1

```python
import matplotlib.pyplot as plt
import numpy as np

def probability(eps, N, mu, M):
  num_trial = 1000
  k = np.random.binomial(N, mu, (num_trial, M))
  p = np.abs(k / N - mu).max(axis = 1) > eps
  count = 0
  for res in p:
    if res:
      count += 1
  return count / num_trial

def hoeffding(N, eps):
  return 2 * np.exp(-2 * N * eps * eps)

P_6 = []
P_60 = []
hoeffding_list_6 = []
hoeffding_list_60 = []
mu = 0.5
M = 2
eps_range = []
for i in range(1, 101, 1):
  eps = i / 100
  eps_range.append(eps)
  P_6.append(probability(eps, 6, mu, M))
  hoeffding_list_6.append(hoeffding(6, eps))
  P_60.append(probability(eps, 60, mu, M))
  hoeffding_list_60.append(hoeffding(60, eps))

plt.figure()
plt.xlabel("epsilon")
plt.plot(eps_range, P_6, "r")
plt.plot(eps_range, hoeffding_list_6, "c")

plt.plot(eps_range, P_60, "g")
plt.plot(eps_range, hoeffding_list_60, "b")
plt.legend(["N = 6", "hoeffiding with N = 6", "N = 60", "hoeffiding with N = 60"])
```

2

```python
import numpy as np
```

```
data_points = []

mu = 0.5

for _ in range(int(mu * 1000)):
  data_points.append(0)

for _ in range(int(1000 - mu * 1000)):
  data_points.append(1)

np.random.shuffle(data_points)

N = 10


def random_draw(data_points, N):
  res = []
  error_count = 0
  for _ in range(N):
    cur_draw = np.random.choice(data_points, 1)[0]
    if cur_draw == 0:
      error_count += 1
    res.append(cur_draw)
  error_rate = error_count / N
  return error_rate

# random_draw(data_points, N)

# b

num_choose = 100

error_rate_list_100 = []

for _ in range(num_choose):
  error_rate_list_100.append(random_draw(data_points, N))

print(np.max(error_rate_list_100))
print(np.min(error_rate_list_100))
print(np.mean(error_rate_list_100))
print(np.std(error_rate_list_100))

diff_count = 0
in_range_count = 0
learned_count = 0

for n in error_rate_list_100:
  if n != mu:
    diff_count += 1
  if np.abs(n - mu) < 0.05:
    in_range_count += 1
  if n <= 0.45 or n >= 0.55:
    learned_count += 1
```

```
print(diff_count)
print(in_range_count)
print(error_rate_list_100)
print(learned_count)
```

5

```
import math
import numpy as np
import matplotlib.pyplot as plt

def f(x):
  # return math.exp(3 * x) / (1 + math.exp(3 * x))
  return math.exp(3 * x) / (1 + math.exp(3 * x)) + np.random.normal(0.1, 0.004, 1)

def axb(a, x, b):
  return np.multiply(a, x) + b

plt.figure()

x = [i * 0.01 for i in range(-100, 101)]
y = [f(i) for i in x]

a_list = []
b_list = []

x1_list = []
x2_list = []

a_best = 0
b_best = 0
min_mse = 9999999

for _ in range(100):
  x1 = np.random.uniform(0, 1.0000001)
  y1 = f(x1)
  x2 = np.random.uniform(0, 1.0000001)
  y2 = f(x2)
  x1_list.append(x1)
  x2_list.append(x2)

  a = (y1 - y2) / (x1 - x2)
  b = (x1 * y2 - x2 * y1) / (x1 - x2)
  a_list.append(a)
  b_list.append(b)

  plt.plot(x, axb(a, x, b), 'lightgrey')

  mse_list = []
```

```
  for _ in range(20):
    x_test = np.random.uniform(0, 1.0000001)
    mse = (a * x_test + b - f(x_test)) ** 2
    mse_list.append(mse)
  if np.mean(mse_list) < min_mse:
    a_best = a
    b_best = b
    min_mse = np.mean(mse_list)

plt.plot(x, y, 'r')
hg_mean = [np.mean(a_list) * i + np.mean(b_list) for i in x]
plt.plot(x, hg_mean, 'b')
plt.show()

a_mean = np.mean(a_list)
b_mean = np.mean(b_list)
print("a: ", a_mean)
print("b: ", b_mean)


bias_list = []
var_list = []
EdEout_list = []

for i in range(len(a_list)):
  x1 = np.random.uniform(0, 1.0000001)
  x2 = np.random.uniform(0, 1.0000001)

  hg_mean_x_fx = (a_mean * x1 + b_mean - f(x1)) ** 2
  bias_list.append(hg_mean_x_fx)
  hg_mean_x_fx = (a_mean * x2 + b_mean - f(x2)) ** 2
  bias_list.append(hg_mean_x_fx)

  a = a_list[i]
  b = b_list[i]

  hg_d_hg_mean = ((a * x1 + b - (a_mean * x1 + b_mean)) ** 2 + (a * x2 + b - (a_mean * x2 + b_mean)) ** 2) / 2
  var_list.append(hg_d_hg_mean)

  hg_d_fx = ((a * x1 + b - f(x1)) ** 2 + (a * x2 + b - f(x2)) ** 2) / 2
  EdEout_list.append(hg_d_fx)

print("bias:", np.mean(bias_list))
print("var:", np.mean(var_list))
print("EdEout(hg):", np.mean(EdEout_list))
```