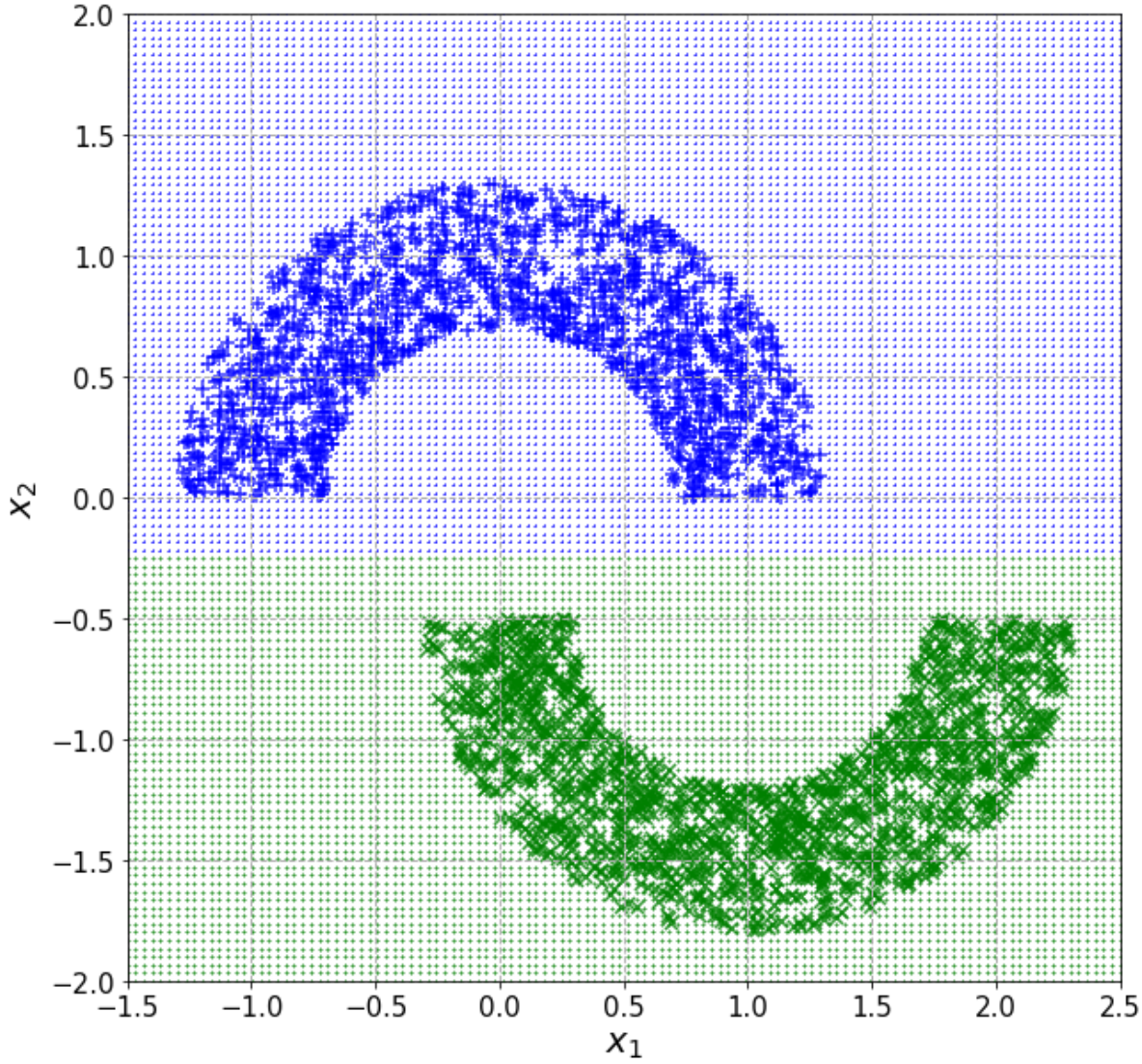
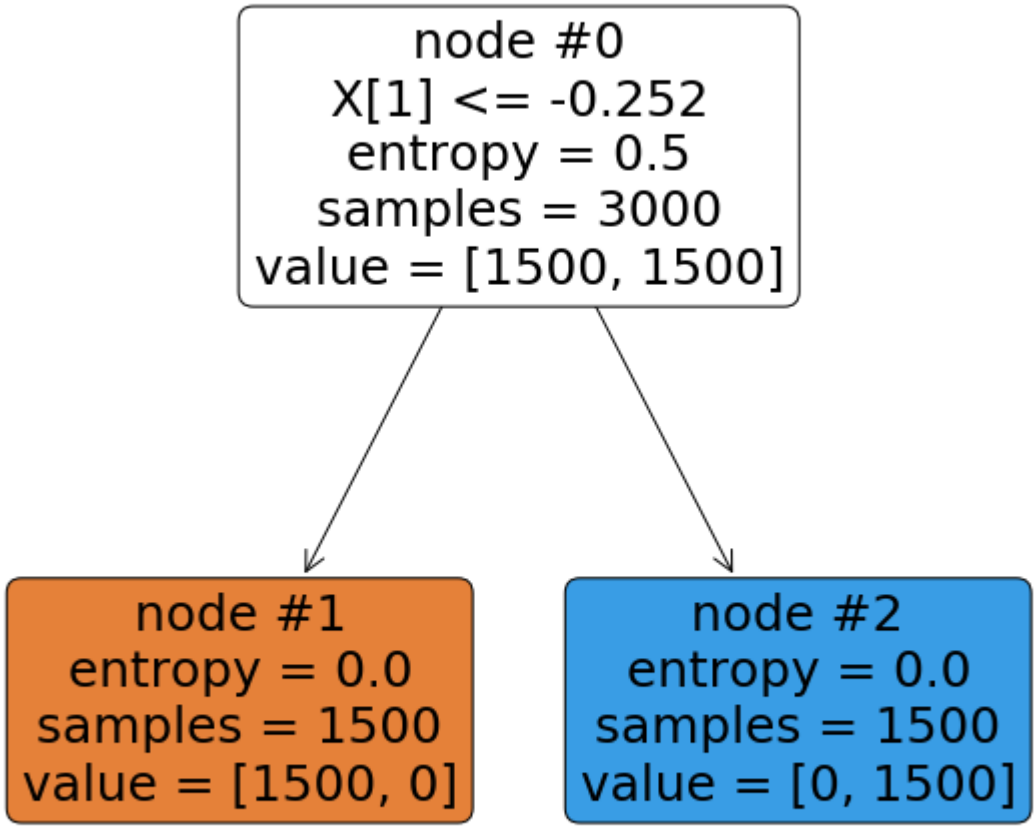
In [11]:
```
#P1_b
for name in dir():
    del globals()[name]

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

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=3000
d=0.5
r=1
w=0.6
K=5

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

X=E[:,0:2]
Y=E[:,2]

clf=tree.DecisionTreeClassifier()
clf=clf.fit(X, Y)

nn=120
nodes1=np.linspace(-1.5,2.5,nn)
nodes2=np.linspace(-2.0,2.0,nn)
x1, x2 = np.meshgrid(nodes1, nodes2)
NodeTag=np.zeros((nn,nn))
crd=np.stack((x1,x2),axis=2)
crd=np.reshape(crd,(nn**2,2),order='C')

l=clf.predict(crd)

fig1=plt.figure(figsize=[10,10])
plt.scatter(X[Y==1,0], X[Y==1,1],s=80, alpha=0.8, c='blue', edgecolors='none',
marker="+",label='Class A')
plt.scatter(X[Y==-1,0], X[Y==-1,1],s=50, alpha=0.8, c='green', edgecolors='non
e', marker="x",label='Class B')
plt.scatter(crd[l==1,0], crd[l==1,1],s=1, alpha=0.8, c='blue', edgecolors='non
```

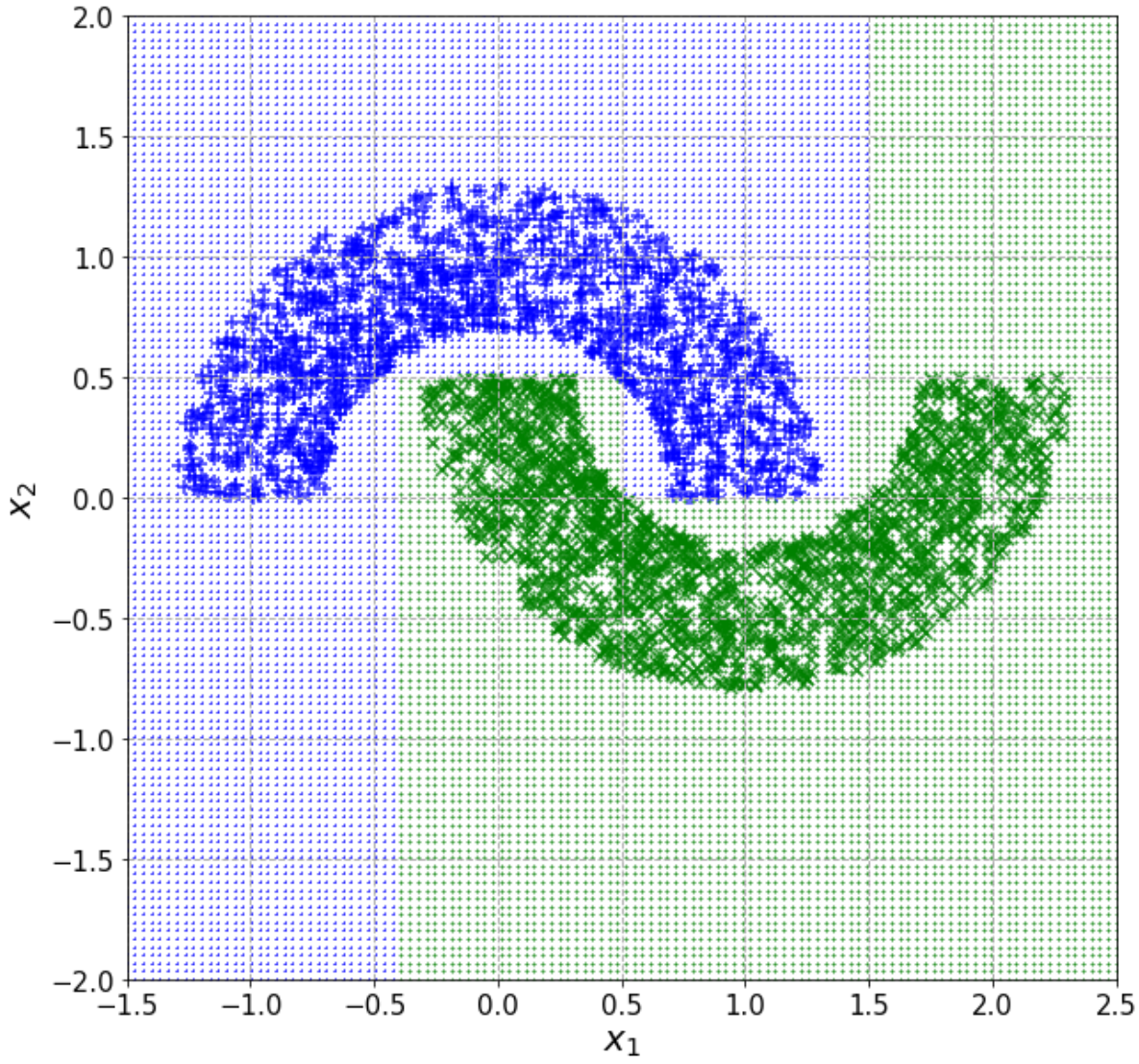
```
e', marker="+")
plt.scatter(crd[l==-1,0], crd[l==-1,1],s=1, alpha=0.8, c='green', edgecolors=
'none', marker="x")
plt.axis([-1.5,2.5,-2,2])
plt.grid('True',linestyle='--', linewidth=1)
plt.xlabel('$x_1$',fontsize=20)
plt.ylabel('$x_2$',fontsize=20)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show(fig1)
# fig1.savefig('p1_b.svg',format='svg')

fig2=plt.figure(figsize=[10,10])
tree.plot_tree(clf,filled=True,impurity=True,node_ids=True,rotate=True,rounded
=True)
plt.show(fig2)
# fig2.savefig('p1_b2.svg',format='svg')
```

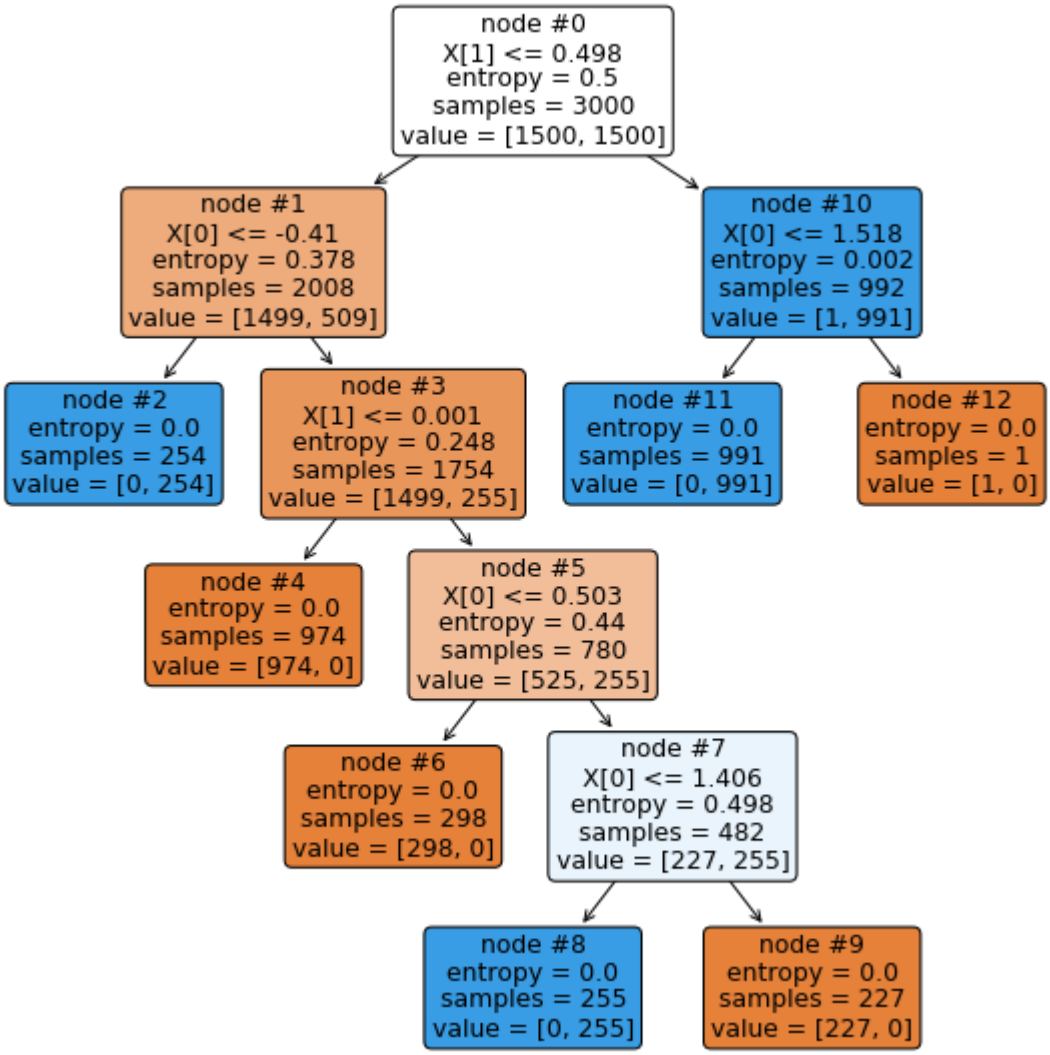

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
x_1
x_2


node #0
X[1] <= -0.252
entropy = 0.5
samples = 3000
value = [1500, 1500]
node #1
entropy = 0.0
samples = 1500
value = [1500, 0]
node #2
entropy = 0.0
samples = 1500
value = [0, 1500]

In [14]:
```
#P1_c
for name in dir():
    del globals()[name]

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

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=3000
d=-0.5
r=1
w=0.6
K=5

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

X=E[:,0:2]
Y=E[:,2]

clf=tree.DecisionTreeClassifier()
clf=clf.fit(X, Y)

nn=120
nodes1=np.linspace(-1.5,2.5,nn)
nodes2=np.linspace(-2.0,2.0,nn)
x1, x2 = np.meshgrid(nodes1, nodes2)
NodeTag=np.zeros((nn,nn))
crd=np.stack((x1,x2),axis=2)
crd=np.reshape(crd,(nn**2,2),order='C')

l=clf.predict(crd)

fig1=plt.figure(figsize=[10,10])
plt.scatter(X[Y==1,0], X[Y==1,1],s=80, alpha=0.8, c='blue', edgecolors='none',
marker="+",label='Class A')
plt.scatter(X[Y==-1,0], X[Y==-1,1],s=50, alpha=0.8, c='green', edgecolors='non
e', marker="x",label='Class B')
plt.scatter(crd[l==1,0], crd[l==1,1],s=1, alpha=0.8, c='blue', edgecolors='non
```

```
e', marker="+")
plt.scatter(crd[l==-1,0], crd[l==-1,1],s=1, alpha=0.8, c='green', edgecolors=
'none', marker="x")
plt.axis([-1.5,2.5,-2,2])
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)
# fig1.savefig('p1_c.svg',format='svg')

fig2=plt.figure(figsize=[10,10])
tree.plot_tree(clf,filled=True,impurity=True,node_ids=True,rotate=True,rounded
=True)
plt.show(fig2)
# fig2.savefig('p1_c2.svg',format='svg')
```


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



In [18]:
```
print(l)
```

```
[-1. -1. -1. ...  1.  1.  1.]
```

In [ ]:
```

```