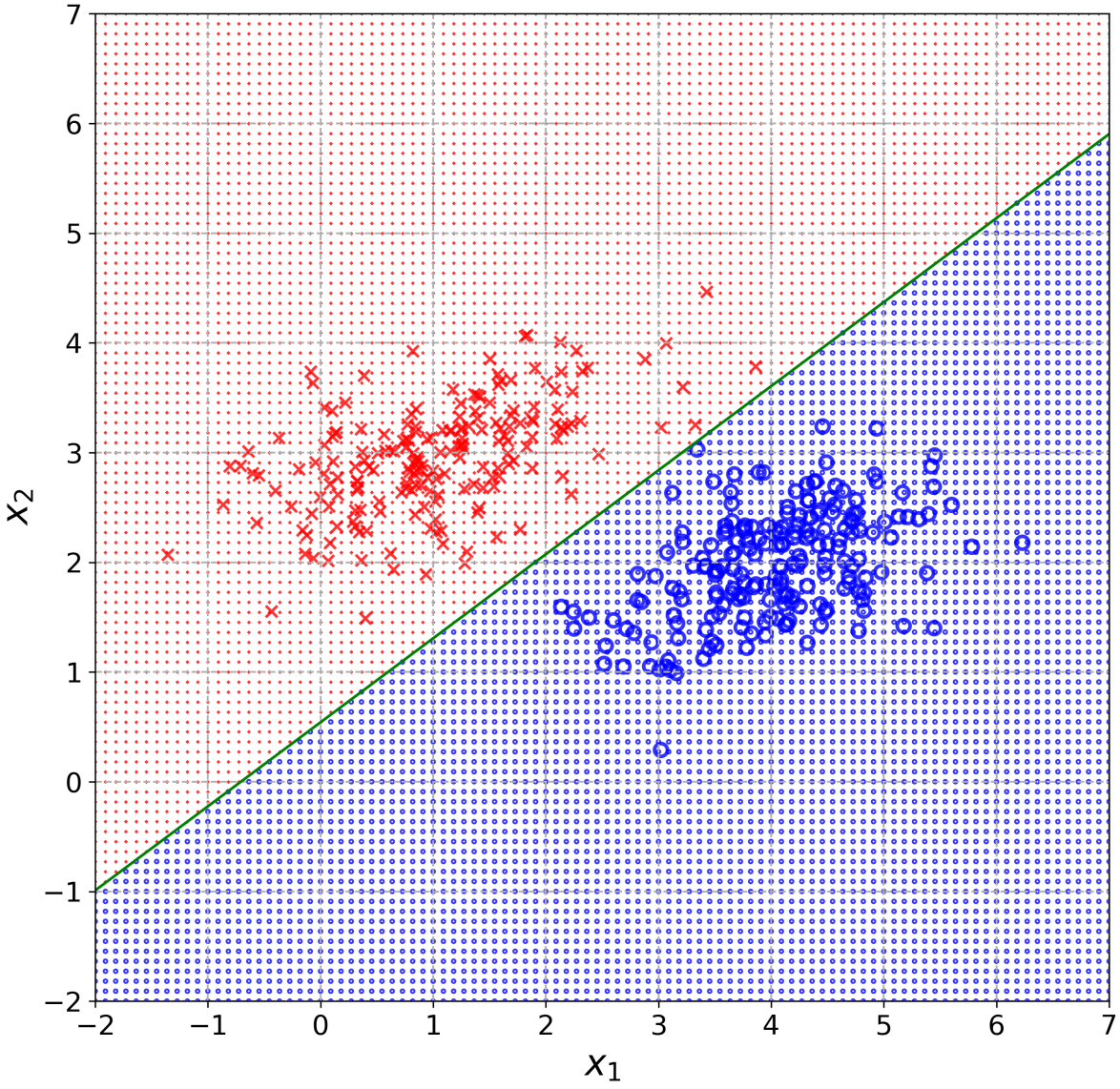
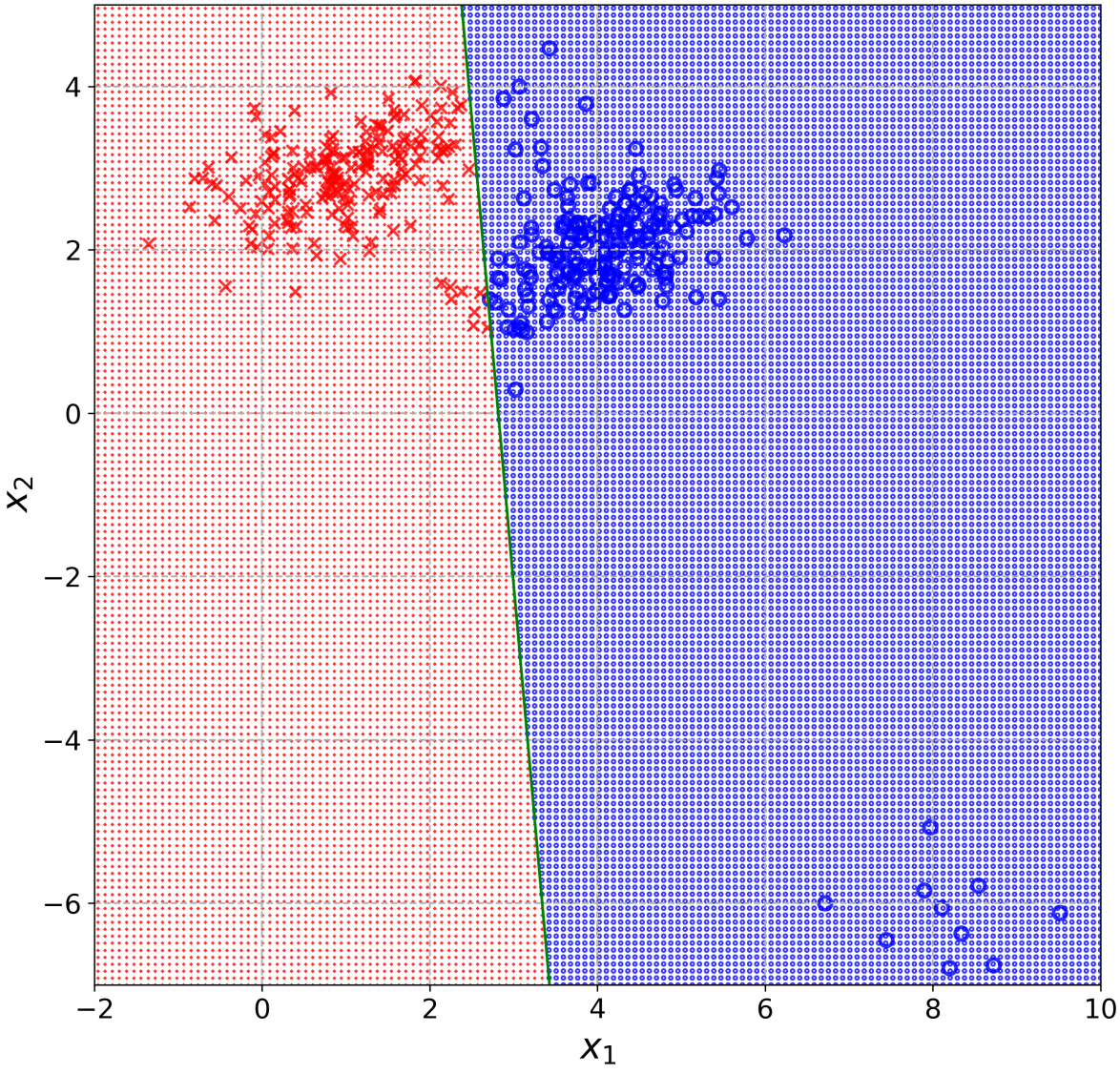
In [90]:
```
#P1_a)
for name in dir():
    del globals()[name]

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

ad=os.getcwd()
ad=ad+'\data\\'
X=np.loadtxt(ad+'DatasetA_data.csv', unpack='True',delimiter=',').T
X=np.hstack((np.ones((X.shape[0],1)),X))
T=np.reshape(np.loadtxt(ad+'DatasetA_labels.csv', unpack='True'),(X.shape[0],1
))

#making hot key coding
T=np.tile(T,(1,int(np.max(T))+1))
for i in np.arange(T.shape[1]):
    T[:,i]-=(i+1)
T=np.equal(T,-1*np.ones((np.shape(T))))*1

#W calculation
W=((np.linalg.inv(X.T@X))@X.T)@T

#classifyimg
y=X@W
C=np.equal(y,np.tile(np.amax(y,axis=1,keepdims=True),(1,T.shape[1])))*1

#printing the grid
nn=100
nodes=np.linspace(-2,7,nn)
x1, x2 = np.meshgrid(nodes, nodes)
NodeTag=np.zeros((nn,nn))
crd=np.stack((x1,x2),axis=2)
crd=np.reshape(crd,(10000,2),order='C')
crd=np.hstack((np.ones((crd.shape[0],1)),crd))

y2=crd@W
C2=np.equal(y2,np.tile(np.amax(y2,axis=1,keepdims=True),(1,T.shape[1])))*1

#hyper planes
x=np.linspace(np.min(X[:,1]-1),np.max(X[:,1])+1,3)
y1=(+0.5*W[0,0]-0.5*W[0,1]+W[1,0]*x)/-W[2,0]
#indexing data according to lables
i0=np.squeeze(np.equal(C[:,0,None],np.ones((C.shape[0],1))))
i1=np.squeeze(np.equal(C[:,1,None],np.ones((C.shape[0],1))))
i2=np.squeeze(np.equal(C2[:,0,None],np.ones((C2.shape[0],1))))
i3=np.squeeze(np.equal(C2[:,1,None],np.ones((C2.shape[0],1))))
#plotting
fig=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(X[i0,1], X[i0,2], alpha=0.8,marker='x', c='r', edgecolors='none',
s=40)
plt.scatter(X[i1,1], X[i1,2], alpha=0.8,marker='o', facecolor='none', edgecolo
rs='blue', s=50, linewidth=2)
plt.scatter(crd[i2,1], crd[i2,2], alpha=0.8,marker='x', c='r', edgecolors='non
e', s=1)
```

```
plt.scatter(crd[i3,1], crd[i3,2], alpha=0.8,marker='o', facecolor='none', edge
colors='blue', s=1, linewidth=2)
plt.plot(x,y1,color='g')
plt.axis([-2,7,-2,7])
# plt.axis('equal')
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()
# fig.savefig('p1_a.svg',format='svg')
```



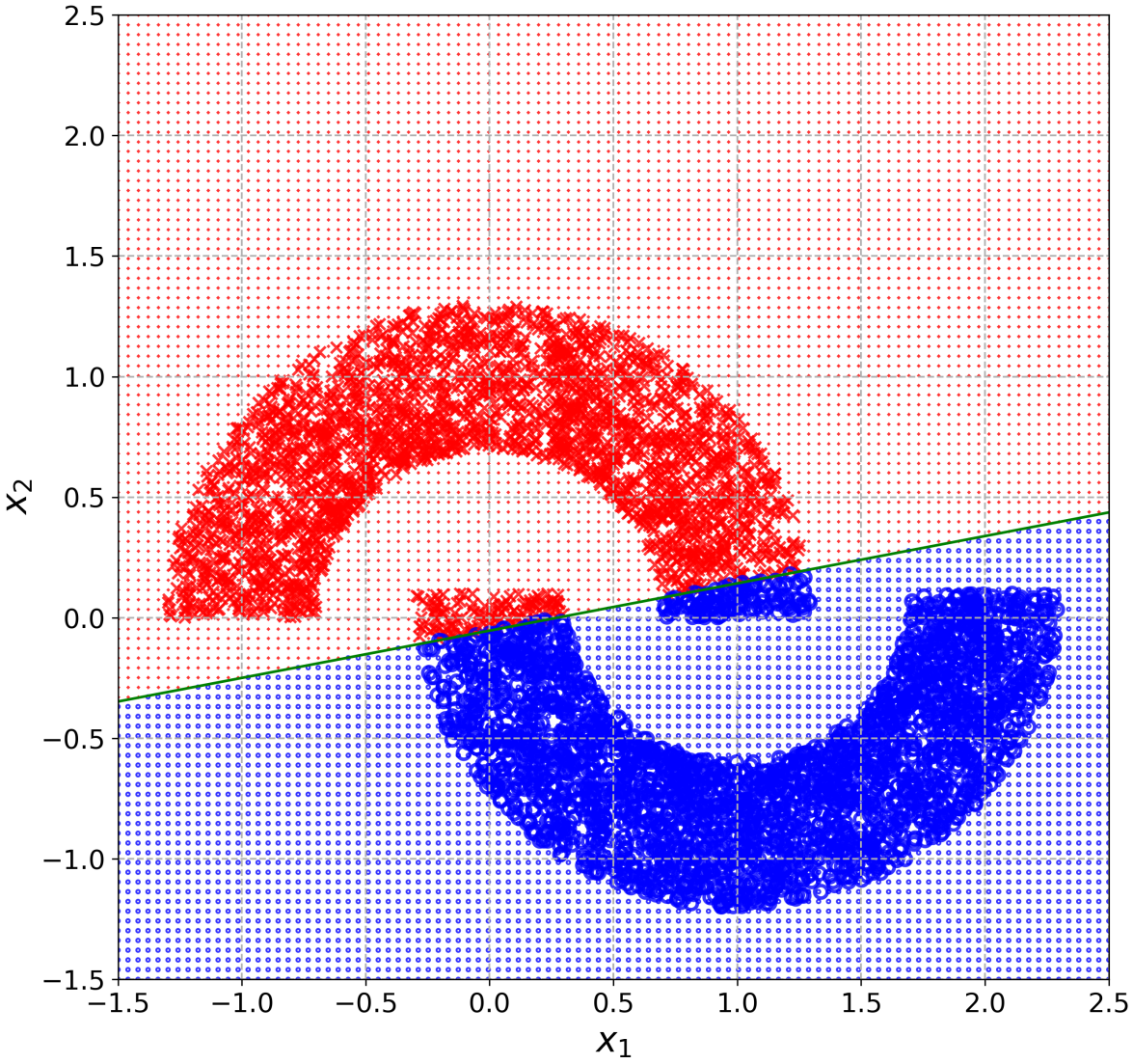
In [91]:
```
#P1_b)
for name in dir():
    del globals()[name]

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

ad=os.getcwd()
ad=ad+'\data\\'
X=np.loadtxt(ad+'DatasetA2_data.csv', unpack='True',delimiter=',').T
X=np.hstack((np.ones((X.shape[0],1)),X))
T=np.reshape(np.loadtxt(ad+'DatasetA2_labels.csv', unpack='True'),(X.shape[0],
1))

#making hot key coding
T=np.tile(T,(1,int(np.max(T))+1))
for i in np.arange(T.shape[1]):
    T[:,i]-=(i+1)
T=np.equal(T,-1*np.ones((np.shape(T))))*1

#W calculation
W=((np.linalg.inv(X.T@X))@X.T)@T

#classifyimg
y=X@W
C=np.equal(y,np.tile(np.amax(y,axis=1,keepdims=True),(1,T.shape[1])))*1

#printing the grid
nn=200
nodes=np.linspace(-7,10,nn)
x1, x2 = np.meshgrid(nodes, nodes)
NodeTag=np.zeros((nn,nn))
crd=np.stack((x1,x2),axis=2)
crd=np.reshape(crd,(40000,2),order='C')
crd=np.hstack((np.ones((crd.shape[0],1)),crd))

y2=crd@W
C2=np.equal(y2,np.tile(np.amax(y2,axis=1,keepdims=True),(1,T.shape[1])))*1

#hyper planes
x=np.linspace(np.min(X[:,1]),np.max(X[:,1]),3)
y1=(+0.5*W[0,0]-0.5*W[0,1]+W[1,0]*x)/-W[2,0]
#indexing data according to lables
i0=np.squeeze(np.equal(C[:,0,None],np.ones((C.shape[0],1))))
i1=np.squeeze(np.equal(C[:,1,None],np.ones((C.shape[0],1))))
i2=np.squeeze(np.equal(C2[:,0,None],np.ones((C2.shape[0],1))))
i3=np.squeeze(np.equal(C2[:,1,None],np.ones((C2.shape[0],1))))
#plotting
fig=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(X[i0,1], X[i0,2], alpha=0.8,marker='x', c='r', edgecolors='none',
s=40)
plt.scatter(X[i1,1], X[i1,2], alpha=0.8,marker='o', facecolor='none', edgecolo
rs='blue', s=50, linewidth=2)
plt.scatter(crd[i2,1], crd[i2,2], alpha=0.8,marker='x', c='r', edgecolors='non
e', s=1)
```

```
plt.scatter(crd[i3,1], crd[i3,2], alpha=0.8,marker='o', facecolor='none', edge
colors='blue', s=1, linewidth=2)
plt.plot(x,y1,color='g')
plt.axis([-2,10,-7,5])
# plt.axis('equal')
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()
# fig.savefig('p1_b.svg',format='svg')
```



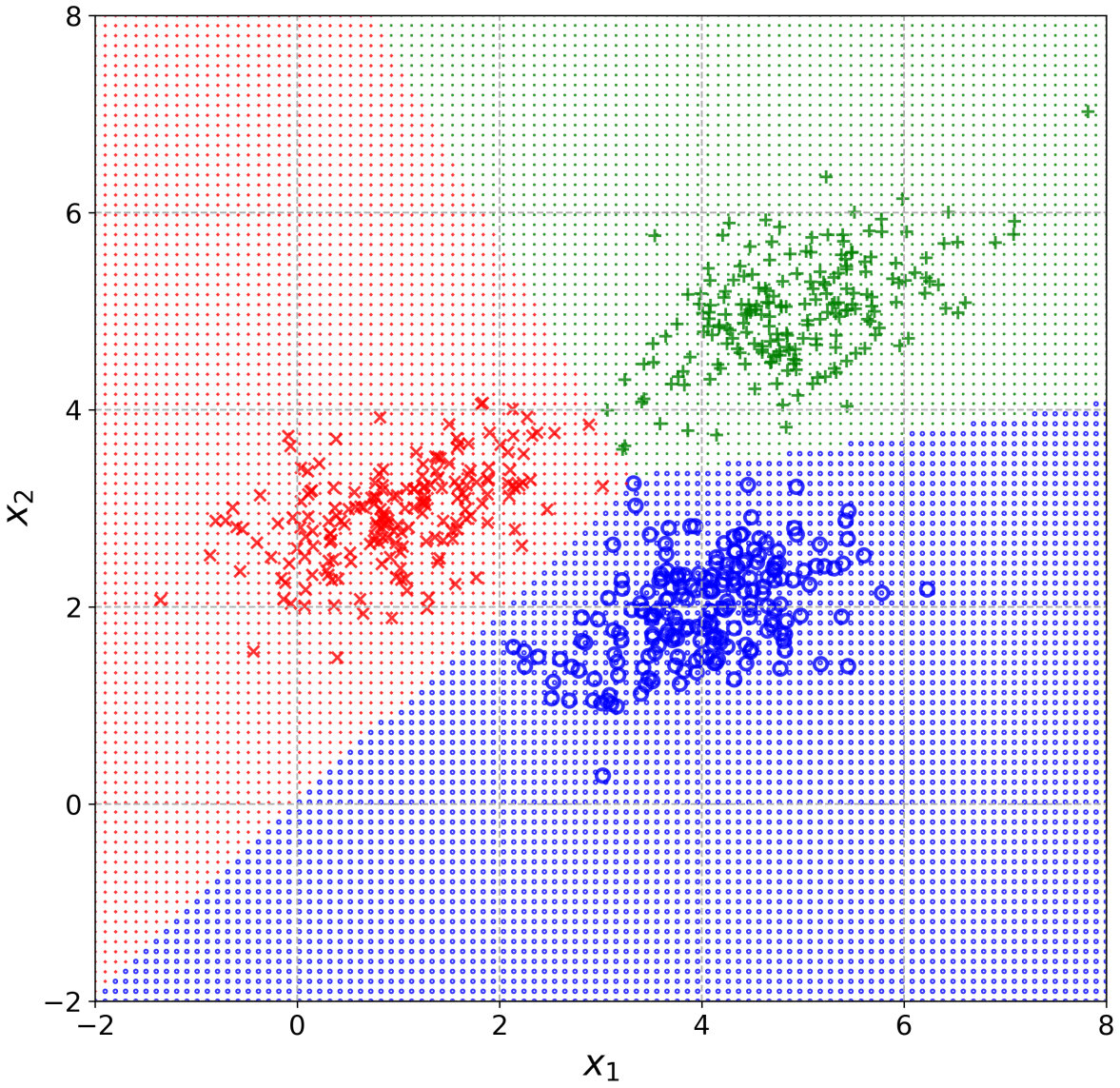
In [92]:
```
#P1_c)
for name in dir():
    del globals()[name]

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

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.zeros((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=np.ones((1,N//2))

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

N=5000
d=-0.1
r=1
w=0.6
E=doublemoon(N,d,r,w)
E=E.T

X=np.vstack((E[:,0:2],E[:,3:5]))
X=np.hstack((np.ones((X.shape[0],1)),X))
T=np.vstack((E[:,2,None],E[:,5,None]))

#making hot key coding
T=np.tile(T,(1,int(np.max(T))+1))
for i in np.arange(T.shape[1]):
    T[:,i]-=(i+1)
T=np.equal(T,-1*np.ones((np.shape(T))))*1

#W calculation
W=((np.linalg.inv(X.T@X))@X.T)@T

#classifyimg
y=X@W
C=np.equal(y,np.tile(np.amax(y,axis=1,keepdims=True),(1,T.shape[1])))*1

#printing the grid
nn=100
nodes=np.linspace(-1.5,2.5,nn)
x1, x2 = np.meshgrid(nodes, nodes)
NodeTag=np.zeros((nn,nn))
crd=np.stack((x1,x2),axis=2)
crd=np.reshape(crd,(10000,2),order='C')
crd=np.hstack((np.ones((crd.shape[0],1)),crd))
```

```
y2=crd@W
C2=np.equal(y2,np.tile(np.amax(y2,axis=1,keepdims=True),(1,T.shape[1])))*1

#hyper planes
x=np.linspace(-5,5,3)
y1=(+0.5*W[0,0]-0.5*W[0,1]+W[1,0]*x)/-W[2,0]
#indexing data according to lables
i0=np.squeeze(np.equal(C[:,0,None],np.ones((C.shape[0],1))))
i1=np.squeeze(np.equal(C[:,1,None],np.ones((C.shape[0],1))))
i2=np.squeeze(np.equal(C2[:,0,None],np.ones((C2.shape[0],1))))
i3=np.squeeze(np.equal(C2[:,1,None],np.ones((C2.shape[0],1))))
#plotting
fig=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(X[i0,1], X[i0,2], alpha=0.8,marker='x', c='r', edgecolors='none',
s=40)
plt.scatter(X[i1,1], X[i1,2], alpha=0.8,marker='o', facecolor='none', edgecolo
rs='blue', s=50, linewidth=2)
plt.scatter(crd[i2,1], crd[i2,2], alpha=0.8,marker='x', c='r', edgecolors='non
e', s=1)
plt.scatter(crd[i3,1], crd[i3,2], alpha=0.8,marker='o', facecolor='none', edge
colors='blue', s=1, linewidth=2)
plt.plot(x,y1,color='g')
plt.axis([-1.5,2.5,-1.5,2.5])
# plt.axis('equal')
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()
# fig.savefig('p1_c.svg',format='svg')
```


2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
x_2
x_1

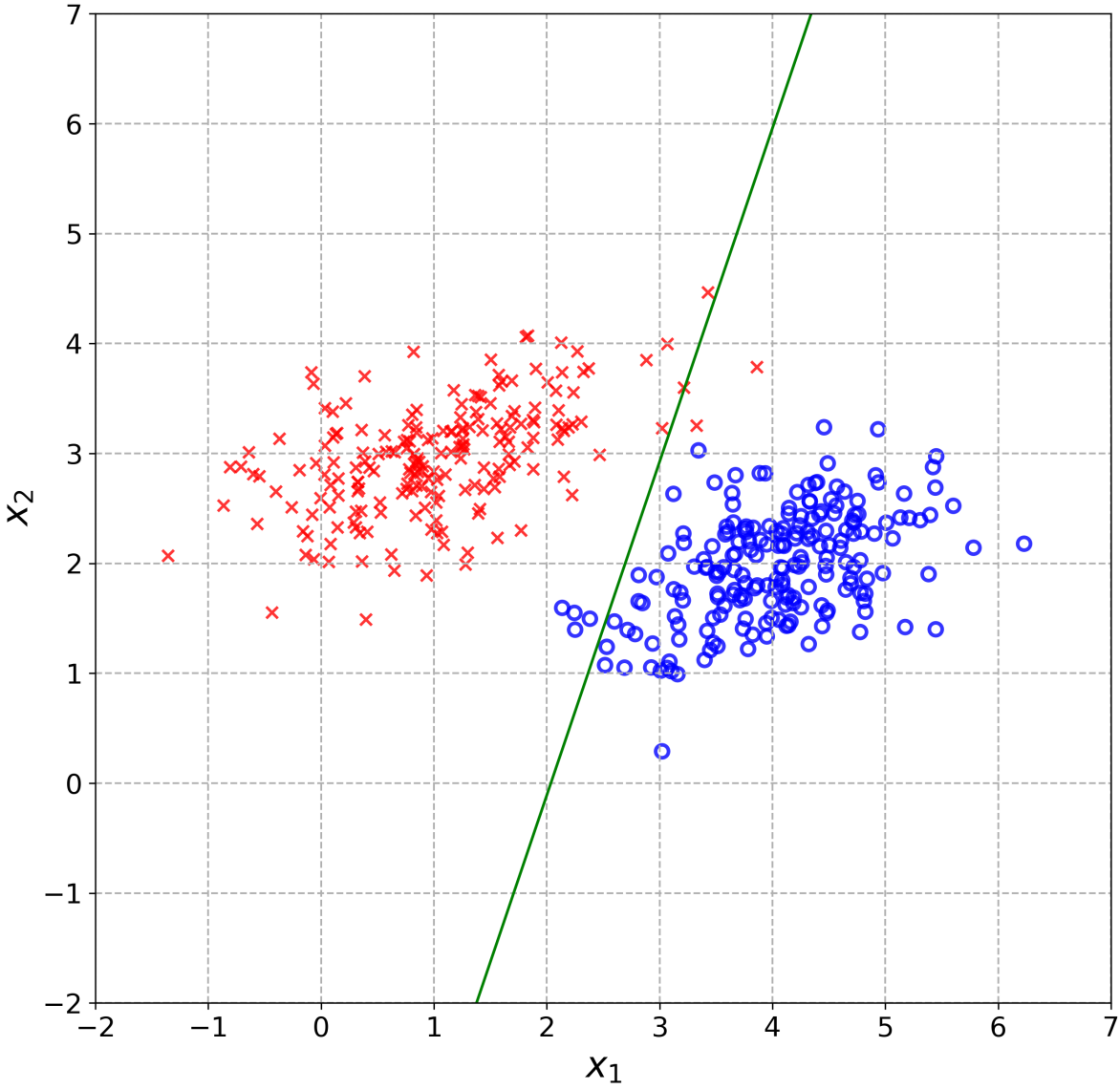
In [93]:
```
#P2)
for name in dir():
    del globals()[name]

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

ad=os.getcwd()
ad=ad+'\data\\'
X=np.loadtxt(ad+'DatasetB_data.csv', unpack='True',delimiter=',').T
X=np.hstack((np.ones((X.shape[0],1)),X))
T=np.reshape(np.loadtxt(ad+'DatasetB_labels.csv', unpack='True'),(X.shape[0],1
))

#making hot key coding
T=np.tile(T,(1,int(np.max(T))+1))
for i in np.arange(T.shape[1]):
    T[:,i]-=(i+1)
T=np.equal(T,-1*np.ones((np.shape(T))))*1

#W calculation
W=((np.linalg.inv(X.T@X))@X.T)@T

#classifyimg
y=X@W
C=np.equal(y,np.tile(np.amax(y,axis=1,keepdims=True),(1,T.shape[1])))*1

#printing the grid
nn=100
nodes=np.linspace(-2,8,nn)
x1, x2 = np.meshgrid(nodes, nodes)
NodeTag=np.zeros((nn,nn))
crd=np.stack((x1,x2),axis=2)
crd=np.reshape(crd,(10000,2),order='C')
crd=np.hstack((np.ones((crd.shape[0],1)),crd))

y2=crd@W
C2=np.equal(y2,np.tile(np.amax(y2,axis=1,keepdims=True),(1,T.shape[1])))*1

#indexing data according to lables
i0=np.squeeze(np.equal(C[:,0,None],np.ones((C.shape[0],1))))
i1=np.squeeze(np.equal(C[:,1,None],np.ones((C.shape[0],1))))
i2=np.squeeze(np.equal(C[:,2,None],np.ones((C.shape[0],1))))
i3=np.squeeze(np.equal(C2[:,0,None],np.ones((C2.shape[0],1))))
i4=np.squeeze(np.equal(C2[:,1,None],np.ones((C2.shape[0],1))))
i5=np.squeeze(np.equal(C2[:,2,None],np.ones((C2.shape[0],1))))
#plotting
fig=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(X[i0,1], X[i0,2], alpha=0.8,marker='x', c='r', s=40)
plt.scatter(X[i1,1], X[i1,2], alpha=0.8,marker='o', facecolor='none', edgecolo
rs='blue', s=50, linewidth=2)
plt.scatter(X[i2,1], X[i2,2], alpha=0.8,marker='+', c='g', s=50)
plt.scatter(crd[i3,1], crd[i3,2], alpha=0.8,marker='x', c='r', edgecolors='non
e', s=1)
plt.scatter(crd[i4,1], crd[i4,2], alpha=0.8,marker='o', facecolor='none', edge
```

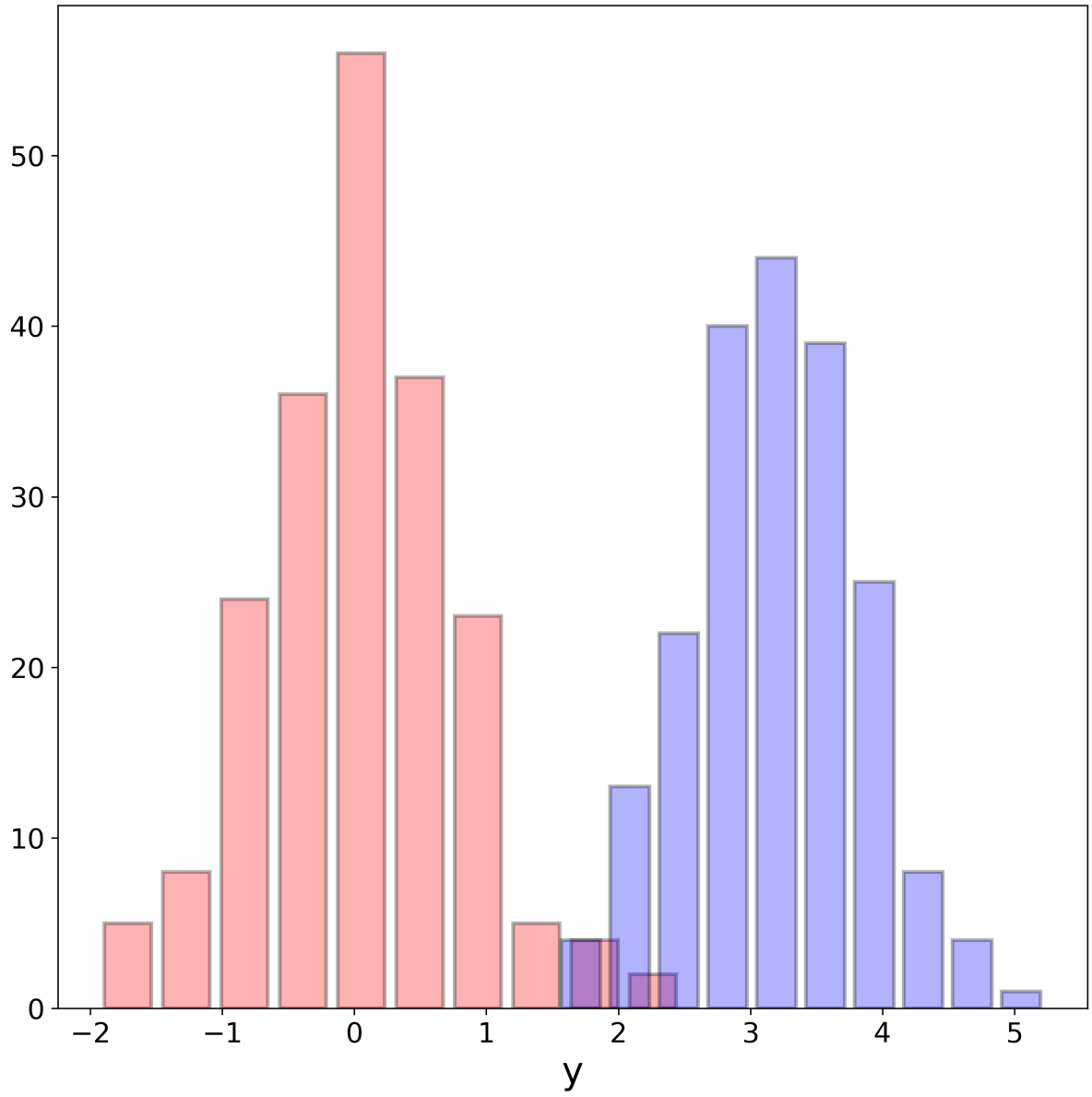
```
colors='blue', s=1, linewidth=2)
plt.scatter(crd[i5,1], crd[i5,2], alpha=0.8,marker='+', c='g', s=1)
plt.axis([-2,8,-2,8])
# plt.axis('scaled')
plt.grid('True',linestyle='--', linewidth=1)
plt.xlabel('$x_1$',fontsize=20)
plt.ylabel('$x_2$',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
# fig.savefig('p2.svg',format='svg')
```



In [108]:
```
#P3_a)
for name in dir():
    del globals()[name]

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

ad=os.getcwd()
ad=ad+'\data\\'
X=np.loadtxt(ad+'DatasetA_data.csv', unpack='True',delimiter=',').T
T=np.reshape(np.loadtxt(ad+'DatasetA_labels.csv', unpack='True'),(X.shape[0],1
))

#making hot key coding
T=np.tile(T,(1,int(np.max(T))+1))
for i in np.arange(T.shape[1]):
    T[:,i]-=(i+1)
T=np.equal(T,-1*np.ones((np.shape(T))))*1
T.astype(int)

#calcultating mean of each class
m=np.zeros((X.shape[1],T.shape[1]))
for i in np.arange(T.shape[1]):
    aa=list(map(bool,T[:,i]))
    m[:,i]=np.mean(X[aa,:],axis=0)

w=m[:,1]-m[:,0]
w/=np.linalg.norm(w)

y=X@w
y0=np.mean(y)-w[1]
C=(y>y0)*1
C=np.reshape(C,(X.shape[0],1))
#making hot key coding
C=np.tile(C,(1,int(np.max(C))+1))
for i in np.arange(C.shape[1]):
    C[:,i]-=(i+1)
C=np.equal(C,-1*np.ones((np.shape(C))))*1

#hyper planes
x=np.linspace(np.min(X[:,1]-1),np.max(X[:,1])+1,3)
y1=(w[0]*x-y0)/-w[1]
#indexing data according to lables
i0=np.squeeze(np.equal(T[:,0,None],np.ones((T.shape[0],1))))
i1=np.squeeze(np.equal(T[:,1,None],np.ones((T.shape[0],1))))

#plotting
fig=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(X[i0,0], X[i0,1], alpha=0.8,marker='x', c='r', edgecolors='none',
s=40)
plt.scatter(X[i1,0], X[i1,1], alpha=0.8,marker='o', facecolor='none', edgecolo
rs='blue', s=50, linewidth=2)
plt.plot(x,y1,color='g')
plt.axis([-2,7,-2,7])
plt.grid('True',linestyle='--', linewidth=1)
```

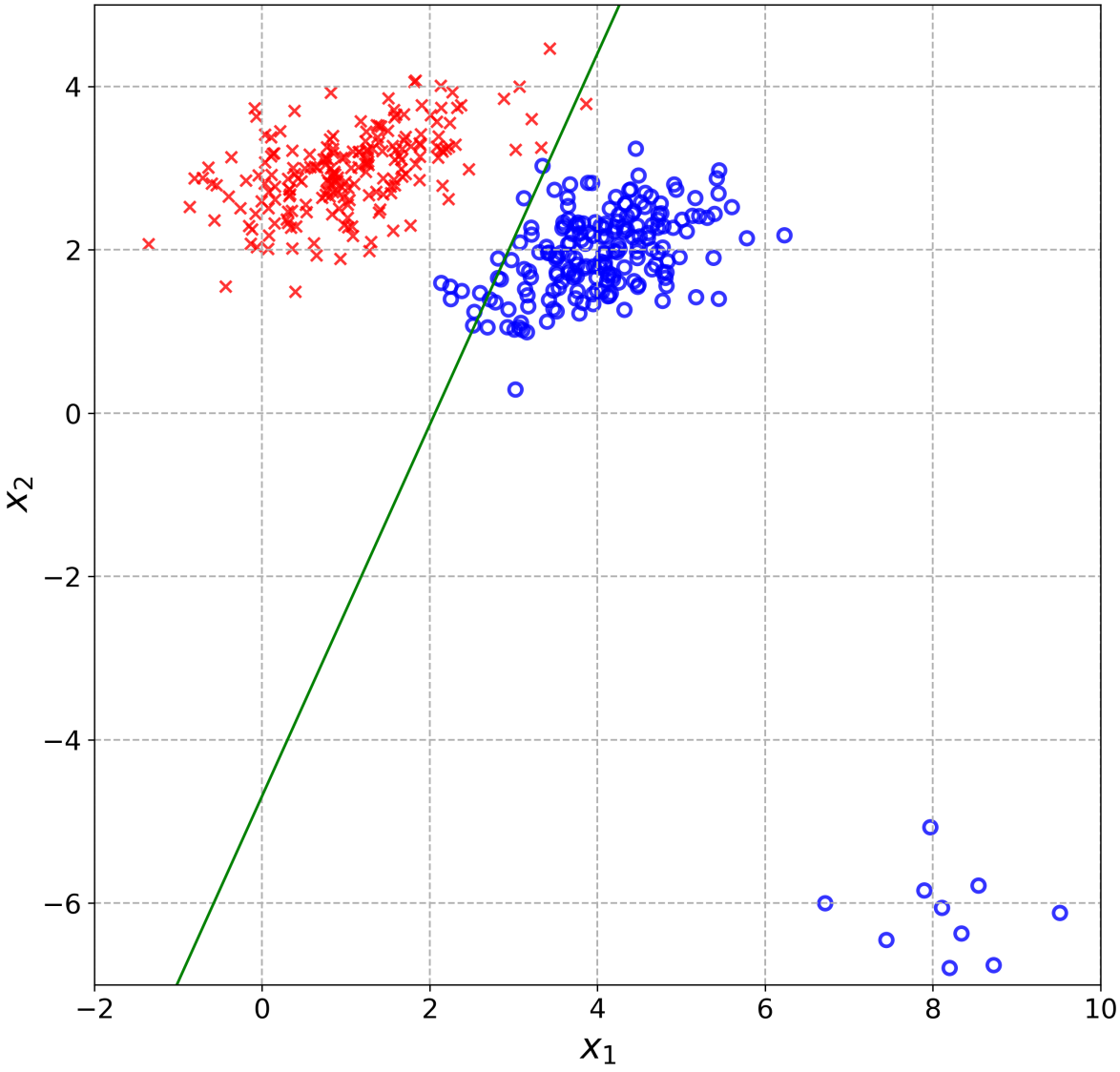
```
plt.xlabel('$x_1$',fontsize=20)
plt.ylabel('$x_2$',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
# fig.savefig('p3_a1.svg',format='svg')

fig2=plt.figure(figsize=[10,10],dpi=300)
plt.hist(y[T[:,0]==1],10,alpha=0.3, align='mid',rwidth=0.8,facecolor='r',edgec
olor='black',linewidth=2)
plt.hist(y[T[:,1]==1],10,alpha=0.3, align='mid',rwidth=0.8,facecolor='b',edgec
olor='black',linewidth=2)
plt.xlabel('y',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
# fig2.savefig('p3_a2.svg',format='svg')
```

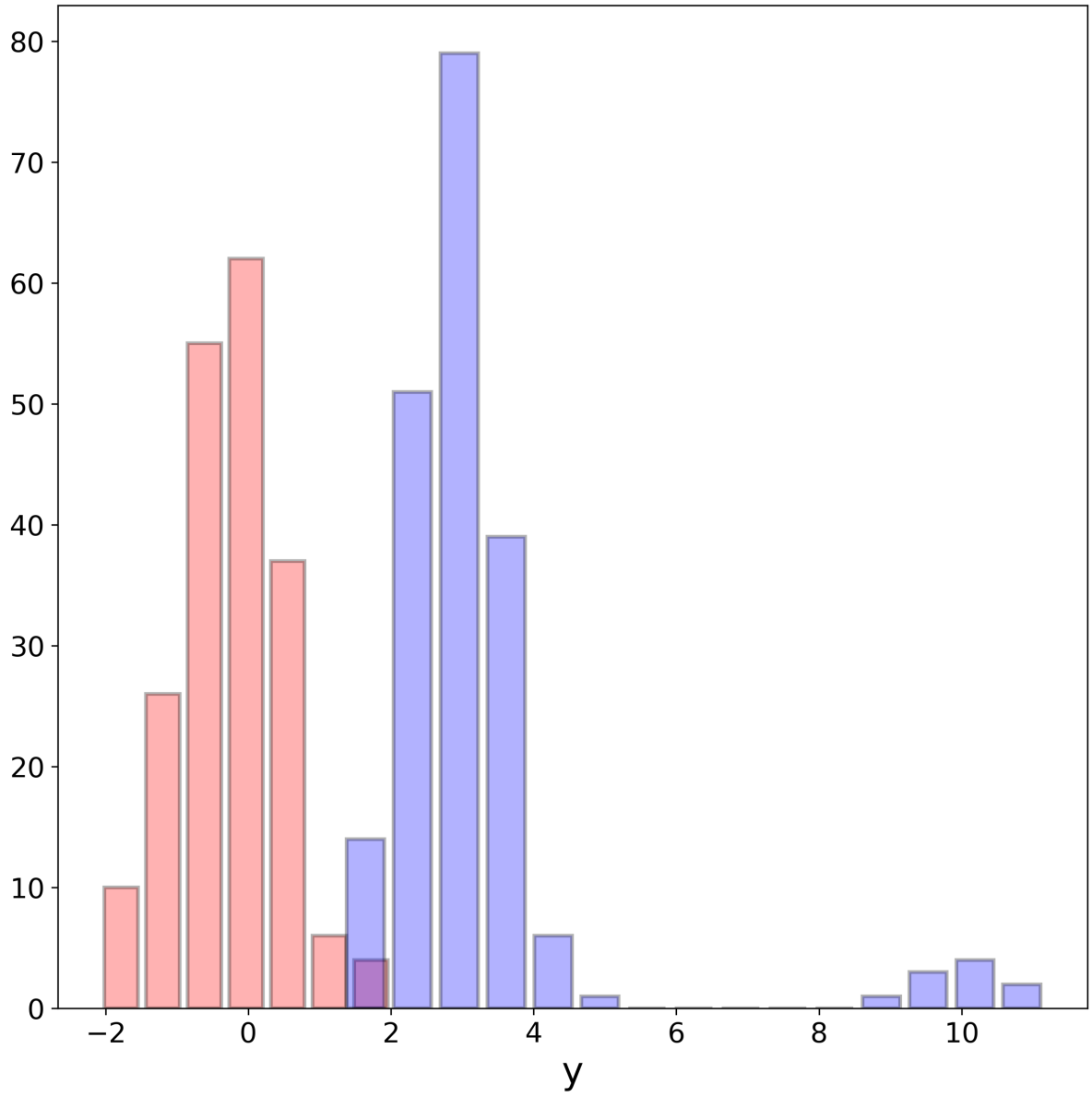

7
6
5
4
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4
5
6
7
x_2
x_1


50
40
30
20
10
0
−2
−1
0
1
2
3
4
5
y

In [104]:
```
#P3_b)
for name in dir():
    del globals()[name]

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

ad=os.getcwd()
ad=ad+'\data\\'
X=np.loadtxt(ad+'DatasetA2_data.csv', unpack='True',delimiter=',').T
T=np.reshape(np.loadtxt(ad+'DatasetA2_labels.csv', unpack='True'),(X.shape[0],
1))

#making hot key coding
T=np.tile(T,(1,int(np.max(T))+1))
for i in np.arange(T.shape[1]):
    T[:,i]-=(i+1)
T=np.equal(T,-1*np.ones((np.shape(T))))*1
T.astype(int)

#calcultating mean of each class
m=np.zeros((X.shape[1],T.shape[1]))
for i in np.arange(T.shape[1]):
    aa=list(map(bool,T[:,i]))
    m[:,i]=np.mean(X[aa,:],axis=0)

w=m[:,1]-m[:,0]
w/=np.linalg.norm(w)

y=X@w
y0=np.mean(y)-w[1]
C=(y>y0)*1
C=np.reshape(C,(X.shape[0],1))
#making hot key coding
C=np.tile(C,(1,int(np.max(C))+1))
for i in np.arange(C.shape[1]):
    C[:,i]-=(i+1)
C=np.equal(C,-1*np.ones((np.shape(C))))*1

#hyper planes
x=np.linspace(np.min(X[:,1]-1),np.max(X[:,1])+1,3)
y1=(w[0]*x-y0)/-w[1]
#indexing data according to lables
i0=np.squeeze(np.equal(T[:,0,None],np.ones((T.shape[0],1))))
i1=np.squeeze(np.equal(T[:,1,None],np.ones((T.shape[0],1))))

#plotting
fig=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(X[i0,0], X[i0,1], alpha=0.8,marker='x', c='r', edgecolors='none',
s=40)
plt.scatter(X[i1,0], X[i1,1], alpha=0.8,marker='o', facecolor='none', edgecolo
rs='blue', s=50, linewidth=2)
plt.plot(x,y1,color='g')
plt.axis([-2,10,-7,5])
plt.grid('True',linestyle='--', linewidth=1)
```

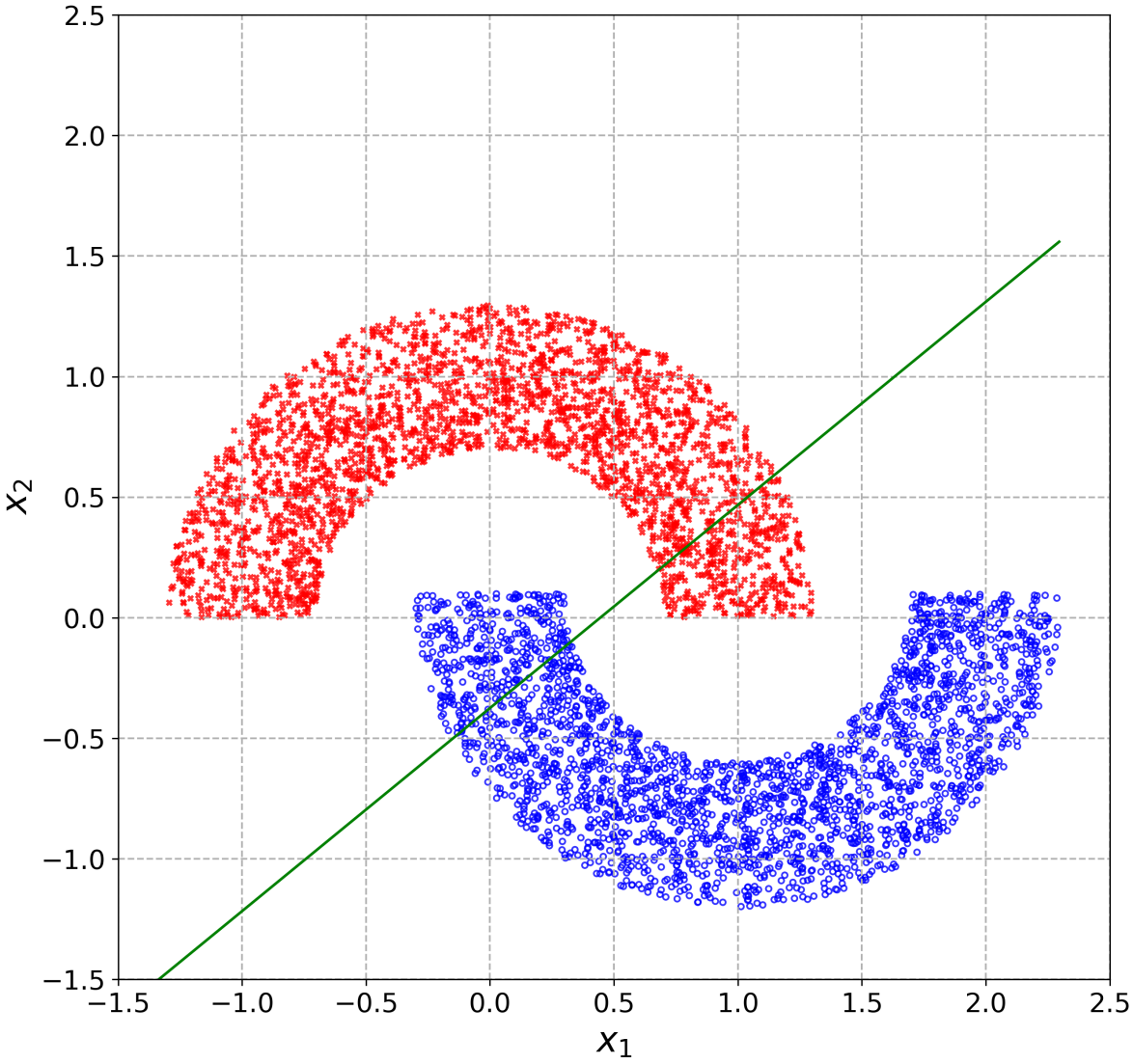
```
plt.xlabel('$x_1$',fontsize=20)
plt.ylabel('$x_2$',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
# fig.savefig('p3_b1.svg',format='svg')


fig2=plt.figure(figsize=[10,10],dpi=300)
plt.hist(y[T[:,0]==1],7,alpha=0.3, align='mid',rwidth=0.8,facecolor='r',edgeco
lor='black',linewidth=2)
plt.hist(y[T[:,1]==1],15,alpha=0.3, align='mid',rwidth=0.8,facecolor='b',edgec
olor='black',linewidth=2)
plt.xlabel('y',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
# fig2.savefig('p3_b2.svg',format='svg')
```

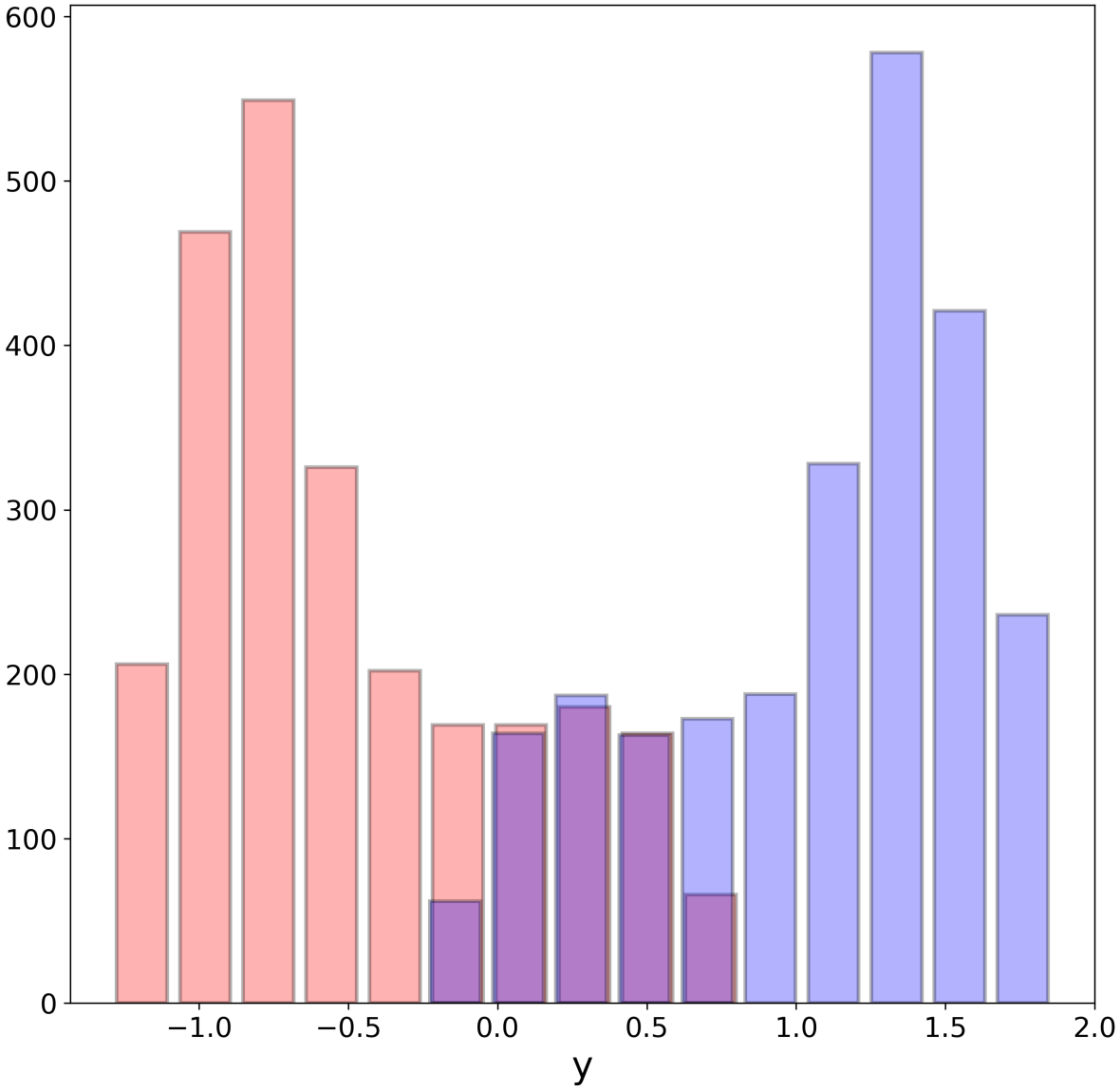

4
2
0
−2
−4
−6
x_2
−2
0
2
4
6
8
10
x_1


80
70
60
50
40
30
20
10
0
−2
0
2
4
6
8
10
y

In [111]:
```
#P3_c)
for name in dir():
    del globals()[name]

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

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.zeros((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=np.ones((1,N//2))

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

N=5000
d=-0.1
r=1
w=0.6
E=doublemoon(N,d,r,w)
E=E.T

X=np.vstack((E[:,0:2],E[:,3:5]))
T=np.vstack((E[:,2,None],E[:,5,None]))

#making hot key coding
T=np.tile(T,(1,int(np.max(T))+1))
for i in np.arange(T.shape[1]):
    T[:,i]-=(i+1)
T=np.equal(T,-1*np.ones((np.shape(T))))*1
T.astype(int)

#calcultating mean of each class
m=np.zeros((X.shape[1],T.shape[1]))
for i in np.arange(T.shape[1]):
    aa=list(map(bool,T[:,i]))
    m[:,i]=np.mean(X[aa,:],axis=0)

w=m[:,1]-m[:,0]
w/=np.linalg.norm(w)

y=X@w
y0=np.mean(y)
C=(y>y0)*1
C=np.reshape(C,(X.shape[0],1))
#making hot key coding
C=np.tile(C,(1,int(np.max(C))+1))
```

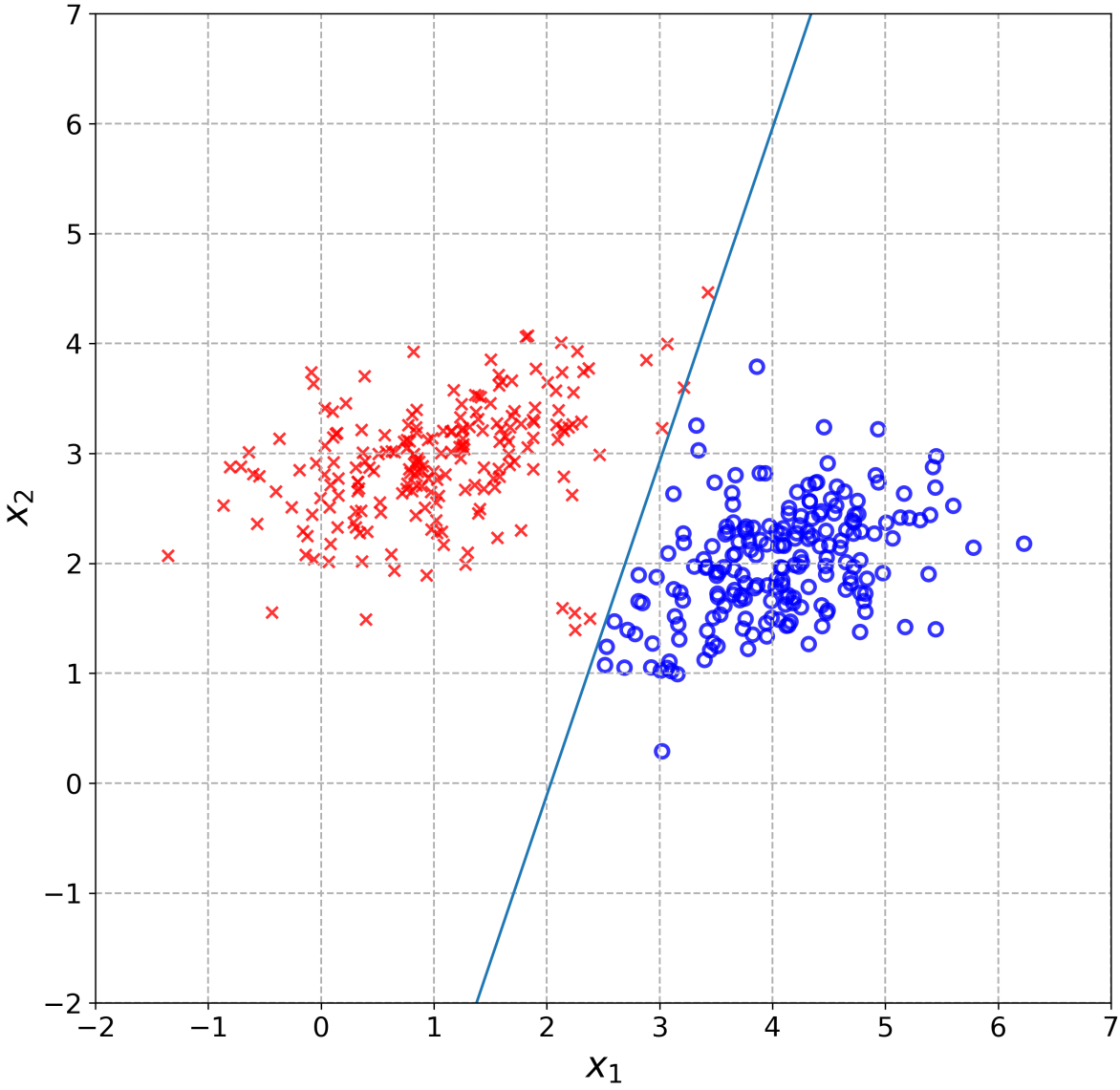
```
for i in np.arange(C.shape[1]):
    C[:,i]-=(i+1)
C=np.equal(C,-1*np.ones((np.shape(C))))*1

#hyper planes
x=np.linspace(np.min(X[:,1]-1),np.max(X[:,1])+1,3)
y1=(w[0]*x-y0)/-w[1]
#indexing data according to lables
i0=np.squeeze(np.equal(T[:,0,None],np.ones((T.shape[0],1))))
i1=np.squeeze(np.equal(T[:,1,None],np.ones((T.shape[0],1))))

#plotting
fig=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(X[i0,0], X[i0,1], alpha=0.8,marker='x', c='r', edgecolors='none',
s=8)
plt.scatter(X[i1,0], X[i1,1], alpha=0.8,marker='o', facecolor='none', edgecolo
rs='blue', s=10, linewidth=1)
plt.plot(x,y1,color='g')
plt.axis([-1.5,2.5,-1.5,2.5])
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()
# fig.savefig('p3_c1.svg',format='svg')


fig2=plt.figure(figsize=[10,10],dpi=300)
plt.hist(y[T[:,0]==1],10,alpha=0.3, align='mid',rwidth=0.8,facecolor='r',edgec
olor='black',linewidth=2)
plt.hist(y[T[:,1]==1],10,alpha=0.3, align='mid',rwidth=0.8,facecolor='b',edgec
olor='black',linewidth=2)
plt.xlabel('y',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
# fig2.savefig('p3_c2.svg',format='svg')
```

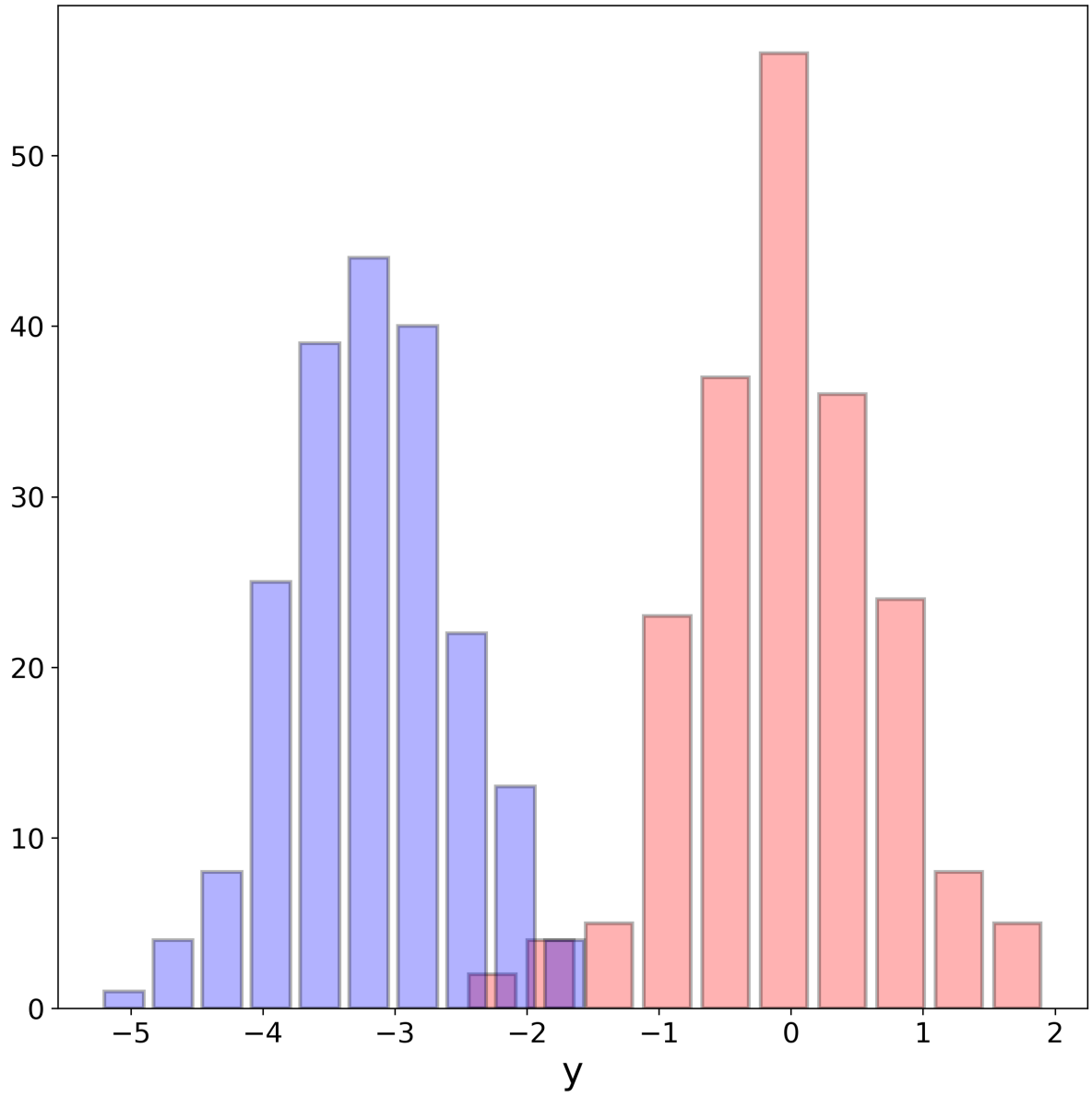

2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
x_2
x_1


600
500
400
300
200
100
0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
y

In [114]:
```
#P4_a)
for name in dir():
    del globals()[name]

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


ad=os.getcwd()
ad=ad+'\data\\'
X=np.loadtxt(ad+'DatasetA_data.csv', unpack='True',delimiter=',').T
T=np.reshape(np.loadtxt(ad+'DatasetA_labels.csv', unpack='True'),(X.shape[0],1
))

#making hot key coding
T=np.tile(T,(1,int(np.max(T))+1))
for i in np.arange(T.shape[1]):
    T[:,i]-=(i+1)
T=np.equal(T,-1*np.ones((np.shape(T))))*1
T.astype(int)

#separating X1 and X2
X1=X[list(map(bool,T[:,0])),:]
X2=X[list(map(bool,T[:,1])),:]
m1=np.mean(X1,axis=0)
m2=np.mean(X2,axis=0)

Sw=np.sum((X1-m1)@(X1-m1).T)+np.sum((X2-m2)@(X2-m2).T)

w=(m2-m1)/Sw
w/=np.linalg.norm(w)

y=X@w
y0=np.mean(y)-w[1]
C=(y>y0)*1
C=np.reshape(C,(X.shape[0],1))
#making hot key coding
C=np.tile(C,(1,int(np.max(C))+1))
for i in np.arange(C.shape[1]):
    C[:,i]-=(i+1)
C=np.equal(C,-1*np.ones((np.shape(C))))*1

#hyper planes
x=np.linspace(np.min(X[:,1]-1),np.max(X[:,1])+1,3)
y1=(w[0]*x-y0)/-w[1]
#indexing data according to lables
i0=np.squeeze(np.equal(C[:,0,None],np.ones((C.shape[0],1))))
i1=np.squeeze(np.equal(C[:,1,None],np.ones((C.shape[0],1))))

#plotting
fig=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(X[i1,0], X[i1,1], alpha=0.8,marker='x', c='r', edgecolors='none',
s=40)
plt.scatter(X[i0,0], X[i0,1], alpha=0.8,marker='o', facecolor='none', edgecolo
rs='blue', s=50, linewidth=2)
```

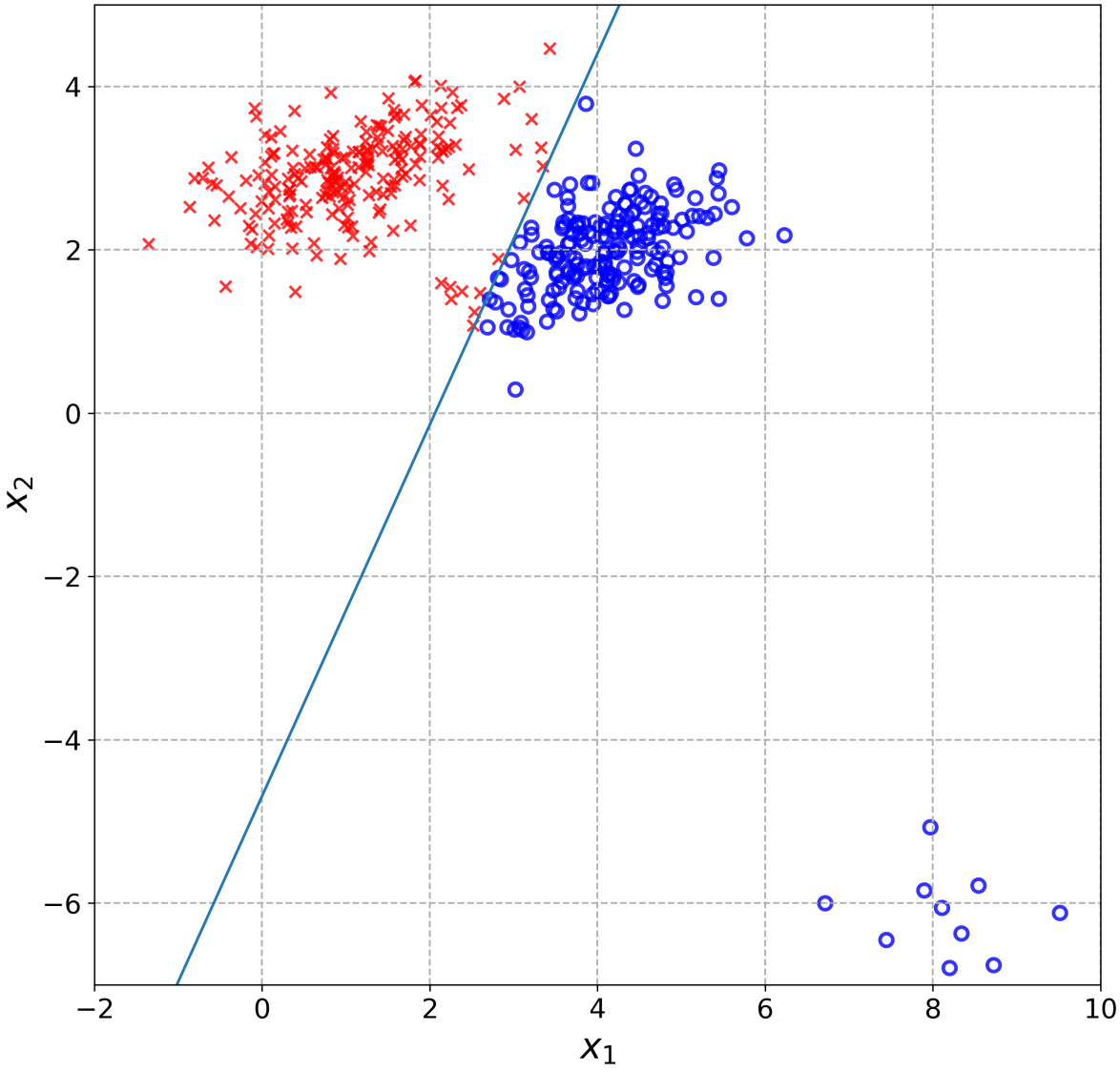
```
plt.plot(x,y1)
plt.axis([-2,7,-2,7])
plt.grid('True',linestyle='--', linewidth=1)
plt.xlabel('$x_1$',fontsize=20)
plt.ylabel('$x_2$',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
# fig.savefig('p4_a1.svg',format='svg')


fig2=plt.figure(figsize=[10,10],dpi=300)
plt.hist(y[T[:,0]==1],10,alpha=0.3, align='mid',rwidth=0.8,facecolor='r',edgec
olor='black',linewidth=2)
plt.hist(y[T[:,1]==1],10,alpha=0.3, align='mid',rwidth=0.8,facecolor='b',edgec
olor='black',linewidth=2)
plt.xlabel('y',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
# fig2.savefig('p4_a2.svg',format='svg')
```

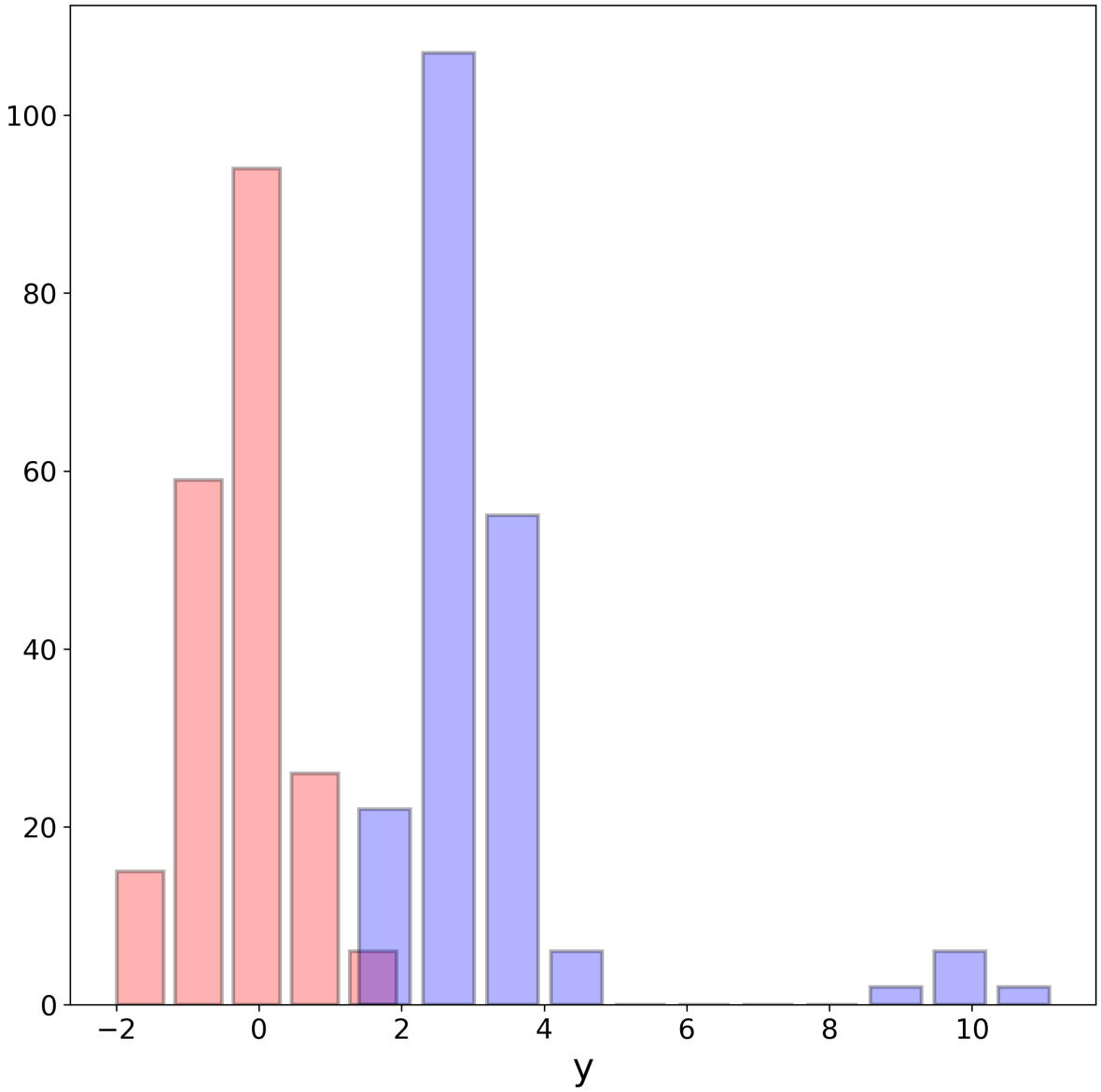

7
6
5
4
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4
5
6
7
x_2
x_1


50
40
30
20
10
0
−5
−4
−3
−2
−1
0
1
2
y

In [115]:
```
#P4_b)
for name in dir():
    del globals()[name]

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

ad=os.getcwd()
ad=ad+'\data\\'
X=np.loadtxt(ad+'DatasetA2_data.csv', unpack='True',delimiter=',').T
T=np.reshape(np.loadtxt(ad+'DatasetA2_labels.csv', unpack='True'),(X.shape[0],
1))

#making hot key coding
T=np.tile(T,(1,int(np.max(T))+1))
for i in np.arange(T.shape[1]):
    T[:,i]-=(i+1)
T=np.equal(T,-1*np.ones((np.shape(T))))*1
T.astype(int)

#separating X1 and X2
X1=X[list(map(bool,T[:,0])),:]
X2=X[list(map(bool,T[:,1])),:]
m1=np.mean(X1,axis=0)
m2=np.mean(X2,axis=0)

Sw=np.sum((X1-m1)@(X1-m1).T)+np.sum((X2-m2)@(X2-m2).T)

w=(m2-m1)/Sw
w/=np.linalg.norm(w)

y=X@w
y0=np.mean(y)-w[1]
C=(y>y0)*1
C=np.reshape(C,(X.shape[0],1))
#making hot key coding
C=np.tile(C,(1,int(np.max(C))+1))
for i in np.arange(C.shape[1]):
    C[:,i]-=(i+1)
C=np.equal(C,-1*np.ones((np.shape(C))))*1

#hyper planes
x=np.linspace(np.min(X[:,1]-1),np.max(X[:,1])+1,3)
y1=(w[0]*x-y0)/-w[1]
#indexing data according to lables
i0=np.squeeze(np.equal(C[:,0,None],np.ones((C.shape[0],1))))
i1=np.squeeze(np.equal(C[:,1,None],np.ones((C.shape[0],1))))

#plotting
fig=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(X[i0,0], X[i0,1], alpha=0.8,marker='x', c='r', edgecolors='none',
s=40)
plt.scatter(X[i1,0], X[i1,1], alpha=0.8,marker='o', facecolor='none', edgecolo
rs='blue', s=50, linewidth=2)
plt.plot(x,y1)
```

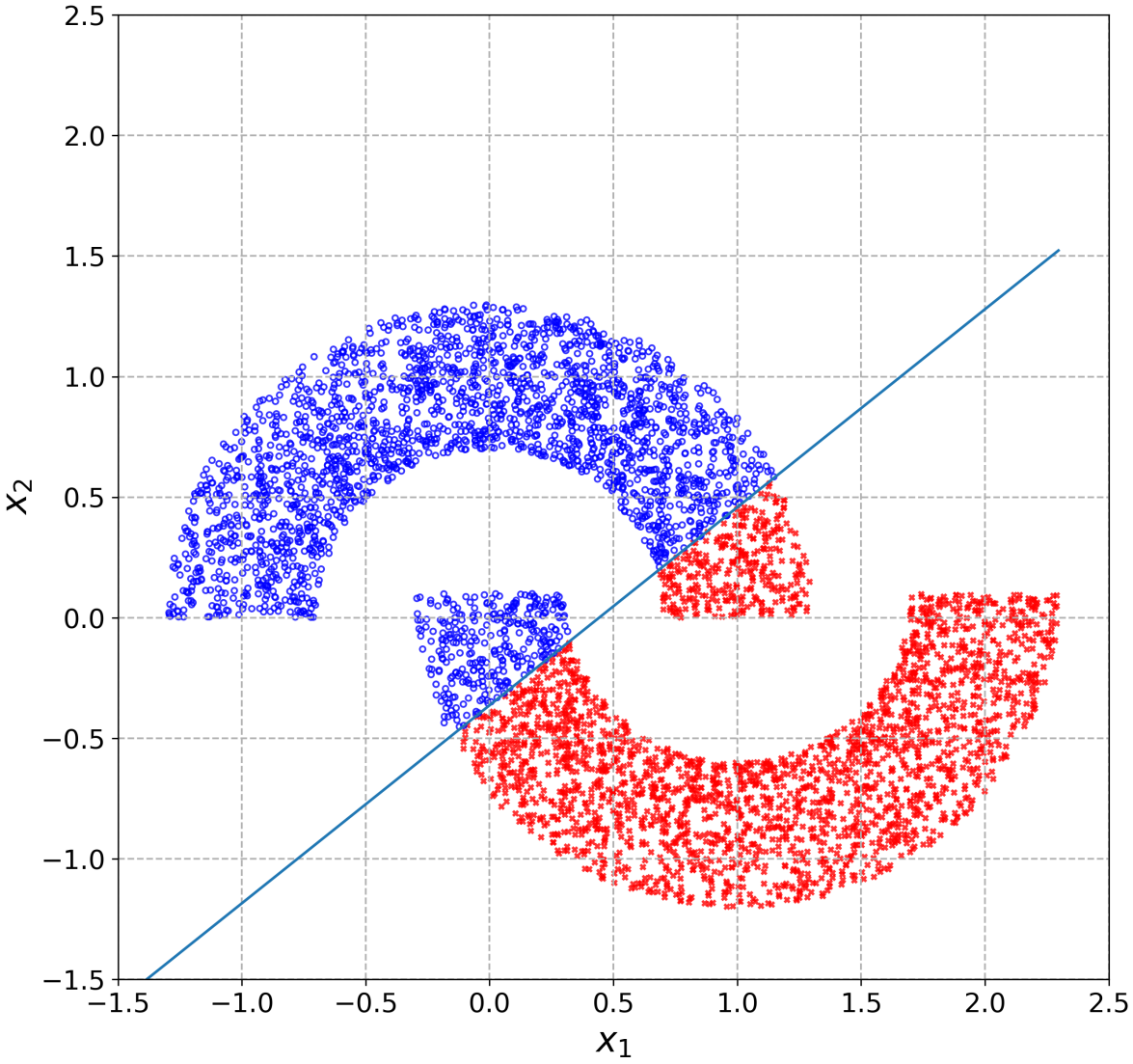
```
plt.axis([-2,10,-7,5])
plt.grid('True',linestyle='--', linewidth=1)
plt.xlabel('$x_1$',fontsize=20)
plt.ylabel('$x_2$',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
# fig.savefig('p4_b1.svg',format='svg')

fig2=plt.figure(figsize=[10,10],dpi=300)
plt.hist(y[T[:,0]==1],5,alpha=0.3, align='mid',rwidth=0.8,facecolor='r',edgeco
lor='black',linewidth=2)
plt.hist(y[T[:,1]==1],11,alpha=0.3, align='mid',rwidth=0.8,facecolor='b',edgec
olor='black',linewidth=2)
plt.xlabel('y',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
# fig2.savefig('p4_b2.svg',format='svg')
```

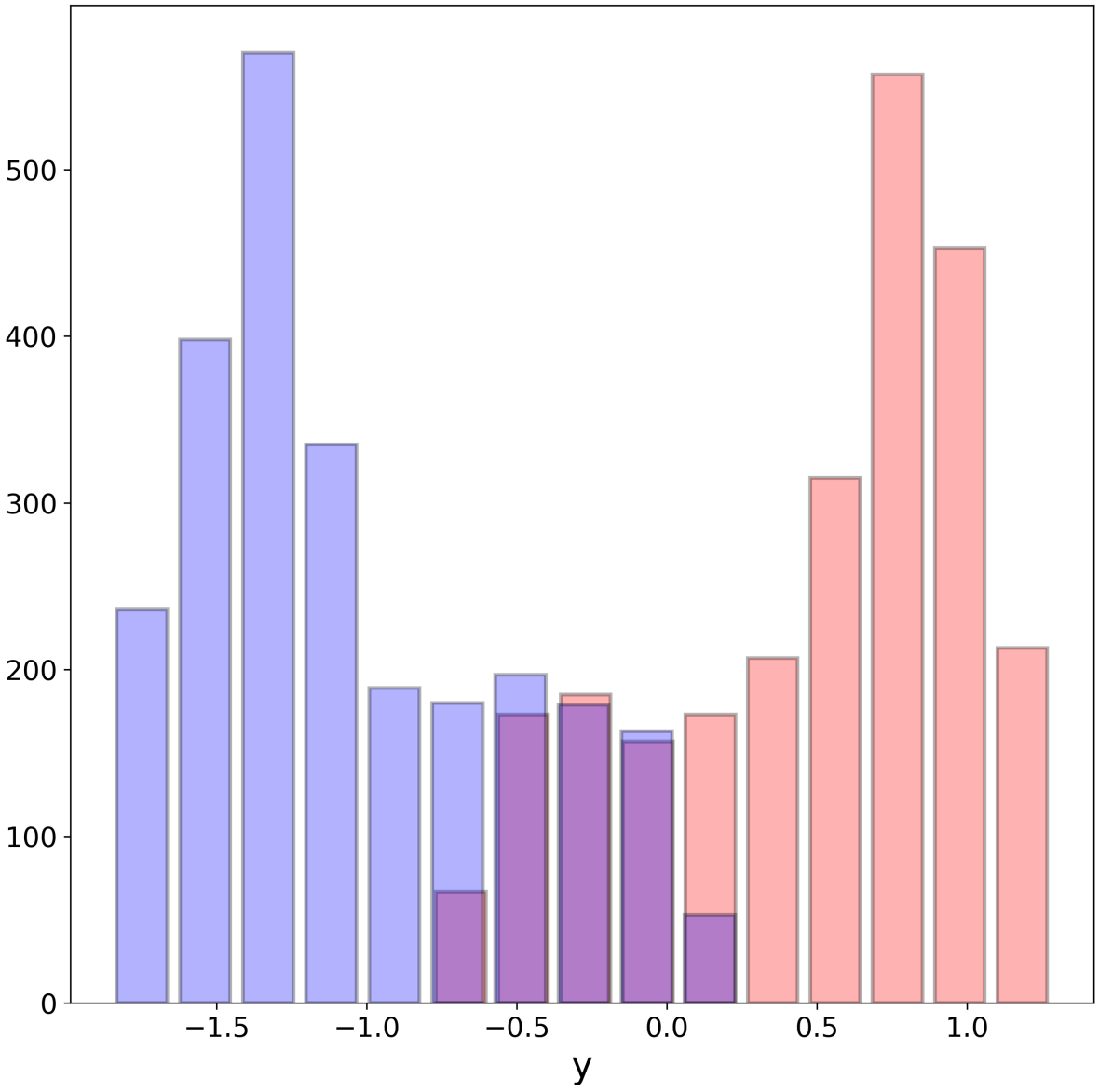

4
2
0
−2
−4
−6
x_2
−2
0
2
4
6
8
10
x_1


100
80
60
40
20
0
−2
0
2
4
6
8
10
y

In [99]:
```
#P4_c)
for name in dir():
    del globals()[name]

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

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.zeros((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=np.ones((1,N//2))

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

N=5000
d=-0.1
r=1
w=0.6
E=doublemoon(N,d,r,w)
E=E.T

X=np.vstack((E[:,0:2],E[:,3:5]))
T=np.vstack((E[:,2,None],E[:,5,None]))
#making hot key coding
T=np.tile(T,(1,int(np.max(T))+1))
for i in np.arange(T.shape[1]):
    T[:,i]-=(i+1)
T=np.equal(T,-1*np.ones((np.shape(T))))*1
T.astype(int)

#separating X1 and X2
X1=X[list(map(bool,T[:,0])),:]
X2=X[list(map(bool,T[:,1])),:]
m1=np.mean(X1,axis=0)
m2=np.mean(X2,axis=0)

Sw=np.sum((X1-m1)@(X1-m1).T)+np.sum((X2-m2)@(X2-m2).T)

w=(m2-m1)/Sw
w/=np.linalg.norm(w)

y=X@w
y0=np.mean(y)
C=(y>y0)*1
C=np.reshape(C,(X.shape[0],1))
#making hot key coding
```

```
C=np.tile(C,(1,int(np.max(C))+1))
for i in np.arange(C.shape[1]):
    C[:,i]-=(i+1)
C=np.equal(C,-1*np.ones((np.shape(C))))*1

#hyper planes
x=np.linspace(np.min(X[:,1]-1),np.max(X[:,1])+1,3)
y1=(w[0]*x-y0)/-w[1]
#indexing data according to lables
i0=np.squeeze(np.equal(C[:,0,None],np.ones((C.shape[0],1))))
i1=np.squeeze(np.equal(C[:,1,None],np.ones((C.shape[0],1))))

#plotting
fig=plt.figure(figsize=[10,10],dpi=300)
plt.scatter(X[i0,0], X[i0,1], alpha=0.8,marker='x', c='r', edgecolors='none',
s=8)
plt.scatter(X[i1,0], X[i1,1], alpha=0.8,marker='o', facecolor='none', edgecolo
rs='blue', s=10, linewidth=1)
plt.plot(x,y1)
plt.axis([-1.5,2.5,-1.5,2.5])
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()
# fig.savefig('p4_c1.svg',format='svg')

fig2=plt.figure(figsize=[10,10],dpi=300)
plt.hist(y[T[:,0]==1],10,alpha=0.3, align='mid',rwidth=0.8,facecolor='r',edgec
olor='black',linewidth=2)
plt.hist(y[T[:,1]==1],10,alpha=0.3, align='mid',rwidth=0.8,facecolor='b',edgec
olor='black',linewidth=2)
plt.xlabel('y',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
# fig2.savefig('p4_c2.svg',format='svg')
```


2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
x_2
x_1



In [ ]: