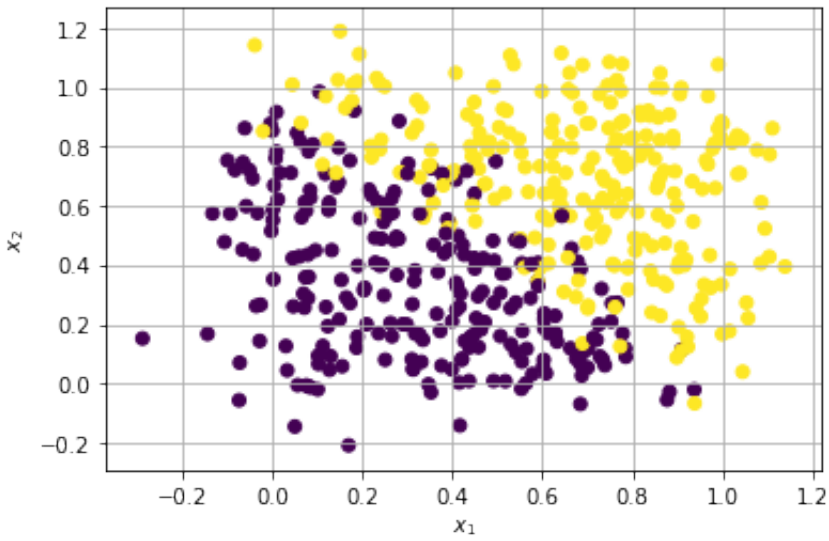
Q1

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import pickle

# Load the dataset
with open('two_class_dataset.pkl', 'rb') as f:
    X, y = pickle.load(f)

# Create a scatter plot the datapoints and assign a color based on the
# note X has three rows (the first is a row of 1)
plt.scatter(X[1,:], X[2,:] , c=y)
plt.xlabel('$x_1$')
plt.ylabel('$x_2$')
plt.grid()
plt.show()
```



In [110]:
```
# w (1, 3)
print(X[:, 0].reshape(-1, 1).shape)
# print(_w.shape)
```

```
(3, 1)
```

In [119]:
```
def nll(_w):
    """ compute the logistic loss (which is equal to the negative log
    given a numpy array X with each column a datapoint, a list of labe
    and a wieght vector w.  Returns a scalar.
    """
    assert np.shape(X)[1]== len(y)
    assert np.shape(X)[0]==np.shape(_w)[0]
    _nnl = 0
    for i in range(len(y)):
        _nnl+= - np.log(1 + np.exp(y[i]*X[:, i].reshape(1, -1) @ _w))
    return _nnl

def grad(_w):
    """ compute the gradient of the logistic loss given a numpy array
    column a datapoint, a list of labels, and a wieght vector w. Retu
    """
    assert np.shape(X)[1]== len(y)
    assert np.shape(X)[0]==np.shape(_w)[0]
    _grad_nnl = 0
    for i in range(len(y)):
        _grad_nnl+= -y[i] / (1 + np.exp(y[i] *(_w).T @ X[:, i].reshape
    # print(np.shape(_grad_nnl))
    return _grad_nnl
```

In [121]:
```
max_its = 300
tau =.03  # convergence is sensitive to step size

w = np.zeros((3,1))  ## pick a random starting point

## run gradient descent
w_new  = w - tau*grad(w)
it = 1

while np.linalg.norm(w-w_new) > .001 and it < max_its :
    w = w_new
    w_new  = w - tau*grad(w)
    it += 1
    # print stats on every tenth iteration
    if it%10 == 0:
        print('iteration:', it, ',  objective value:', nll(w))

print(w)
print(grad(w))
```

```
iteration: 10 ,  objective value: [[-1658.13632598]]
iteration: 20 ,  objective value: [[-1911.26419761]]
iteration: 30 ,  objective value: [[-1821.71683109]]
iteration: 40 ,  objective value: [[-1842.00289472]]
```

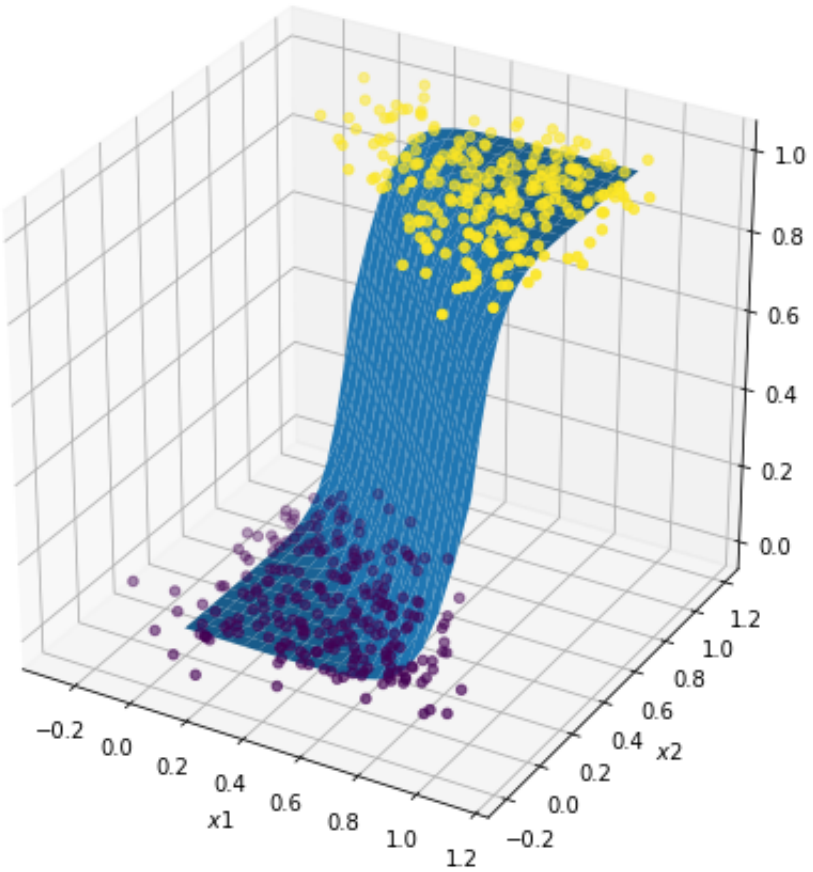
```
iteration: 50 ,  objective value: [[-1882.1726068]]
iteration: 60 ,  objective value: [[-1921.01180038]]
iteration: 70 ,  objective value: [[-1956.34716676]]
iteration: 80 ,  objective value: [[-1988.48209356]]
iteration: 90 ,  objective value: [[-2017.81848975]]
iteration: 100 ,  objective value: [[-2044.69425107]]
iteration: 110 ,  objective value: [[-2069.39140019]]
iteration: 120 ,  objective value: [[-2092.14777659]]
iteration: 130 ,  objective value: [[-2113.16591774]]
iteration: 140 ,  objective value: [[-2132.61979483]]
iteration: 150 ,  objective value: [[-2150.65999637]]
iteration: 160 ,  objective value: [[-2167.41777274]]
iteration: 170 ,  objective value: [[-2183.00823152]]
iteration: 180 ,  objective value: [[-2197.53289081]]
iteration: 190 ,  objective value: [[-2211.08174049]]
iteration: 200 ,  objective value: [[-2223.73492209]]
iteration: 210 ,  objective value: [[-2235.56410947]]
iteration: 220 ,  objective value: [[-2246.63365289]]
iteration: 230 ,  objective value: [[-2257.0015337]]
iteration: 240 ,  objective value: [[-2266.72016656]]
iteration: 250 ,  objective value: [[-2275.83707772]]
iteration: 260 ,  objective value: [[-2284.39548191]]
iteration: 270 ,  objective value: [[-2292.43477568]]
iteration: 280 ,  objective value: [[-2299.99096136]]
iteration: 290 ,  objective value: [[-2307.0970133]]
iteration: 300 ,  objective value: [[-2313.78319554]]
[[-12.52406529]
 [ 12.7059725 ]
 [ 12.43331915]]
[[ 0.12206812]
 [-0.12448892]
 [-0.12057335]]
```

In [122]:
```
%matplotlib inline
from mpl_toolkits.mplot3d import axes3d

# Plot the logistic surface and the data points
xx = np.linspace(-0,1,20)
yy = np.linspace(-0,1,20)
XX,YY = np.meshgrid(xx,yy)
ZZ = 1/(1+np.exp(-(w[0]+XX*w[1]+YY*w[2])))

fig = plt.figure(figsize=(8,8))
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(XX, YY, ZZ)
ax.scatter(X[1,:], X[2,:], (np.array(y)+1)/2, c=y)
plt.xlabel('$x1$')
plt.ylabel('$x2$')
```

Out[122]: Text(0.5, 0, '$x2$')



Q2

a

In [ ]: