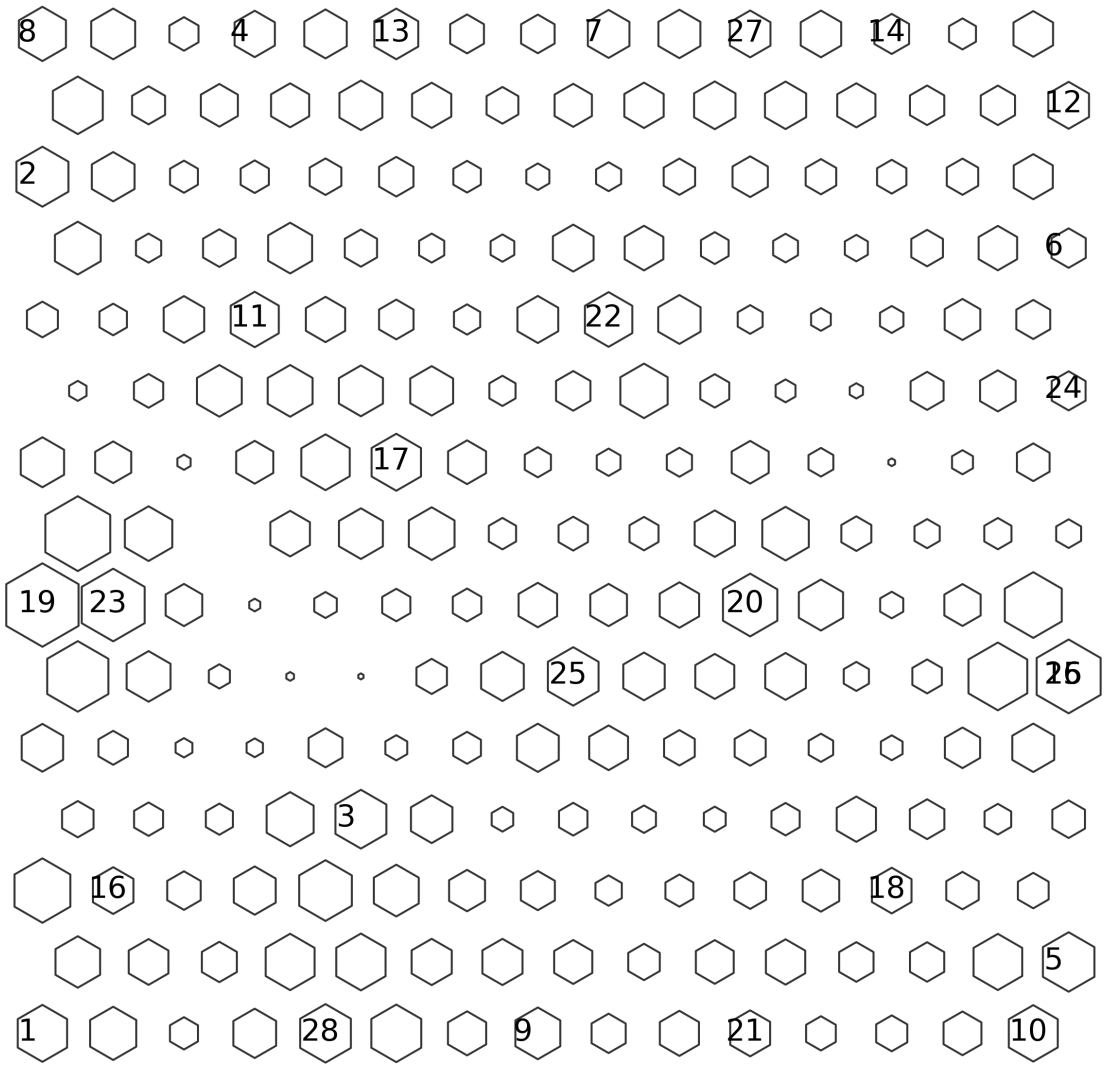

medium
small
large

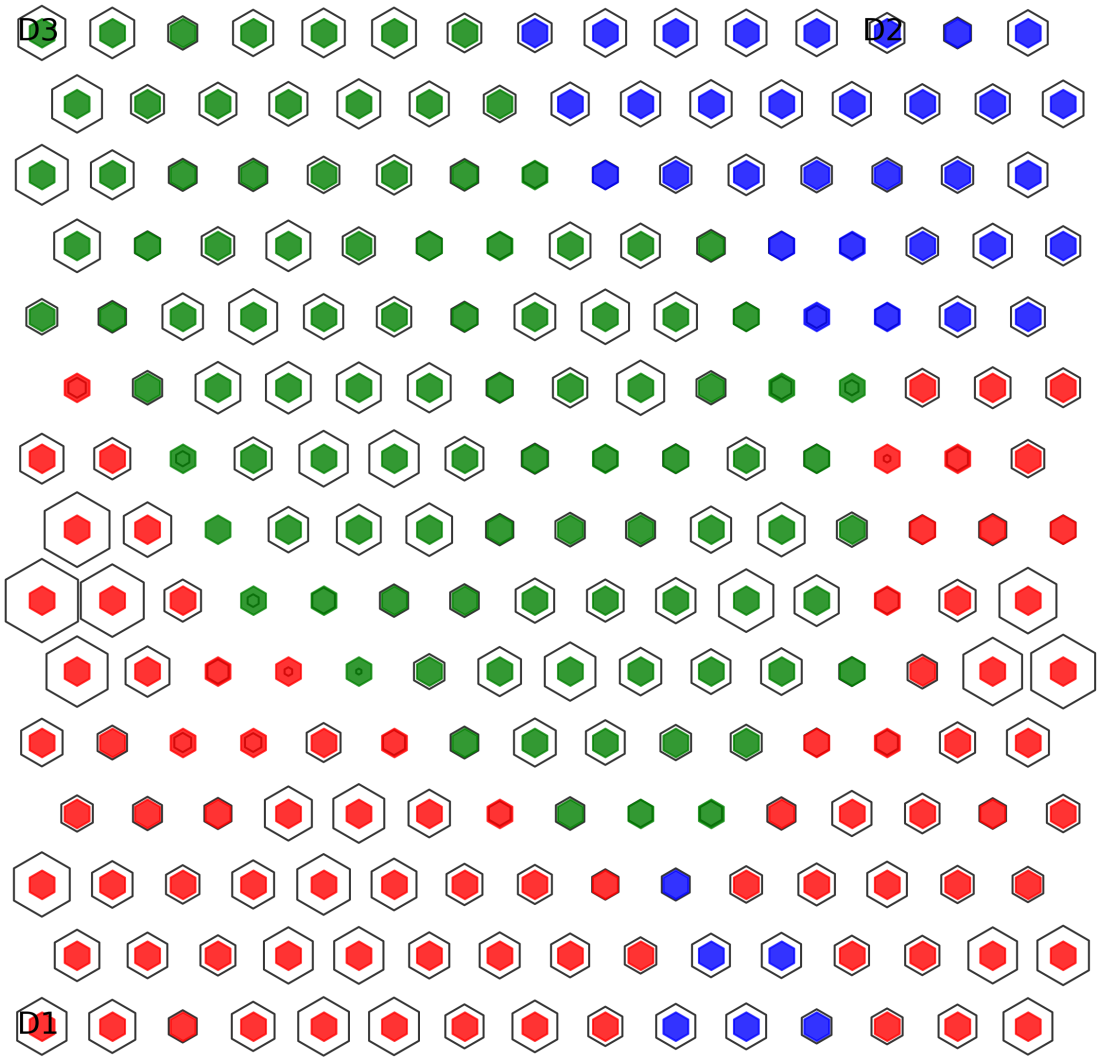

fly
run
swim

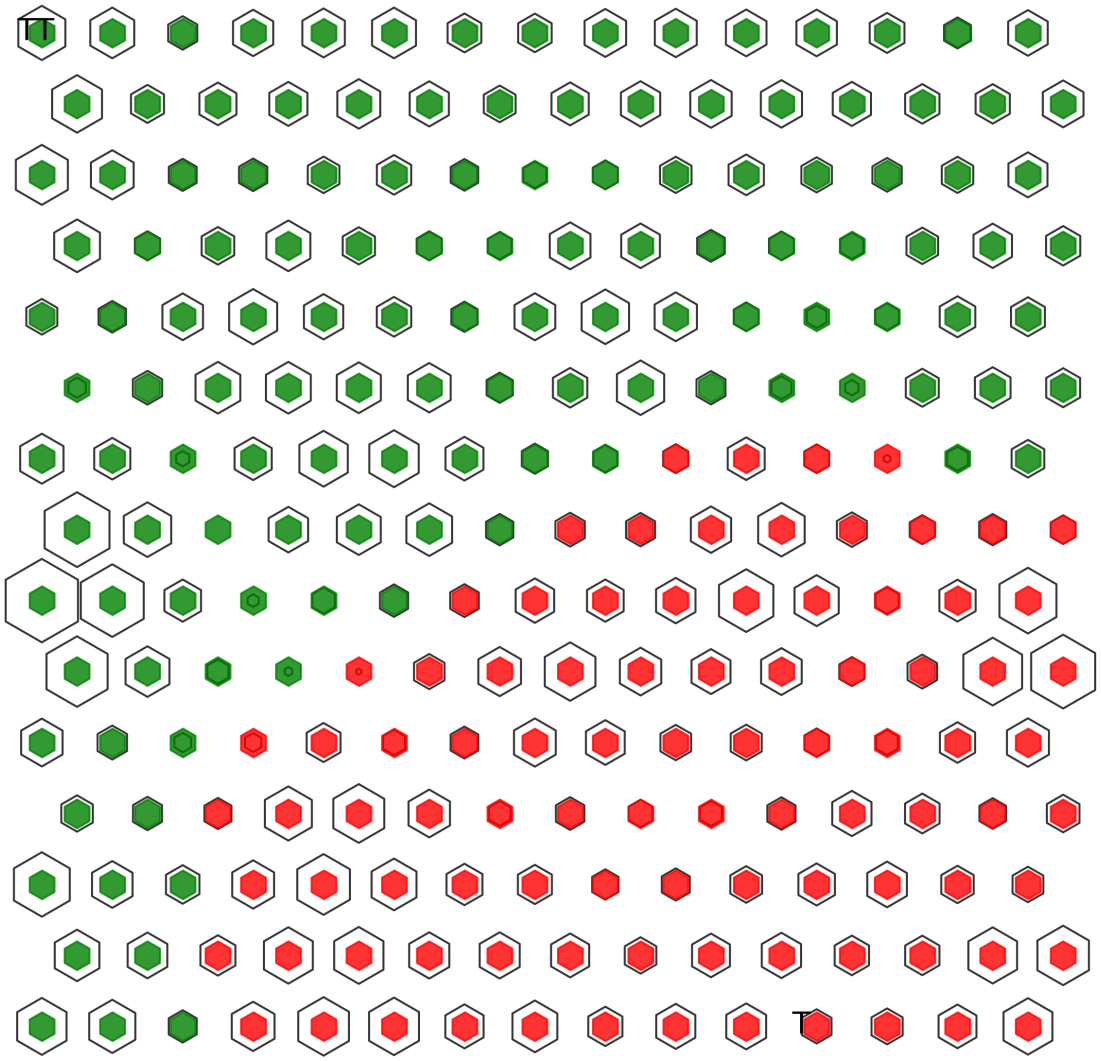

feathers
hair

In [3]:
```
#P2
for name in dir():
    del globals()[name]

import numpy as np
import matplotlib.pyplot as plt
import os

#import from data folder
ad=os.getcwd()
ad=ad+'\data\\'
X=np.loadtxt(open(ad+"out2.csv", "rb"), delimiter=",")

labels=np.array(['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', '26', '27', '28'])

Traits=np.array(['D1', 'D2', 'D3', 'T', 'TT', 'Years', 'AOE', 'Productivity',
'P19', 'P17', 'Grant', 'Writes'])

MapDim=int(np.sqrt(X.shape[0]))

DMap=np.reshape(X[:,0],(MapDim,MapDim),order='F')
DMap=-DMap
DMap=DMap-np.min(DMap)
DMap=DMap/np.max(DMap)
DMap=10**DMap
DMap=DMap-np.min(DMap)
DMap=DMap/np.max(DMap)

X=X[:,1:]
k=np.argmax(X,axis=0)

#making the grid
nodes=np.linspace(0,MapDim-1,MapDim)
x1, y1 = np.meshgrid(nodes, nodes)
i=np.arange(1,MapDim,2)
x1[i,:]=x1[i,:]+0.5

#plotting
fig1=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(x1, y1, alpha=0.8,marker='h', facecolor='none', edgecolors='k', s=
1700*DMap)
for j in np.arange(len(labels)):
    c=np.array([k[j]//MapDim,k[j]%MapDim]) # getting the winner
    c=c*2/2
    if c[1]%2==1:
        c[0]=c[0]+ 0.5
    plt.text(c[0]-0.35,c[1]+0.1,labels[j], fontsize=15)
plt.gca().invert_yaxis()
plt.axis('off')
plt.show(fig1)
# fig1.savefig('P2_1.svg',format='svg')
```

```
#_______
X=X[:,28:]
k=np.argmax(X,axis=0)

Map1=np.reshape(X[:,0],(MapDim,MapDim),order='F')-np.reshape(X[:,1],(MapDim,Ma
pDim),order='F')
Map2=np.reshape(X[:,0],(MapDim,MapDim),order='F')-np.reshape(X[:,2],(MapDim,Ma
pDim),order='F')
Map3=np.reshape(X[:,1],(MapDim,MapDim),order='F')-np.reshape(X[:,2],(MapDim,Ma
pDim),order='F')

fig2=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(x1, y1, alpha=0.8,marker='h', facecolor='none', edgecolors='k', s=
1700*DMap)
plt.scatter(x1[(Map1>=0) * (Map2>=0)], y1[(Map1>=0) * (Map2>=0)], alpha=0.8,ma
rker='h', c='r', s=200)
plt.scatter(x1[(Map2<=0) * (Map3<=0)], y1[(Map2<=0) * (Map3<=0)], alpha=0.8,ma
rker='h', c='g', s=200)
plt.scatter(x1[(Map1<0) * (Map3>0)], y1[(Map1<0) * (Map3>0)], alpha=0.8,marker
='h', c='b', s=200)
for j in np.arange(3):
    c=np.array([k[j]//MapDim,k[j]%MapDim])
    c=c*2/2
    if c[1]%2==1:
        c[0]=c[0]+ 0.5
    plt.text(c[0]-0.35,c[1]+0.1,Traits[j], fontsize=15)
plt.gca().invert_yaxis()
plt.axis('off')
plt.show(fig2)
# fig2.savefig('P2_2.svg',format='svg')

#_______

Map1=np.reshape(X[:,3],(MapDim,MapDim),order='F')-np.reshape(X[:,4],(MapDim,Ma
pDim),order='F')

fig3=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(x1, y1, alpha=0.8,marker='h', facecolor='none', edgecolors='k', s=
1700*DMap)
plt.scatter(x1[Map1>=0], y1[Map1>=0], alpha=0.8,marker='h', c='r', s=200)
plt.scatter(x1[Map1<0], y1[Map1<0], alpha=0.8,marker='h', c='g', s=200)
for j in np.array([3,4]):
    c=np.array([k[j]//MapDim,k[j]%MapDim])
    c=c*2/2
    if c[1]%2==1:
        c[0]=c[0]+ 0.5
    plt.text(c[0]-0.35,c[1]+0.1,Traits[j], fontsize=15)
plt.gca().invert_yaxis()
plt.axis('off')
plt.show(fig3)
# fig3.savefig('P2_3.svg',format='svg')
```


8
4
13
7
27
14
12
2
6
11
22
24
17
19
23
20
25
26
3
16
18
5
1
28
9
21
10

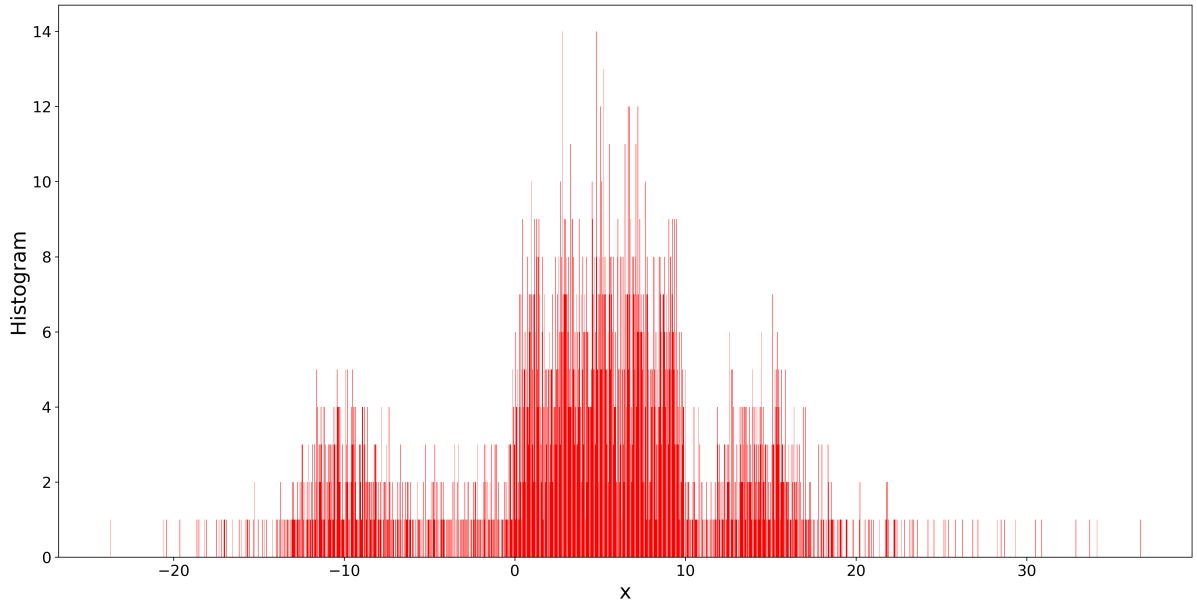

D3
D2
D1

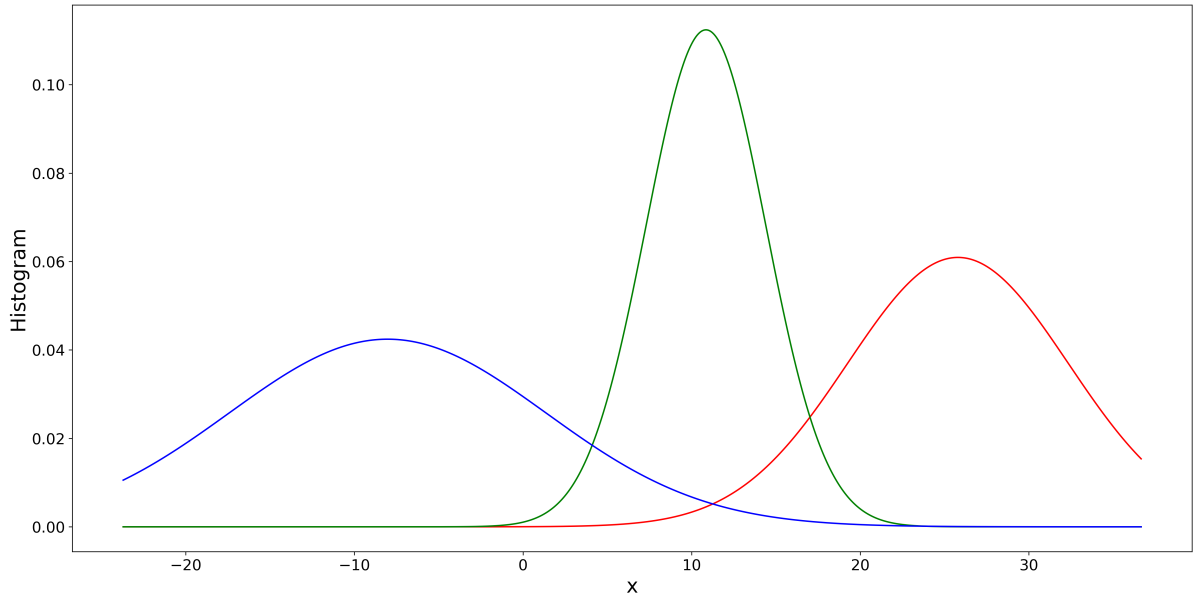


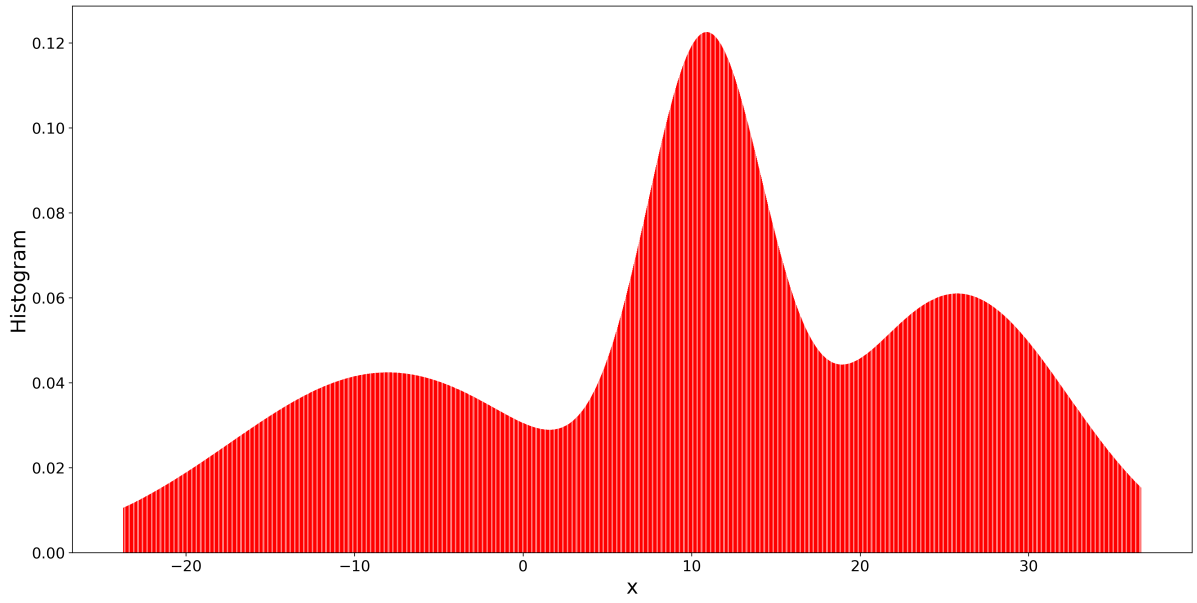
In [ ]:
```
#P3_a
for name in dir():
    del globals()[name]

import numpy as np
import matplotlib.pyplot as plt
import os

def Histogram(X,N):
    b=np.linspace(np.min(X),np.max(X),num=N)
    h=np.zeros(N)
    for m in np.arange(len(X)):
        e=np.abs(b-X[m])
        e=np.where(e==np.min(e),1,0)
        h+=e
    return h,b
```

In [7]:
```
#P3_b
for name in dir():
    del globals()[name]

import numpy as np
import matplotlib.pyplot as plt
import os

def Histogram(X,N):
    b=np.linspace(np.min(X),np.max(X),num=N)
    h=np.zeros(N)
    for m in np.arange(len(X)):
        e=np.abs(b-X[m])
        e=np.where(e==np.min(e),1,0)
        h+=e
    return h,b

#import from data folder
ad=os.getcwd()
ad=ad+'\data\\'
X=np.loadtxt(open(ad+"DatasetA.csv", "rb"), delimiter=",")

Nb=2000
data=Histogram(X,Nb)
h=data[0]
x=data[1]

fig1=plt.figure(figsize=[20,10],dpi=300)
plt.bar(x,h,width=50/Nb,facecolor='r',edgecolor=None,linewidth=0.5)
plt.ylabel('Histogram',fontsize=20)
plt.xlabel('x',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show(fig1)
# fig1.savefig('P3_5.svg',format='svg')


K=3

#initialization
pi=np.tile(np.ones(K)/K,(Nb,1))
sigma=np.tile(np.ones(K),(Nb,1))
mu=np.tile(np.random.choice(x,size=K,replace=False),(Nb,1))
x=np.tile(x,(K,1)).T
p=pi*(1/np.sqrt(2*np.pi*sigma))*np.exp(-(x-mu)**2/(2*sigma))
p/=np.tile(np.sum(p,axis=1,keepdims=True),(1,K))
ll=np.sum(np.log(np.sum(p,axis=1)))

Iter=1
while Iter<100:
    #updating
    mu=np.sum(p*x,axis=0)/np.sum(p,axis=0)
    sigma=np.sum(p*(x-mu)**2,axis=0)/np.sum(p,axis=0)
    pi=np.sum(p,axis=0)/Nb

    pi=np.tile(pi,(Nb,1))
```

```
    sigma=np.tile(sigma,(Nb,1))
    mu=np.tile(mu,(Nb,1))
    p=pi*(1/np.sqrt(2*np.pi*sigma))*np.exp(-(x-mu)**2/(2*sigma))
    p/=np.tile(np.sum(p,axis=1,keepdims=True),(1,K))

    ll_new=np.sum(np.log(np.sum(p,axis=1)))
    if (ll_new-ll)/ll_new<1e-5:
        break
    ll=ll_new
    Iter+=1

mu=np.sum(p*x,axis=0)/np.sum(p,axis=0)
sigma=np.sum(p*(x-mu)**2,axis=0)/np.sum(p,axis=0)
pi=np.sum(p,axis=0)/Nb
p=(1/np.sqrt(2*np.pi*sigma))*np.exp(-(x-mu)**2/(2*sigma))

fig1=plt.figure(figsize=[20,10],dpi=300)
plt.plot(x[:,0],p[:,0],c="r")
plt.plot(x[:,0],p[:,1],c="g")
plt.plot(x[:,0],p[:,2],c="b")
plt.ylabel('Histogram',fontsize=20)
plt.xlabel('x',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()

fig1=plt.figure(figsize=[20,10],dpi=300)
plt.bar(x[:,0],np.sum(p,axis=1),width=50/Nb,facecolor='r',edgecolor=None,linew
idth=0.5)
plt.ylabel('Histogram',fontsize=20)
plt.xlabel('x',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
# fig1.savefig('P3_6.svg',format='svg')
```


Histogram
x
0
2
4
6
8
10
12
14
−20
−10
0
10
20
30

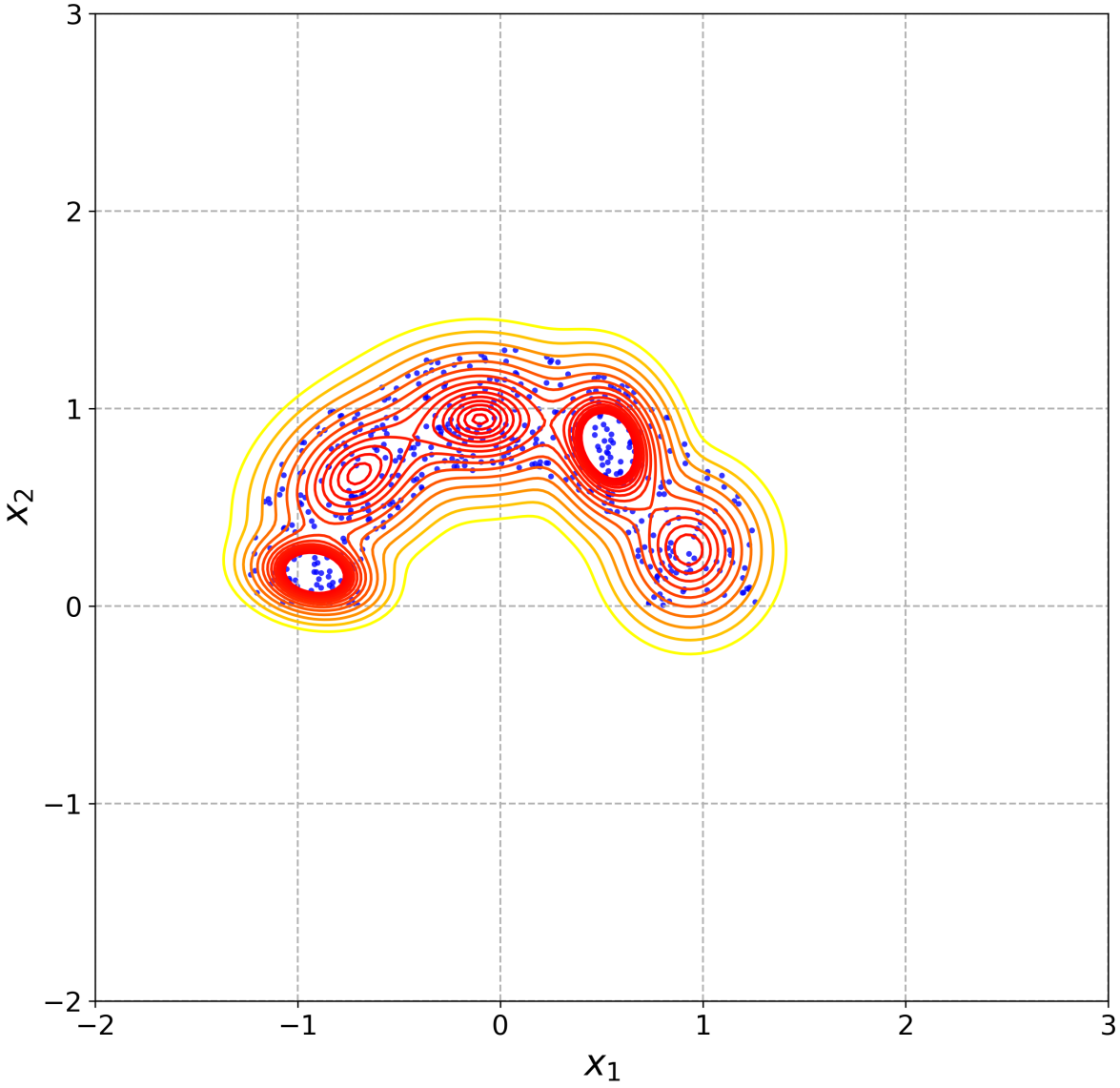

Histogram
x
0.00
0.02
0.04
0.06
0.08
0.10
−20
−10
0
10
20
30

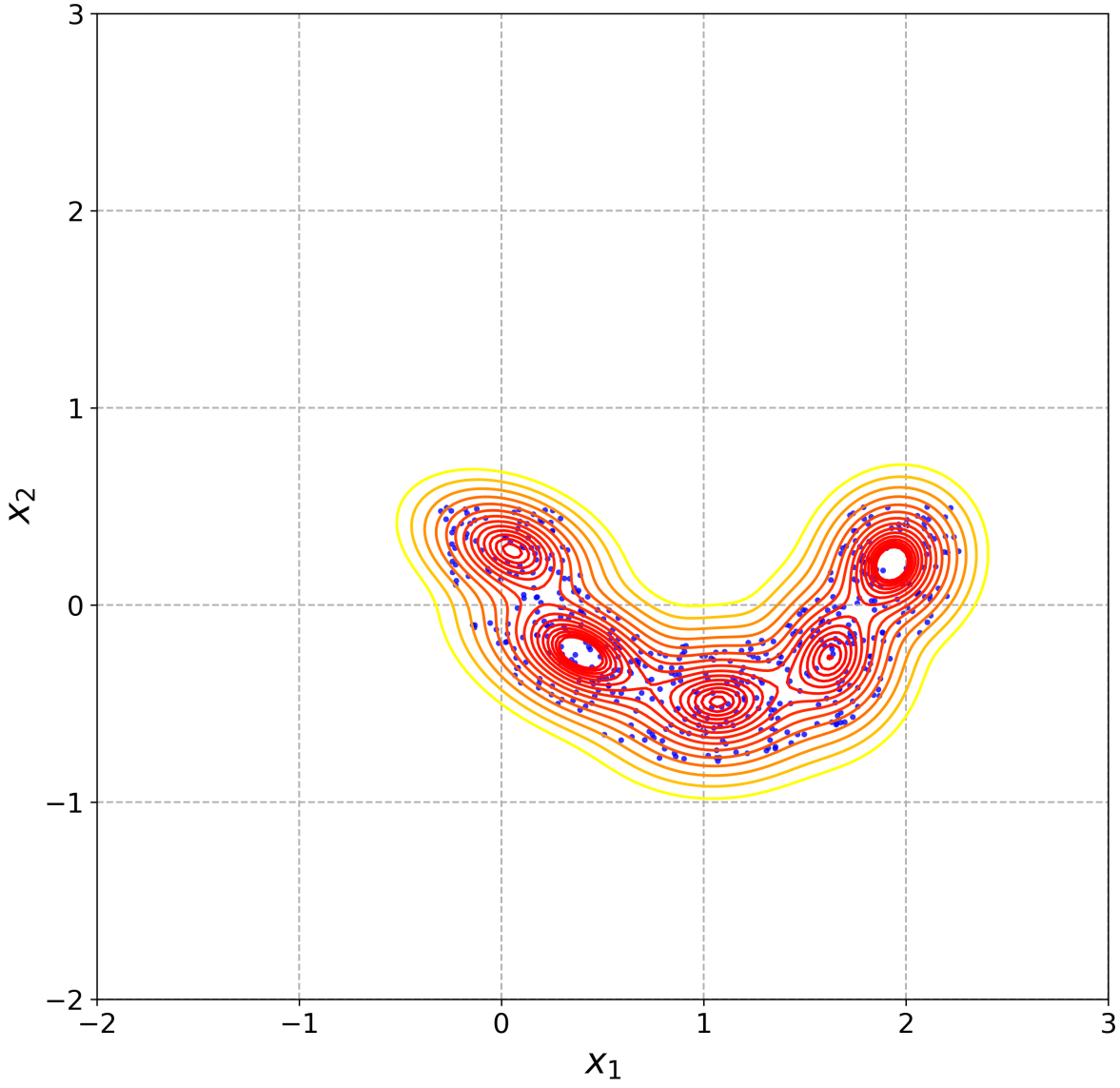

Histogram
x
0.00
0.02
0.04
0.06
0.08
0.10
0.12
−20
−10
0
10
20
30

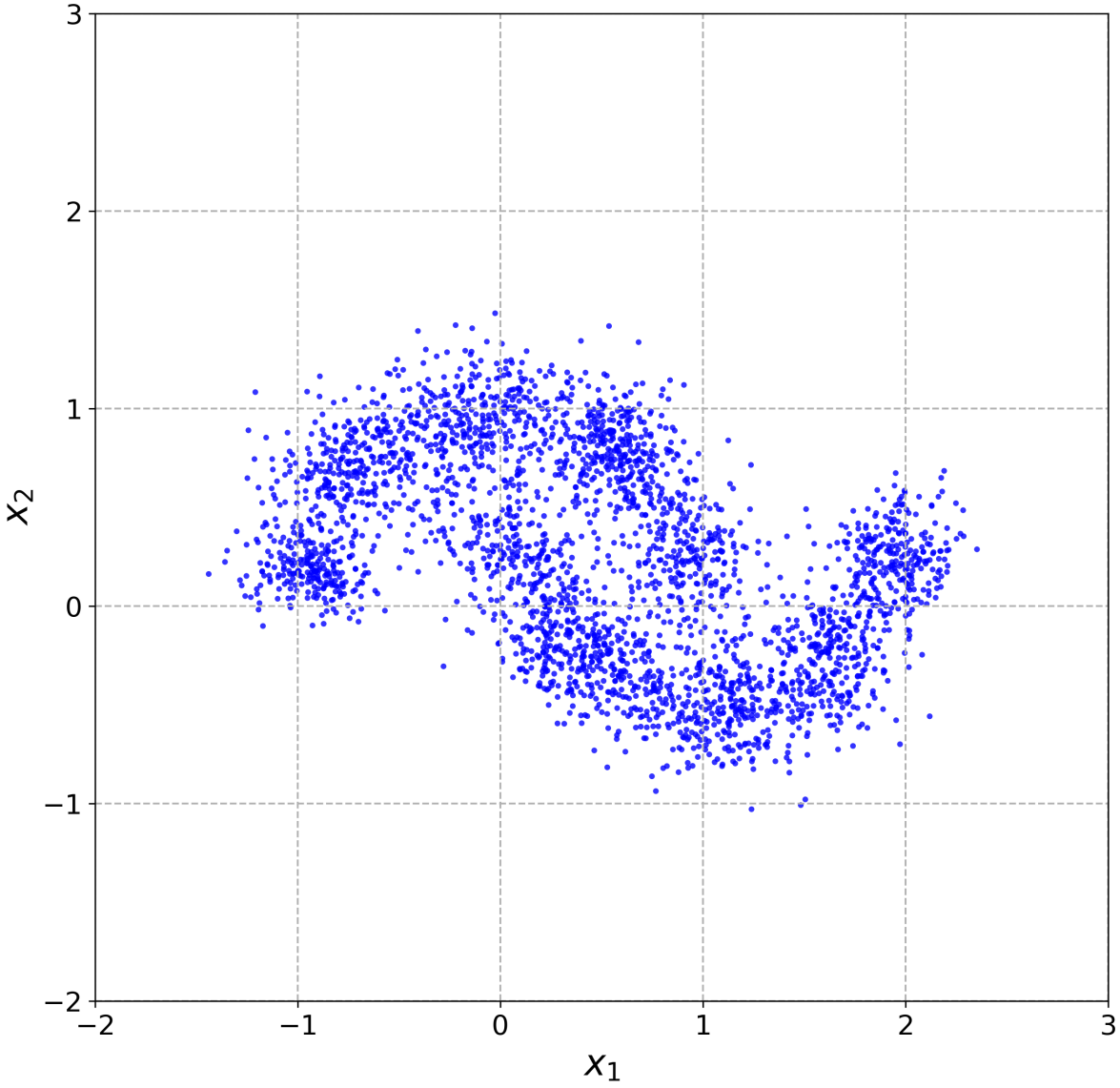
In [8]:
```
#P4
for name in dir():
    del globals()[name]

import numpy as np
import matplotlib.pyplot as plt
from sklearn import mixture

def doublemoon(N,d,r,w):
    ro1=np.random.uniform(low=r-w/2,high=r+w/2,size=N//2)
    t1=np.random.uniform(low=0,high=np.pi,size=N//2)
    x1=ro1*np.cos(t1)
    y1=ro1*np.sin(t1)
    l1=np.ones((1,N//2))

    ro2=np.random.uniform(low=r-w/2,high=r+w/2,size=N//2)
    t2=np.random.uniform(low=np.pi,high=2*np.pi,size=N//2)
    x2=ro2*np.cos(t2)+r
    y2=ro2*np.sin(t2)-d
    l2=-1*np.ones((1,N//2))

    E1=np.vstack((x1,y1,l1))
    E2=np.vstack((x2,y2,l2))
    E=np.hstack((E1,E2))
    return E

N=1000
d=-0.5
r=1
w=0.6
K=5

E = doublemoon(N,d,r,w).T

X_train1=E[0:500,0:2]
X_train2=E[500:,0:2]
# fit Gaussian Mixture Models
clf1 = mixture.GaussianMixture(n_components=K, covariance_type='full')
clf1.fit(X_train1)
clf2 = mixture.GaussianMixture(n_components=K, covariance_type='full')
clf2.fit(X_train2)

# display predicted scores by the model as a contour plot
x = np.linspace(-2., 3.,200)
y = np.linspace(-2., 3.,200)
X, Y = np.meshgrid(x, y)
XX = np.array([X.ravel(), Y.ravel()]).T
Z1 = -clf1.score_samples(XX)
Z1 = Z1.reshape(X.shape)
Z2 = -clf2.score_samples(XX)
Z2 = Z2.reshape(X.shape)

fig1=plt.figure(figsize=[10,10],dpi=300)
plt.contour(X, Y, Z1, levels=np.logspace(-1, .6, 15),cmap='autumn')
plt.scatter(X_train1[:,0], X_train1[:,1],s=10, alpha=0.8, c='b', edgecolors='n
one')
```

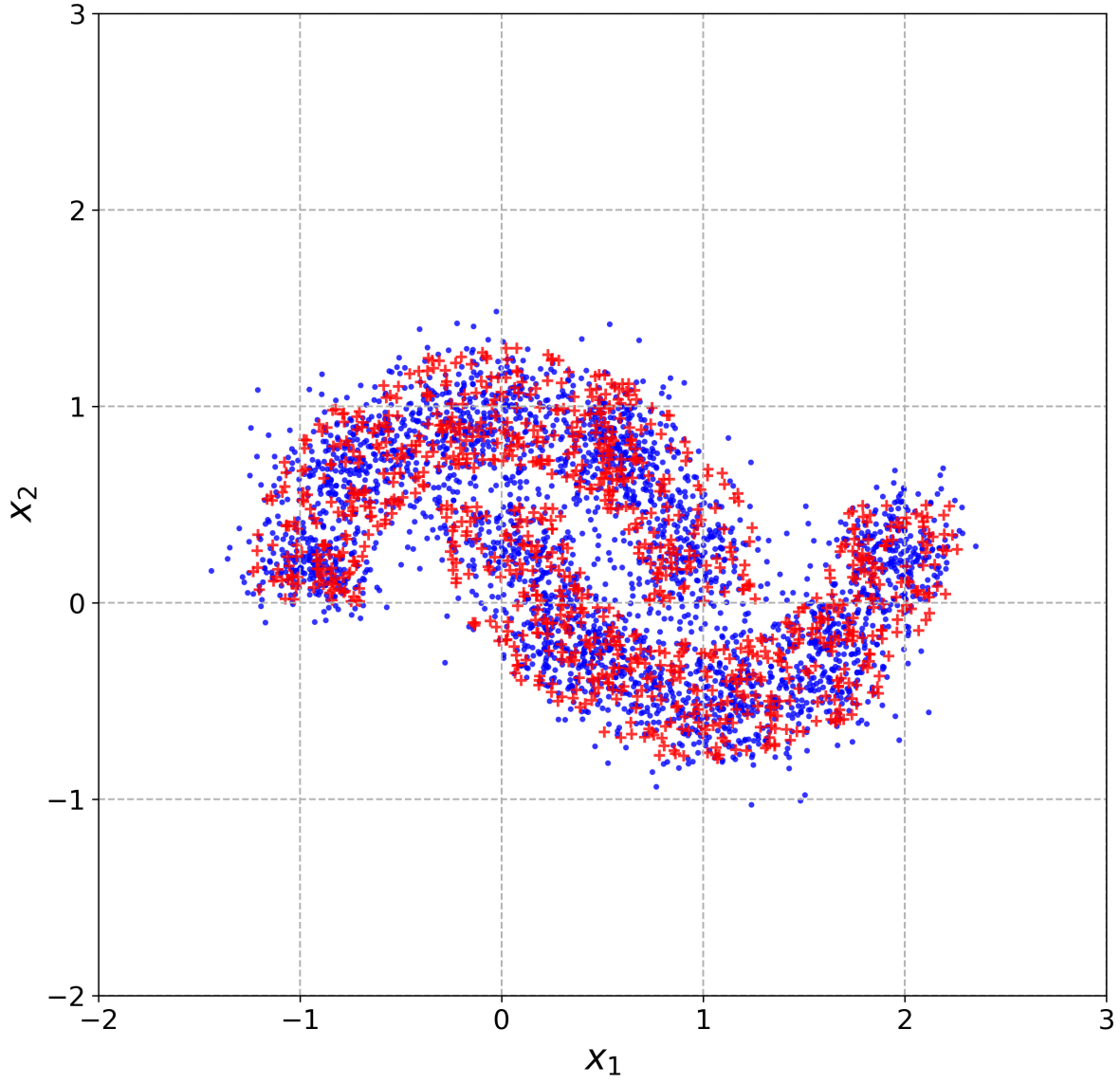
```
plt.grid('True',linestyle='--', linewidth=1)
plt.xlabel('$x_1$',fontsize=20)
plt.ylabel('$x_2$',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
# fig1.savefig('P4_14.svg',format='svg')

fig2=plt.figure(figsize=[10,10],dpi=300)
plt.contour(X, Y, Z2, levels=np.logspace(-1, .6, 15),cmap='autumn')
plt.scatter(X_train2[:,0], X_train2[:,1],s=10, alpha=0.8, c='b', edgecolors='n
one')
plt.grid('True',linestyle='--', linewidth=1)
plt.xlabel('$x_1$',fontsize=20)
plt.ylabel('$x_2$',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
# fig2.savefig('P4_15.svg',format='svg')

#_____________

N=3000
S1=clf1.sample(N//2)
S1=S1[0]
S2=clf2.sample(N//2)
S2=S2[0]

fig3=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(S1[:,0], S1[:,1],s=10, alpha=0.8, c='b', edgecolors='none')
plt.scatter(S2[:,0], S2[:,1],s=10, alpha=0.8, c='b', edgecolors='none')
plt.grid('True',linestyle='--', linewidth=1)
plt.axis([-2,3,-2,3])
plt.xlabel('$x_1$',fontsize=20)
plt.ylabel('$x_2$',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
# fig3.savefig('P4_16.svg',format='svg')

fig4=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(S1[:,0], S1[:,1],s=10, alpha=0.8, c='b', edgecolors='none')
plt.scatter(S2[:,0], S2[:,1],s=10, alpha=0.8, c='b', edgecolors='none')
plt.scatter(E[:,0], E[:,1],s=40, alpha=0.8, c='r', edgecolors='none',marker=
'+')
plt.grid('True',linestyle='--', linewidth=1)
plt.axis([-2,3,-2,3])
plt.xlabel('$x_1$',fontsize=20)
plt.ylabel('$x_2$',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
# fig4.savefig('P4_8.svg',format='svg')
#_________________

N=1000
```

```
E = doublemoon(N,d,r,w).T
l=E[:,-1]
SS1=clf1.score_samples(E[:,:2])
SS2=clf2.score_samples(E[:,:2])
d=np.where(SS1>SS2,1,-1)

acc=np.sum(np.abs(d+l)/2)/E.shape[0]
print("Classifier Accuracy=%s" % acc)

fig5=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(E[d+l==2,0], E[d+l==2,1], alpha=0.8,marker='+', c='b', edgecolors=
'none', s=75)
plt.scatter(E[d+l==-2,0], E[d+l==-2,1], alpha=0.8,marker='x', c='g', edgecolor
s='none', s=55)
plt.scatter(E[d+l==0,0], E[d+l==0,1], alpha=0.8,marker='*', c='r', edgecolors=
'none', s=95)
plt.axis([-2,3,-2,3])
plt.grid('True',linestyle='--', linewidth=1)
plt.xlabel('$x_1$',fontsize=20)
plt.ylabel('$x_2$',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show(fig5)
# fig5.savefig('P4_17.svg',format='svg')
```

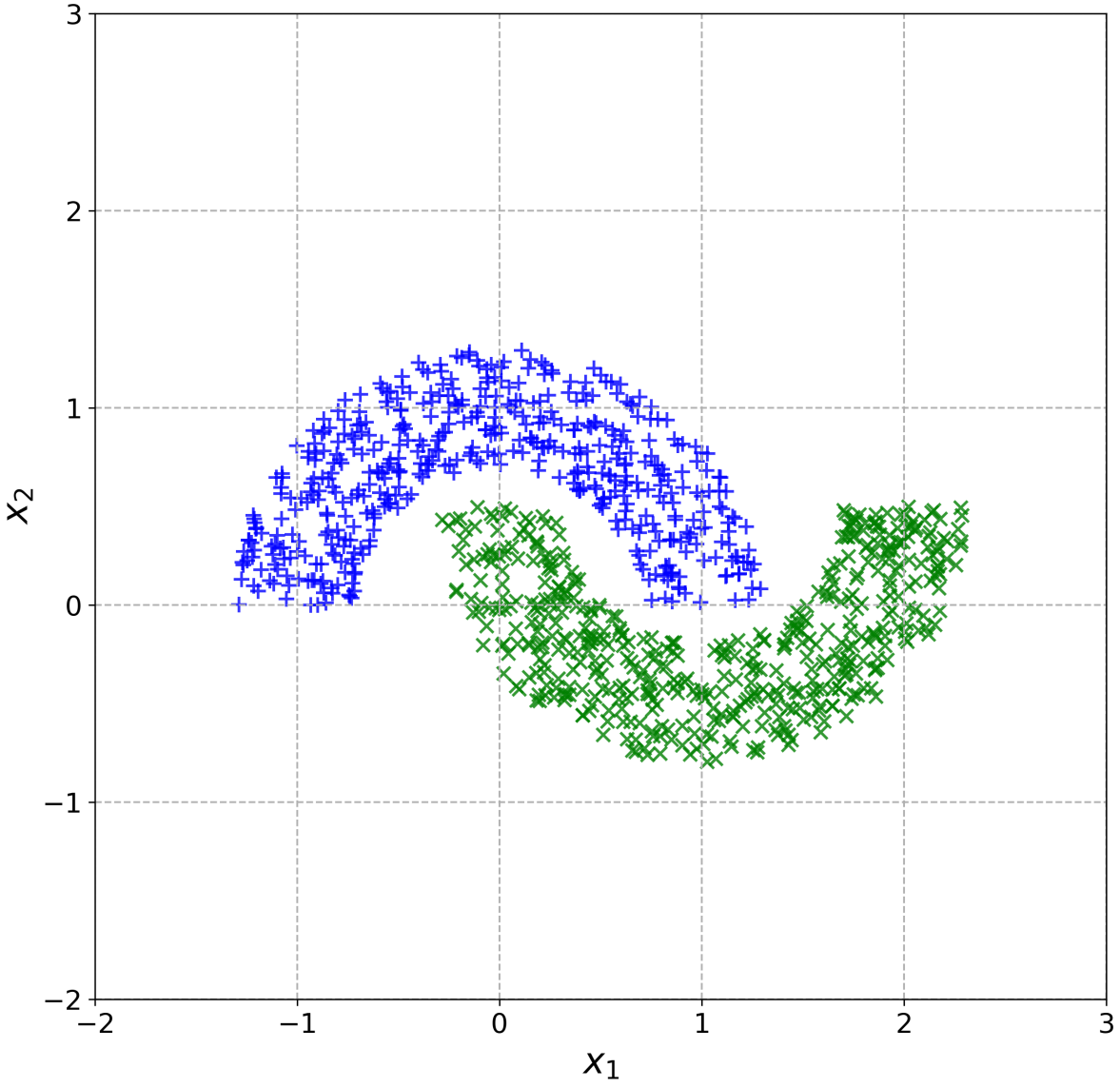

3
2
1
0
−1
−2
−2
−1
0
1
2
3
x_2
x_1


3
2
1
0
−1
−2
−2
−1
0
1
2
3
x_2
x_1


3
2
1
0
−1
−2
−2
−1
0
1
2
3
x_2
x_1




```
Classifier Accuracy=1.0
```



In [ ]: