In [9]:
```
import pandas as pd
import numpy as np
```

In [11]:
```
#groupby 1 or 0
def group(data):

    d = {}
    ones = []
    zeros = []
    for i in range(len(data)):
        if data[i][-1] == 0:
            zeros.append(data[i])
        else:
            ones.append(data[i])
    d[1] = ones
    d[0] = zeros
    return d

#finds mean and std for each feature grouped by y value
def meanstdby_y(data):

    sep = group(data)
    meanstds = {}
    for y, row in sep.items():
        meanstds[y] = meanstdhelper(row)
    return meanstds


#calculates p(x) for each feature separated by y value, for a single x vector
def gaussby_y(d, test):

    p = {}
    for y, meanstd in d.items():
        p[y] = 1
        for i in range(len(meanstd)):
            mean, std = meanstd[i]
            x = test[i]
            p[y] *= gausshelper(x, mean, std)
    return p


#collects y predictions for test x vectors
def predict(d, x_test):

    l = []
    for i in range(len(x_test)):
        result = predicthelper(d, x_test[i])
        l.append(result)
    return l
```

In [18]:
```
#_____________________________________ helper funcs
#predicts y for a single x vector
def predicthelper(d, test):

    yL = 0
    p = 0
    for y, prob in gaussby_y(d, test).items():
        if prob > p:
            p = prob
            yL = y
    return yL

#gaussian function calculator
def gausshelper(x, mean, std):

    denom = 1/(np.sqrt(2*np.pi)*std)
    num = np.power(np.e,(-1/2)*(x-mean)*(x-mean)/np.power(std,2))
    final = denom*num
    return final

#finds mean and std for each feature, dropping the y column
def meanstdhelper(data):

    meanstds = [(np.mean(feature), np.std(feature)) for feature in zip(*data)]
    return meanstds[:6]
```

In [19]:
```
train = pd.read_csv("train-data.csv",header = None)
trainset = train.to_numpy()
test = pd.read_csv("test-data.csv", header = None)
testset = test.to_numpy()
```

In [20]:
```
trainset.shape, testset.shape
```

Out[20]:
```
((24420, 7), (8141, 6))
```

In [*]:
```
#percentage of predictions i got right in the training set
d = meanstdby_y(trainset) #getting the data dict from the training set
np.mean(predict(d,trainset) == train[[6]].to_numpy().flatten())
```

In [15]:
```
predictions = predict(d, testset)
len(predictions)
```

Out[15]:
```
8141
```

In [17]:
```
np.savetxt("predictions.csv", predictions, delimiter =", ")
```

In [ ]:

In [ ]:

In [ ]: