

CS589 ASSIGNMENT 2

Name: Dorian Benhamou Goldfajn

Email: dbenhamougol@umass.edu

Discussed With: Aryan Nair

```
In [21]: import pangolin as pg
```

Question 1

```
In [22]: from scipy.stats import norm
```

```
In [23]: x = [1, 2, 3, 4, 5, 6]
y = [0, 1.5, 1.0, 0.5, 1.2, 2.5]
slopes_values = [0, 0.25, 0.5, 0.75, 1]

def Probability_Y_Given_S(slope_value, x, y):
    y_equation = slope_value * x
    prob_y = norm.pdf(y, loc=y_equation, scale=1)
    return prob_y

probs_of_all_s = [1 for _ in range(len(slopes_values))]
for i, s in enumerate(slopes_values):
    for j in range(len(x)):
        cur_x, cur_y = x[j], y[j]
        prob_y_given_s = Probability_Y_Given_S(s, cur_x, cur_y)
        probs_of_all_s[i] *= prob_y_given_s

posterior = [.2 * p for p in probs_of_all_s]
sm = sum(posterior)
normal_posterior = [p / sm for p in posterior]
normal_posterior
```

```
Out[23]: [0.009456905156440607,
0.7751024227873098,
0.21523816927002018,
0.00020250214073843964,
6.454908376547967e-10]
```

Question 2

```
In [26]: inputs = [1, 2, 3, 4, 5, 6]
outputs = [0, 1.5, 1.0, 0.5, 1.2, 2.5]

w = pg.categorical([.2,.2,.2,.2,.2])
s_values = pg.makerv([0, 0.25, 0.5, 0.75, 1])
s_floats = [0, 0.25, 0.5, 0.75, 1]
noise = pg.normal(0, 1)

y = [pg.normal(s_values[w] * x, 1) for x in inputs]
```

```
w_samples, n_samples = pg.sample((w, noise), y, outputs)
w_samples = [.25 * w for w in w_samples]

probs = [0 for i in range(len(s_values))]
for j, s in enumerate(w_samples):
    for i, sval in enumerate(s_floats):
        if s == sval:
            probs[i] += 1

probs = [p / len(w_samples) for p in probs]

posterior = [p * .2 for p in probs]

s = sum(posterior)
normal_posterior = [p / s for p in posterior]
normal_posterior
```

Out[26]: [0.01129999999999998, 0.7761, 0.2122, 0.0003999999999999996, 0.0]

Question 3

In [27]: `import matplotlib.pyplot as plt`

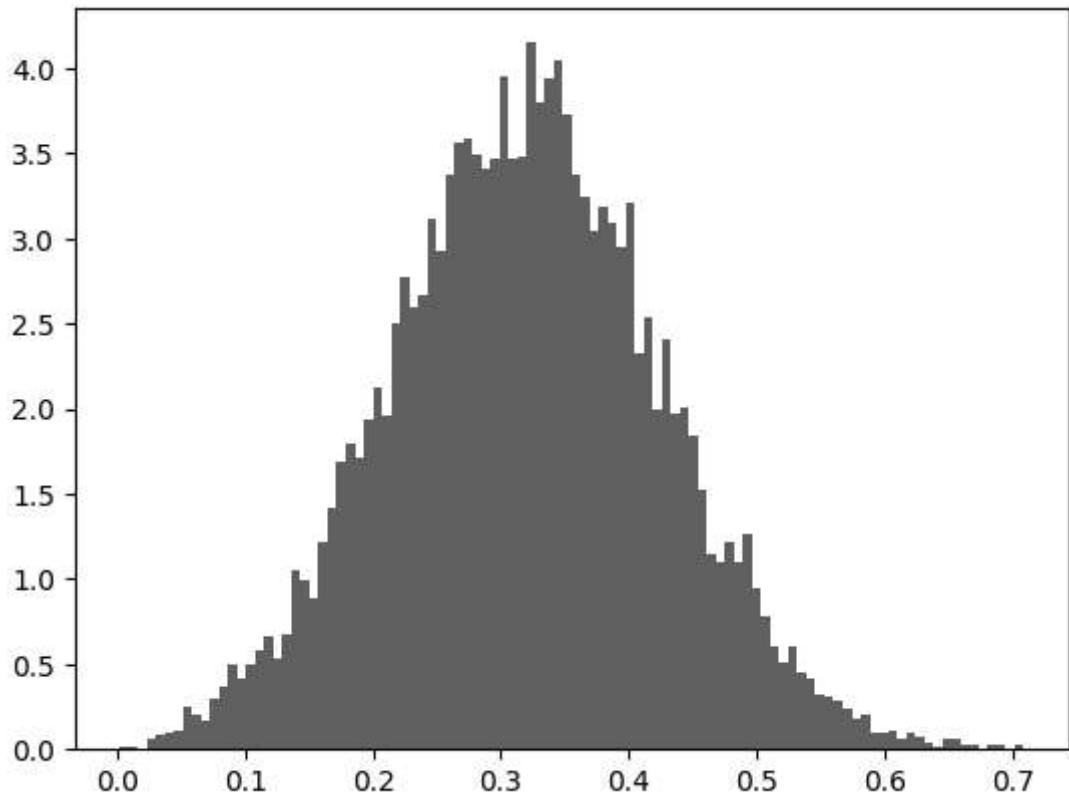
In [28]: `inputs = [1, 2, 3, 4, 5, 6]`
`outputs = [0, 1.5, 1.0, 0.5, 1.2, 2.5]`

```
s = pg.uniform(0, 1)
s_floats = [i * .1 for i in range(0, 10)]
noise = pg.normal(0, 1)
```

```
y = [pg.normal(s * x, 1) for x in inputs]
```

```
s_samples, n_samples = pg.sample((s, noise), y, outputs)
plt.hist(s_samples, density=True, bins=100)
```

```
Out[28]: (array([0.01415681, 0.01415681, 0.          , 0.05662723, 0.08494085,
   0.09909766, 0.11325447, 0.25482255, 0.19819532, 0.1698817 ,
   0.29729298, 0.36807702, 0.4954883 , 0.41054745, 0.4954883 ,
   0.58042915, 0.66537 , 0.53795872, 0.67952752, 1.04760272,
   0.99097764, 0.89187799, 1.21748681, 1.41567935, 1.68466198,
   1.79791278, 1.71297563, 1.93948071, 2.12352351, 1.95363751,
   2.50575774, 2.77473153, 2.59069868, 2.66147718, 3.11450114,
   2.93046244, 3.36931686, 3.56751197, 3.58166876, 3.49674275,
   3.41178724, 3.46841441, 3.94974539, 3.46842905, 3.48257121,
   4.1479405 , 3.79402066, 3.9356052 , 4.04884295, 3.7232367 ,
   3.36931686, 3.24191939, 3.04371061, 3.18527854, 3.08619401,
   2.94461305, 3.21359213, 2.32171414, 2.53407673, 1.99610789,
   2.4066549 , 1.9677943 , 2.01027316, 1.84038316, 1.5289337 ,
   1.14670028, 1.10423455, 1.21748424, 1.10422989, 1.25995462,
   0.94850517, 0.77862364, 0.60874726, 0.50964457, 0.60874212,
   0.45301739, 0.41054701, 0.32560625, 0.31144946, 0.28313587,
   0.24066752, 0.18403832, 0.19819511, 0.09909755, 0.09909755,
   0.11325435, 0.05662717, 0.09909755, 0.07078456, 0.04247038,
   0.01415679, 0.05662717, 0.05662717, 0.02831359, 0.02831359,
   0.          , 0.02831383, 0.02831359, 0.          , 0.02831359]),
array([0.00215142, 0.00921515, 0.01627889, 0.02334263, 0.03040637,
   0.03747011, 0.04453385, 0.05159759, 0.05866133, 0.06572507,
   0.0727888 , 0.07985254, 0.08691628, 0.09398002, 0.10104376,
   0.1081075 , 0.11517124, 0.12223498, 0.12929872, 0.13636245,
   0.14342619, 0.15048993, 0.15755367, 0.1646174 , 0.17168115,
   0.17874488, 0.18580863, 0.19287236, 0.19993611, 0.20699984,
   0.21406358, 0.22112732, 0.22819106, 0.23525479, 0.24231854,
   0.24938227, 0.256446 , 0.26350975, 0.2705735 , 0.27763724,
   0.28470096, 0.29176471, 0.29882845, 0.3058922 , 0.31295592,
   0.32001966, 0.32708341, 0.33414716, 0.34121087, 0.34827462,
   0.35533836, 0.36240211, 0.36946583, 0.37652957, 0.38359332,
   0.39065704, 0.39772078, 0.40478453, 0.41184828, 0.41891199,
   0.42597574, 0.43303949, 0.44010323, 0.44716695, 0.4542307 ,
   0.46129444, 0.46835819, 0.47542191, 0.48248565, 0.4895494 ,
   0.49661314, 0.50367689, 0.51074064, 0.51780432, 0.52486807,
   0.53193182, 0.53899556, 0.54605931, 0.55312306, 0.5601868 ,
   0.56725055, 0.57431424, 0.58137798, 0.58844173, 0.59550548,
   0.60256922, 0.60963297, 0.61669672, 0.62376046, 0.63082415,
   0.6378879 , 0.64495164, 0.65201539, 0.65907913, 0.66614288,
   0.67320663, 0.68027037, 0.68733406, 0.69439781, 0.70146155,
   0.7085253 ]),
<BarContainer object of 100 artists>)
```

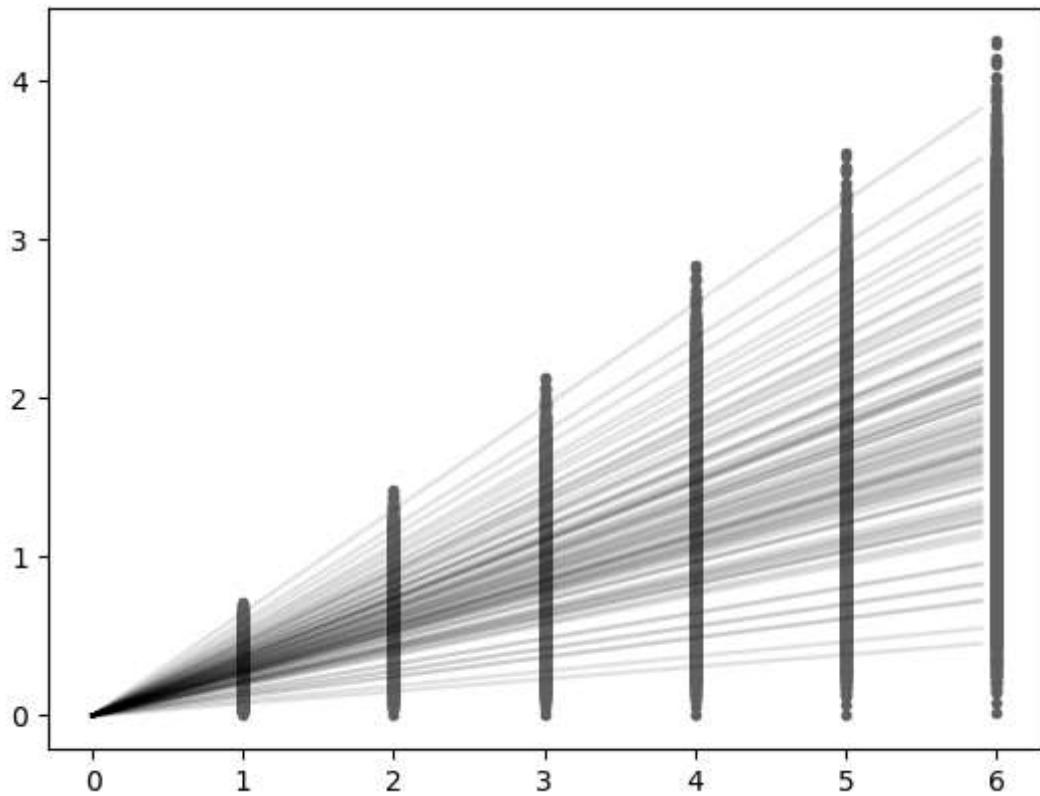


Question 4

```
In [29]: import numpy as np

x,y = [x for s in s_samples for x in inputs], [s * x for s in s_samples for x in in

plt.plot(x,y,'.')
for s in s_samples[::100]:
    xs = np.arange(0,6,.1)
    plt.plot(xs, s*xs,'k-',alpha=0.1)
```



Question 5

```
In [30]: inputs = [1, 2, 3, 4, 5, 6]
outputs = [0, 1.5, 1.0, 0.5, 1.2, 2.5]

s = pg.uniform(0, 1)
s_floats = [0, 0.25, 0.5, 0.75, 1]
noise = pg.normal(0, 1)
bias = pg.uniform(-1,1)

y = [pg.normal(bias + s * x, 1) for x in inputs]

s_samples, n_samples, bias_samples = pg.sample((s, noise, bias), y, outputs)

probs = [0 for i in range(len(s_floats))]
for j, s in enumerate(s_samples):
    for i, sval in enumerate(s_floats):
        if s <= sval + .25 and s >= sval:
            probs[i] += 1

probs = [p / len(s_samples) for p in probs]

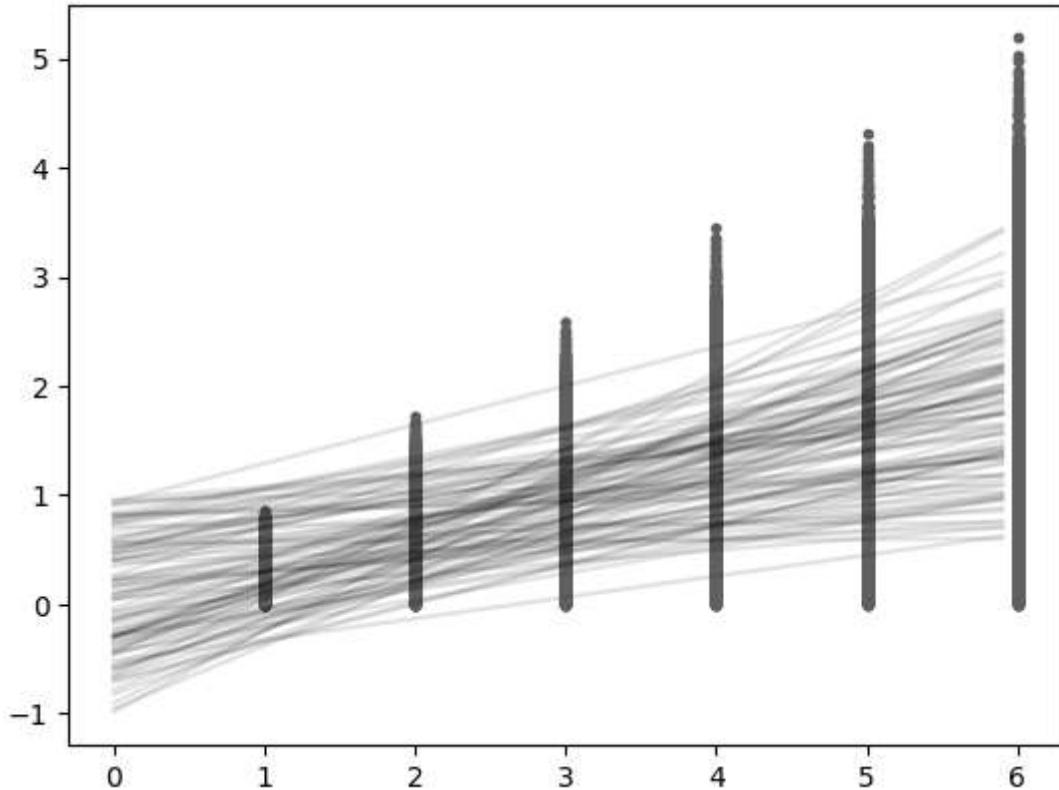
posterior = [p * .2 for p in probs]

s = sum(posterior)
normal_posterior = [p / s for p in posterior]
normal_posterior
```

```

x,y = [x for s in s_samples for x in inputs], [s * x for s in s_samples for x in inputs]
plt.plot(x,y,'.')
for (b, s) in zip(bias_samples[::100], s_samples[::100]):
    xs = np.arange(0,6,.1)
    plt.plot(xs, b + s*xs,'k-',alpha=0.1)

```



Question 6

```

In [31]: import numpy as np
from sklearn.datasets import load_diabetes
X,y = load_diabetes(return_X_y=True)
y = y - np.mean(y)
y = y / np.std(y)

X_train = X[1::2]
X_test = X[::2]
y_train = y[1::2]
y_test = y[::2]

num_train, num_features = X_train.shape
num_test = X_test.shape[0]

X_train = pg.makerv(X_train)

dist_y_train = pg.slot()
w = pg.slot()
with pg.Loop(num_features) as i:
    w[i] = pg.normal(0, 10)

```

```

with pg.Loop(len(X_train)) as j:
    x = X_train[j]
    dist_y_train[j] = pg.normal(w @ x, 1)

dist_y_test = [pg.normal(w @ X_test[i], 1) for i in range(len(X_test))]

e = pg.E(dist_y_test, dist_y_train, y_train)
print("mean: ", np.mean(e))
w_samples = pg.sample((dist_y_test), dist_y_train, [y for y in y_train], niter=100)

mean:  0.016898083

```

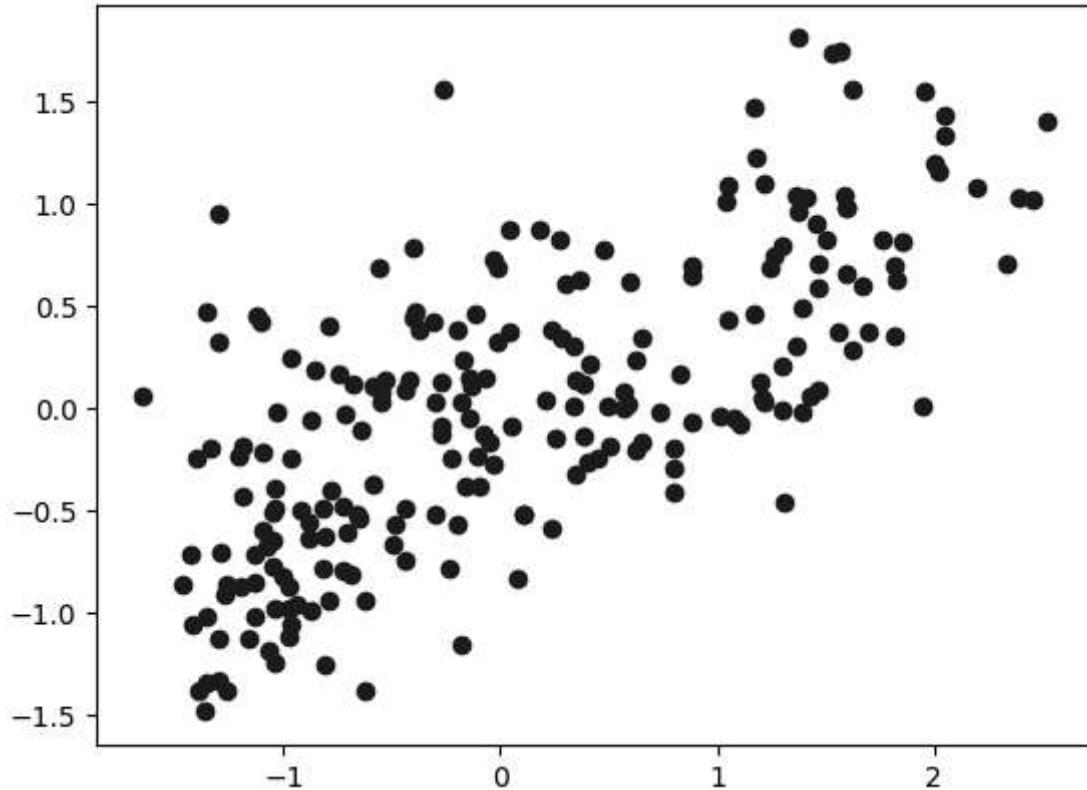
In [32]:

```

import matplotlib.pyplot as plt
X_test = np.array(X_test)
w_samples = np.array([np.median(row) for row in w_samples])
y_test = np.array(y_test)
plt.scatter(y_test, w_samples, color='blue')

```

Out[32]: <matplotlib.collections.PathCollection at 0x2063a03af10>



In [33]:

```

# calculate mean squared difference
sq_diff = (w_samples - y_test) ** 2
mean_sq_diff = np.mean(sq_diff)
mean_sq_diff

```

Out[33]: 0.5697750148423903

Question 7

```
In [ ]: from sklearn.datasets import load_breast_cancer
X,y = load_breast_cancer(return_X_y=True)

X_train = X[1::2]
X_test = X[::2]
y_train = y[1::2]
y_test = y[::2]

num_train, num_features = X_train.shape
num_test = X_test.shape[0]

X_train = pg.makerv(X_train)

w = pg.slot()
with pg.Loop(num_features) as i:
    w[i] = pg.normal(0, 10)

dist_y_train = pg.slot()
with pg.Loop(len(X_train)) as j:
    x = X_train[j]
    dist_y_train[j] = pg.bernoulli(pg.sigmoid(w @ x))

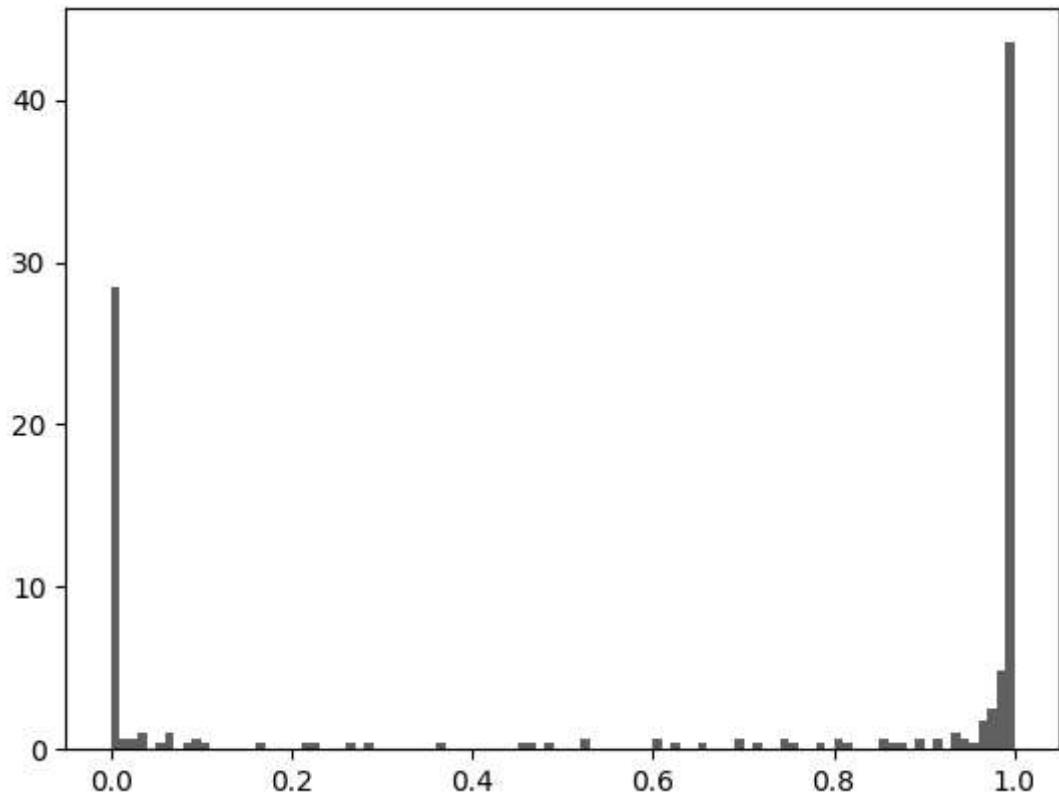
sigmoids = [pg.sigmoid(w @ X_test[i]) for i in range(len(X_test))]
bernoullis = [pg.bernoulli(sigmoids[i]) for i in range(len(sigmoids))]

e= pg.E(bernoullis, dist_y_train, y_train)
print("mean: ", np.mean(e))
zeroes = [expected_val for expected_val in e if expected_val == 0]
ones = [expected_val for expected_val in e if expected_val == 1]
```

```
In [ ]: plt.hist(e, density=True, bins=100)
plt.hist(zeroes, density=True, bins=100)
plt.hist(ones, density=True, bins=100)
```

```
In [17]: plt.hist(e, density=True, bins=100)
```

```
Out[17]: (array([28.42105327,  0.7017544 ,  0.7017544 ,  1.0526316 ,  0.        ,
   0.35087727,  1.05263141,  0.        ,  0.350877 ,  0.70175453,
   0.35087727,  0.        ,  0.        ,  0.        ,  0.        ,
   0.        ,  0.350877 ,  0.        ,  0.        ,  0.        ,
   0.        ,  0.350877 ,  0.350877 ,  0.        ,  0.        ,
   0.        ,  0.35087648,  0.        ,  0.35087753,  0.        ,
   0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
   0.        ,  0.35087753,  0.        ,  0.        ,  0.        ,
   0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
   0.35087648,  0.35087753,  0.        ,  0.35087648,  0.        ,
   0.        ,  0.        ,  0.70175506,  0.        ,  0.        ,
   0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
   0.70175506,  0.        ,  0.35087753,  0.        ,  0.        ,
   0.35087544,  0.        ,  0.        ,  0.        ,  0.70175506,
   0.        ,  0.35087544,  0.        ,  0.        ,  0.70175506,
   0.35087753,  0.        ,  0.        ,  0.35087544,  0.        ,
   0.70175506,  0.35087753,  0.        ,  0.        ,  0.        ,
   0.70175506,  0.35087753,  0.35087753,  0.        ,  0.70175506,
   0.        ,  0.70175506,  0.        ,  1.05263258,  0.70175506,
   0.35087753,  1.75437718,  2.45614269,  4.91228539,  43.50881342]),
 array([0.        ,  0.01      ,  0.02      ,  0.03      ,  0.04      ,
   0.05      ,  0.06      ,  0.07      ,  0.08      ,  0.09      ,
   0.1        ,  0.11      ,  0.12      ,  0.13      ,  0.14      ,
   0.15000001,  0.16      ,  0.17      ,  0.18000001,  0.19      ,
   0.2        ,  0.20999999,  0.22      ,  0.23      ,  0.23999999,
   0.25      ,  0.25999999,  0.27000001,  0.28      ,  0.28999999,
   0.30000001,  0.31      ,  0.31999999,  0.33000001,  0.34      ,
   0.34999999,  0.36000001,  0.37      ,  0.38      ,  0.38999999,
   0.40000001,  0.41      ,  0.41999999,  0.43000001,  0.44      ,
   0.44999999,  0.46000001,  0.47      ,  0.47999999,  0.49000001,
   0.5        ,  0.50999999,  0.51999998,  0.52999997,  0.54000002,
   0.55000001,  0.56      ,  0.56999999,  0.57999998,  0.58999997,
   0.60000002,  0.61000001,  0.62      ,  0.63      ,  0.63999999,
   0.64999998,  0.66000003,  0.67000002,  0.68000001,  0.69      ,
   0.69999999,  0.70999998,  0.72000003,  0.73000002,  0.74000001,
   0.75      ,  0.75999999,  0.76999998,  0.77999997,  0.79000002,
   0.80000001,  0.81      ,  0.81999999,  0.82999998,  0.83999997,
   0.85000002,  0.86000001,  0.87      ,  0.88      ,  0.88999999,
   0.89999998,  0.91000003,  0.92000002,  0.93000001,  0.94      ,
   0.94999999,  0.95999998,  0.97000003,  0.98000002,  0.99000001,
   1.        ]),
 <BarContainer object of 100 artists>)
```



```
In [ ]: e_samples = pg.sample(e)
```

```
In [19]: print(e_samples)
e_samples = [np.median(sample) for sample in e_samples]
print(np.mean(e_samples))
e_samples
plt.hist(e_samples, bins=100)
```


0.6350877192982456

