

CSL7670 : Fundamentals of Machine Learning

Lab Report



॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

Name:	SOUHITYA KUNDU
Roll Number:	M20PH209
Program:	MSc-MTech(Physics & Materials Sc.)

Chapter 1

Lab-7

1.1 Objective

The objective is to learn about probability and implement it .

1.2 Problem-1

The main objective of the first problem is to implement the theoretical aspects of probability and apply it on a dataset containing marks:

- First task is to understand and compute the histogram based on the given dataset
- Second we obtained the probability distribution by normalizing the histogram.
- Thirdly, we generate the probability of the number of students who have got marks more than 75/100.
- Fourth, we tried for a Gaussian fit and observed the fit for any anomaly

Solution 1(a):

```
1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[1]:
5
6
7  import matplotlib.pyplot as plt
8  import numpy as np
9  import pandas as pd
10 from scipy.stats import norm
11 from scipy.optimize import curve_fit
12
13 #dataset import
14 df = pd.read_excel("marks.xlsx")
15
16 #1a Compute a histogram
17 hist, bins, _ = plt.hist(df, bins=5, density=True, alpha=0.6, color='b',
18     ↪ label='Histogram')
19
20 plt.legend()
21 plt.show()
```

```

21 print(df.shape)
22
23
24 # In[ ]:

```

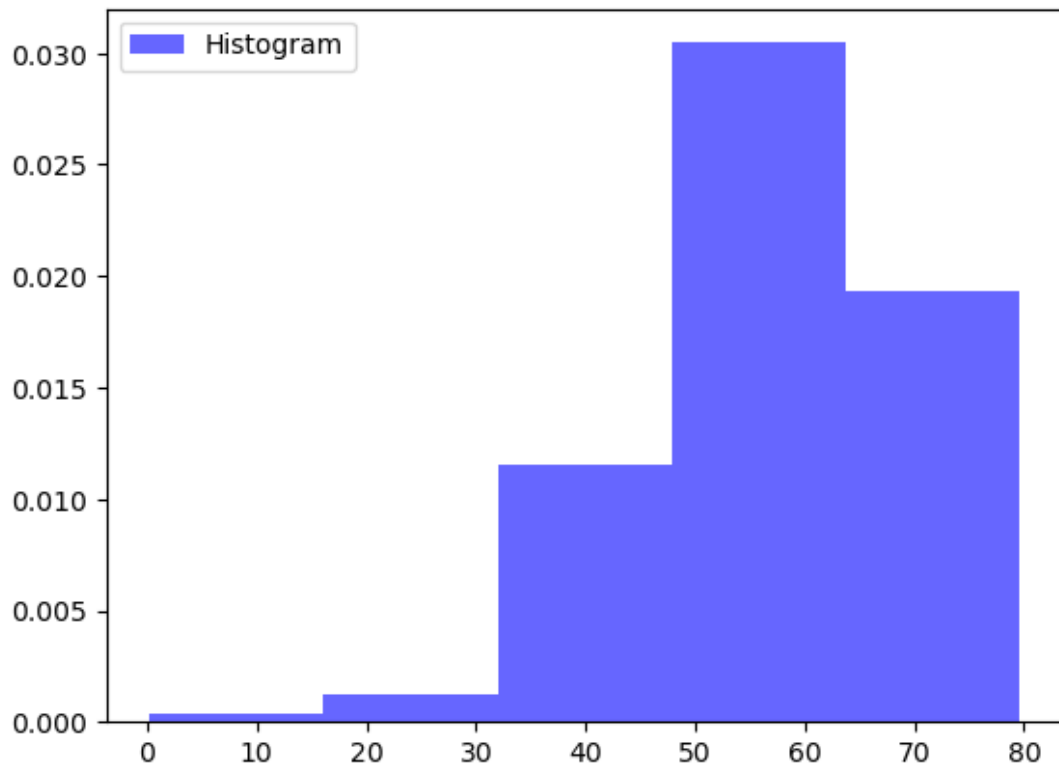


Figure 1.1: The histogram.

Solution 1(b):

```

1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[6]:
5
6
7  import matplotlib.pyplot as plt
8  import numpy as np
9  import pandas as pd
10 from scipy.stats import norm
11 from scipy.optimize import curve_fit
12
13 #dataset import
14 df = pd.read_excel("marks.xlsx")
15
16
17 #1a Compute a histogram
18 # hist, bins, _ = plt.hist(df, bins=5, density=True, alpha=0.6, color='b',
19     ↪ label='Histogram')

```

```

20
21
22 # 1b
23 # I took the row numbers from teh dataset for total length of teh dataset
24 total_count = df.shape[0]
25
26 # Bin width calculation
27 bin_widths = np.diff(bins)
28
29 # Normalizing the histogram .
30 normalized_hist = hist / (total_count * bin_widths)
31
32 # Plot the normalized histogram
33 plt.bar(bins[:-1], normalized_hist, width=bin_widths, alpha=0.8, color='r'
34         ↪ , label='Normalized_Histogram')
35
36 # Add labels and legend
37 plt.xlabel('The_Values')
38 plt.ylabel('Probability_Density')
39 plt.legend()
40 plt.show()
41
42 # In[ ]:

```

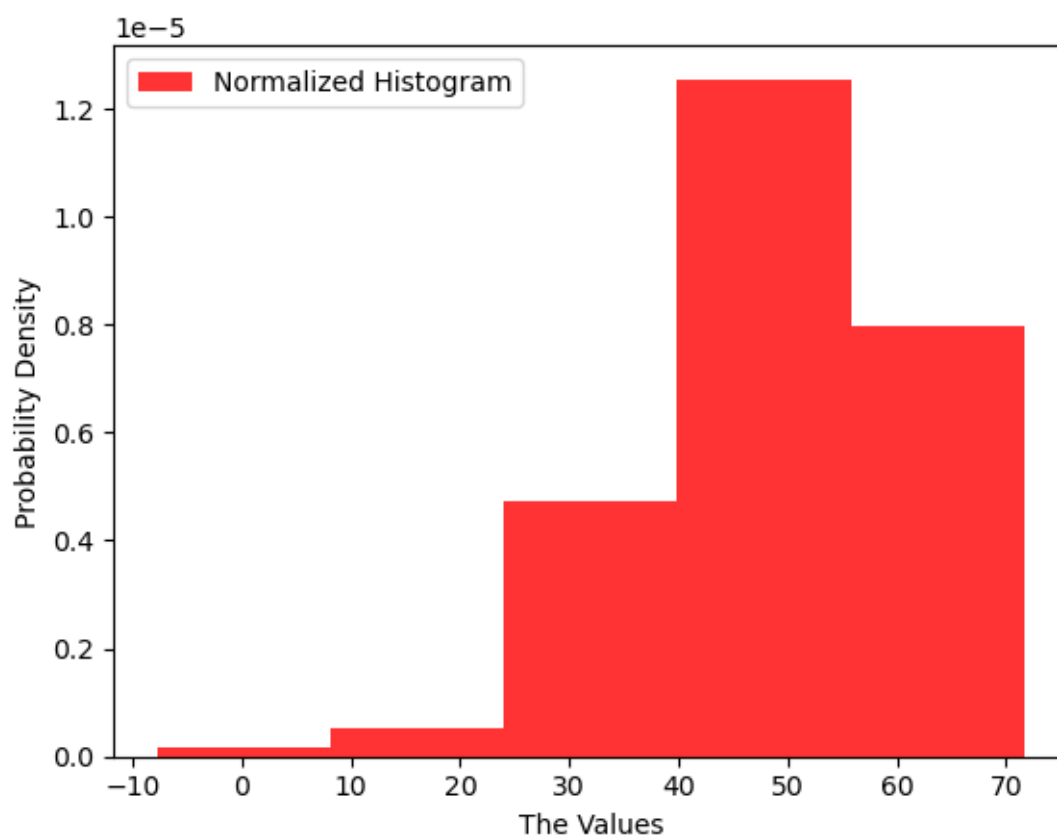


Figure 1.2: The probability distribution after normalizing histogram.

Solution 1(c):

```

1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[5]:
5
6
7  import matplotlib.pyplot as plt
8  import numpy as np
9  import pandas as pd
10 from scipy.stats import norm
11 from scipy.optimize import curve_fit
12
13 #dataset import
14 df = pd.read_excel("marks.xlsx")
15
16
17 #1a Compute a histogram
18 # hist, bins, _ = plt.hist(df, bins=5, density=True, alpha=0.6, color='b',
19     ↪ label='Histogram')
20
21 # 1b
22 # I took the row numbers from teh dataset for total length of teh dataset
23 total_count = df.shape[0]
24
25 # Bin width calculation
26 bin_widths = np.diff(bins)
27
28 # Normalizing the histogram .
29 normalized_hist = hist / (total_count * bin_widths)
30
31 # Plot the normalized histogram
32 # plt.bar(bins[:-1], normalized_hist, width=bin_widths, alpha=0.8, color='
33     ↪ r', label='Normalized Histogram')
34
35 # Add labels and legend
36 plt.xlabel('The Values')
37 plt.ylabel('Probability Density')
38 plt.legend()
39
40 # 1c) probability of people with 75+ marks
41 plt.hist(df, bins=20, density=False, alpha=0.6, color='b', label='
42     ↪ Histogram')
43
44 # In[6]:
45
46
47 count_num = int(input("Enter counts of marks with 75+: \t"))
48 probabiltiy_75 = count_num/total_count
49 print(f"Probability of marks greater than 75+: {probabiltiy_75}")
50

```

```
51  
52 # In[ ]:
```

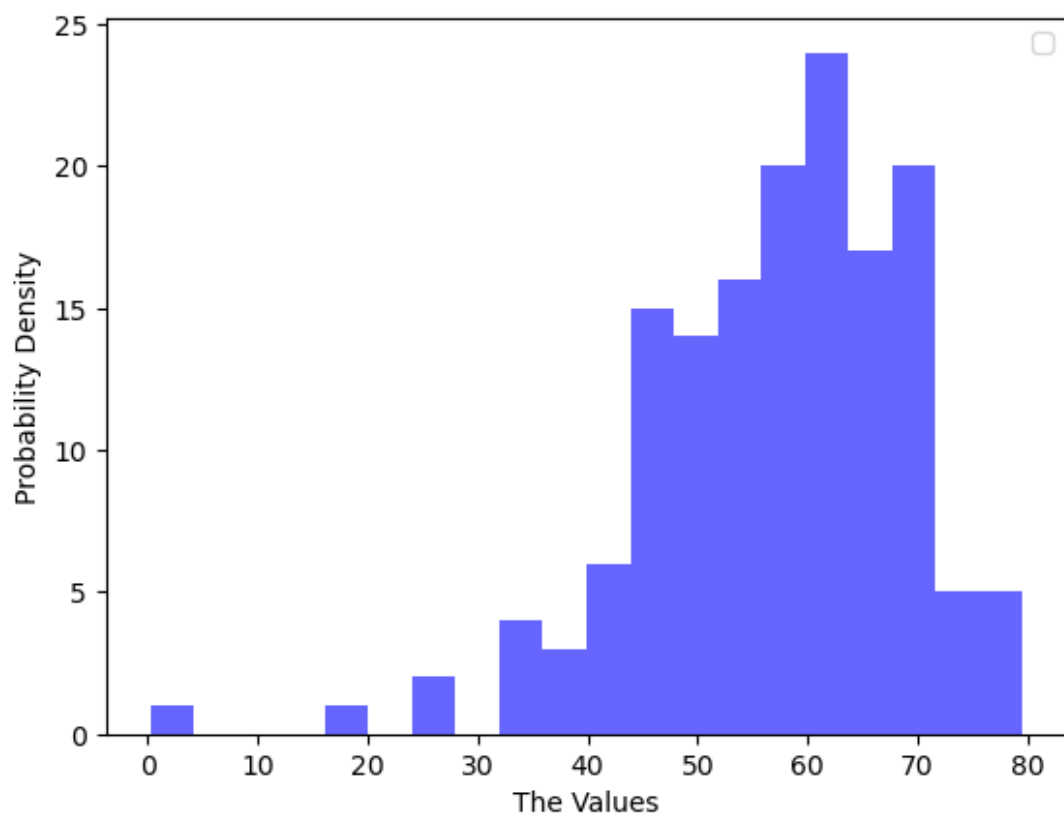


Figure 1.3: The probability of students getting more than 75 marks can be calculated from here onwards.

Enter	counts	of	marks	with	75+: 5	
Probability	of	marks	greater	than	75+:0.032679738562091505	

Solution 1(d):

```

1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[22]:
5
6
7  import matplotlib.pyplot as plt
8  import numpy as np
9  import pandas as pd
10 from scipy.stats import norm
11 from scipy.optimize import curve_fit
12
13 #dataset import
14 df = pd.read_excel("marks.xlsx")
15
16
17 # In[63]:
18
19
20 # 1d)
21 #
22     ↪ -----
23     ↪
24 # NameError                                Traceback (most recent call
25     ↪ last)
26 # Cell In[29], line 12
27 #           8 #Normalizing the marks
28 #           10 amplitude_estimate = 1 # Initial estimate for the amplitude
29 # ---> 12 params, covariance = curve_fit(gaussian, np.arange(len(df)),
30     ↪ data, p0=[mean_estimate, stddev_estimate, amplitude_estimate], maxfev
31     ↪ =100000)
32 #           14 # Extract the parameters of the fitted Gaussian
33 #           15 mu, sigma, A = params
34
35 # NameError: name 'gaussian' is not defined
36
37 # Hence we will create a function by the name of gaussian first and pass
38     ↪ it in the curve_fit.
39 # *****
40 # Create a histogram
41 hist, bins, _ = plt.hist(df, bins=20, density=True, alpha=0.6, color='g',
42     ↪ label='Histogram')
43
44 # Calculate the total count of data points
45 total_count = len(df)
46
47 # Calculate the bin widths
48 bin_widths = np.diff(bins)
49

```



```

43 # Normalize the histogram by dividing bin counts by total count and bin
    ↪ width
44 normalized_hist = hist / (total_count * bin_widths)
45
46 # Plot the normalized histogram
47 plt.bar(bins[:-1], normalized_hist, width=bin_widths, alpha=0.6, color='b'
    ↪ , label='Normalized_Histogram')
48
49 # Add labels and legend
50 plt.xlabel('Value')
51 plt.ylabel('Probability_Density')
52 plt.legend()
53 # *****
54
55 df1 = df.to_numpy()
56 df1 = df1.flatten()
57 def gaussian(x, mu , sigma , A):
58     return A*np.exp(-(x-mu)**2/(2*sigma**2))
59
60 # Fit the data to the Gaussian distribution
61 mean_estimate = np.mean(df1)
62 stddev_estimate = np.std(df1)
63 # making the data zero mean and std=1
64 df1 = (df1 - mean_estimate)/stddev_estimate
65
66
67 #Normalizing the marks
68
69 amplitude_estimate = 1 # Initial estimate for the amplitude
70
71 params, covariance = curve_fit(gaussian, np.arange(len(df1)), df1, p0=[
    ↪ mean_estimate, stddev_estimate, amplitude_estimate], maxfev=100000)
72
73 # Extract the parameters of the fitted Gaussian
74 mu, sigma, A = params
75
76 # Generate the x values for the plot
77 x = np.linspace(-20, 20, 1000)
78
79 # Calculate the fitted Gaussian curve
80 fit = gaussian(x, mu, sigma, A)
81
82 # plt.hist(df, bins=5, density=True, alpha=0.6, color='b', label='
    ↪ Histogram')
83
84
85 # hist, bins, _ = plt.hist(df, bins=10, density=True, alpha=0.6, color='g
    ↪ ', label='Histogram')
86 plt.plot(x, fit, 'r-', label='Fitted_Gaussian')
87
88 # Add labels and legend
89 plt.xlabel('Marks')
90 plt.ylabel('Probability_Density')
91 plt.legend()

```

```
92 plt.grid()
93
94 # Show the plot
95 plt.show()
96
97 # Print the estimated parameters of the Gaussian distribution
98 print("Estimated Mean:", mu)
99 print("Estimated Standard Deviation:", sigma)
100
101
102
103 # In[62]:
104
105
106 import seaborn as sns
107 sns.distplot(df, bins = 20, kde=True)
108
109
110 # In[ ]:
```

ANSWER 1d: The dataset seems to have a left skewness which is not normally distributed by nature. But while fitting the curve we have fitted it in the form of Gaussian nature which can't be true. Gaussian means normally or symmetric fit. But if the data itself has left skewness then the gaussian fit is wrong in this case.

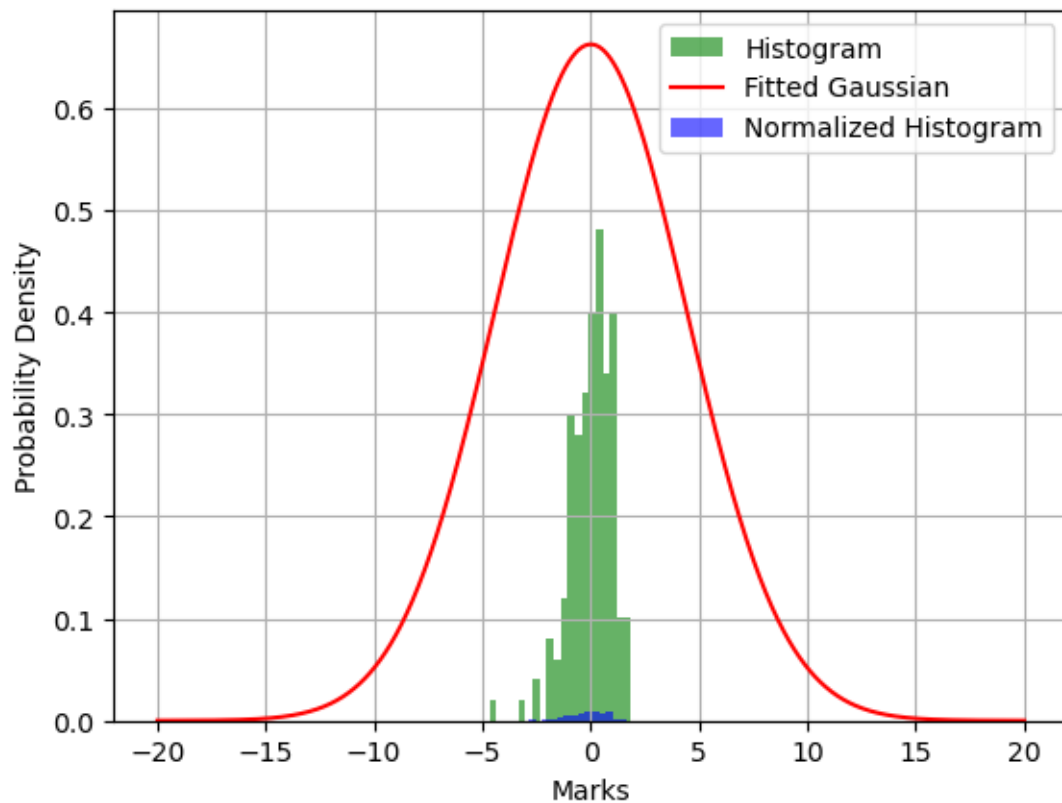


Figure 1.4: The Gaussian fit based on code

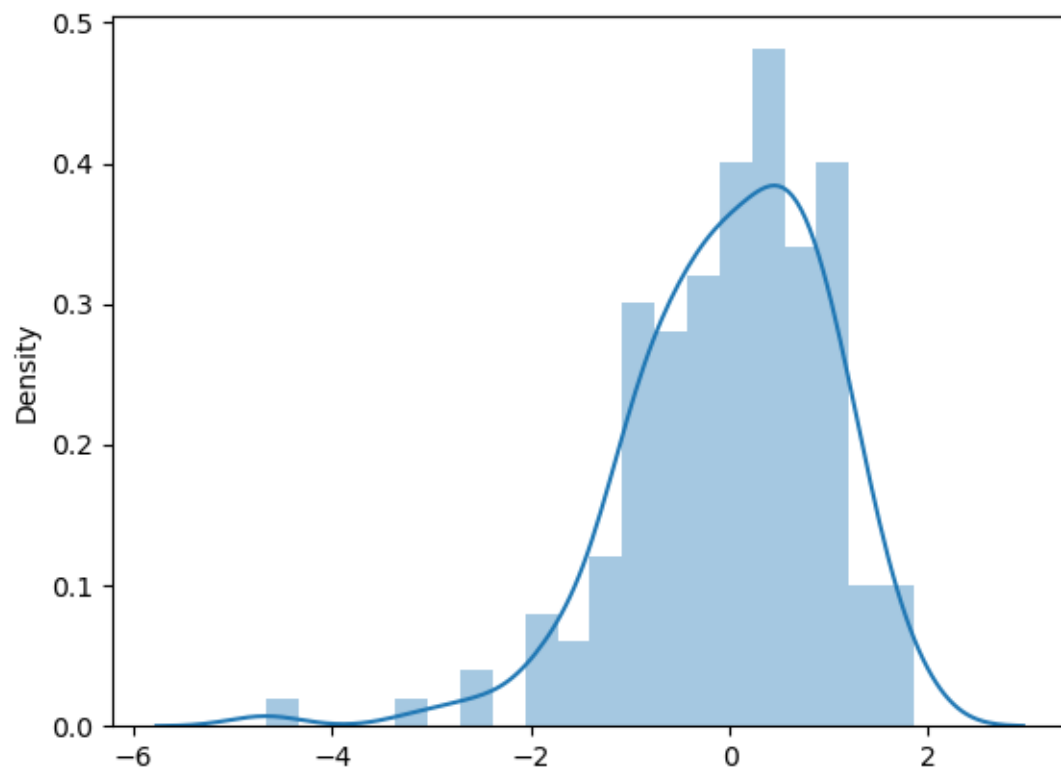


Figure 1.5: The Original fit is a bit left skewed type, not normalized plot, which is done by seaborn plot.

1.3 Problem-2

- First plot $N(0, 1)$
- Second Plot $U[0,5]$
- Thirdly, Plot $N(0, 0.8)$
- Fourth, I have plotted $N(0, 0.6)$

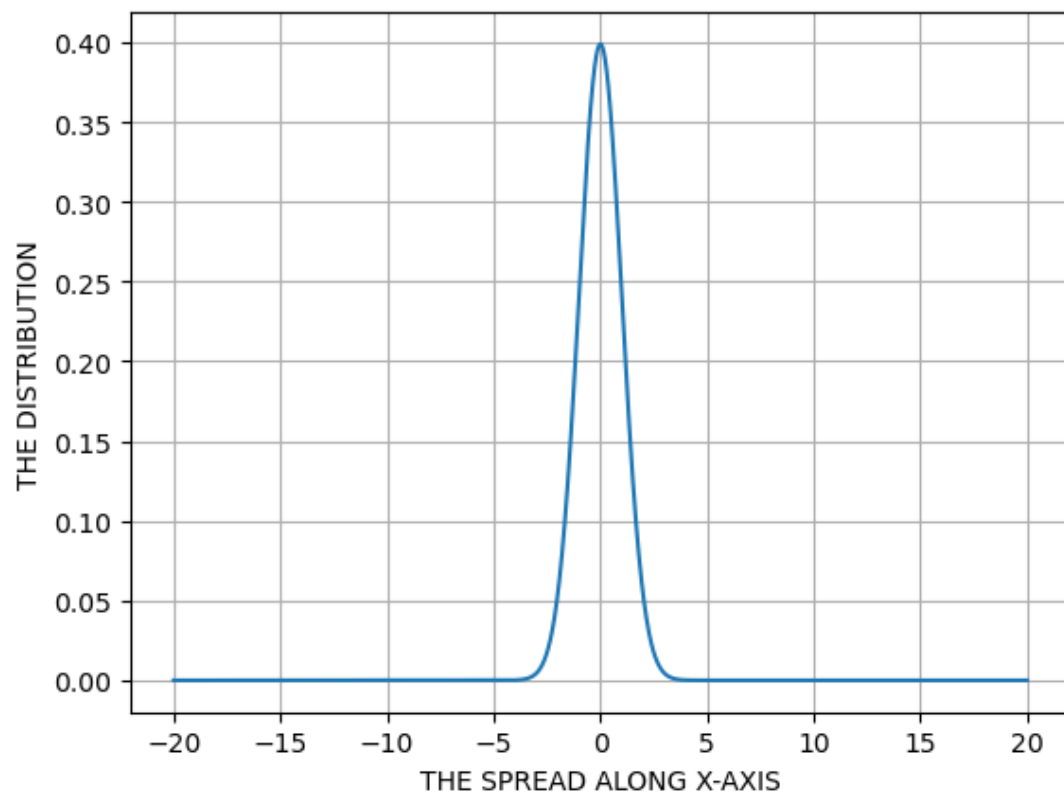
Solution 2(a,b,c,d):

```

1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[17]:
5
6
7  # 1(A) N(0, 1)
8  import numpy as np
9  import matplotlib.pyplot as plt
10 from scipy.stats import norm
11 import statistics
12
13
14 x_axis = np.arange(-20, 20, 0.01)
15
16 mean = 0 #CALCULATING THE MEAN
17 sd = 1    # CALCULATING THE STANDARD DEVIATION
18 plt.plot(x_axis, norm.pdf(x_axis, mean, sd))
19 plt.xlabel("THE SPREAD ALONG X-AXIS")
20 plt.ylabel("THE DISTRIBUTION")
21 plt.grid()
22 plt.show()
23
24
25
26 # In[51]:
27
28
29 # 1(B) U[0,5]
30
31 import numpy as np
32 import matplotlib.pyplot as plt
33
34 values = np.random.uniform(0,5)
35 plt.hist(values)
36 plt.title('Uniform Distribution')
37 plt.ylabel('The Density Values')
38 plt.xlabel('Spread of the Values')
39 plt.show()
40
41
42 # In[12]:
43
44

```

```
45 # 1(C) N (0, 0.8)
46
47 import numpy as np
48 import matplotlib.pyplot as plt
49 from scipy.stats import norm
50 import statistics
51
52 # Plot between -10 and 10 with .001 steps.
53 x_axis = np.arange(-20, 20, 0.01)
54
55 mean = 0 #CALCULATING THE MEAN
56 sd = 0.8 # CALCULATING THE STANDARD DEVIATION
57 plt.plot(x_axis, norm.pdf(x_axis, mean, sd))
58 plt.xlabel("THE SPREAD ALONG X-AXIS")
59 plt.ylabel("THE DISTRIBUTION")
60 plt.grid()
61 plt.show()
62
63
64
65 # In[9]:
66
67
68 # 1(D) N (0, 0.6)
69 import numpy as np
70 import matplotlib.pyplot as plt
71 from scipy.stats import norm
72 import statistics
73
74 # Plot between -10 and 10 with .001 steps.
75 x_axis = np.arange(-20, 20, 0.01)
76
77 mean = 0 #CALCULATING THE MEAN
78 sd = 0.6 # CALCULATING THE STANDARD DEVIATION
79 plt.plot(x_axis, norm.pdf(x_axis, mean, sd))
80 plt.xlabel("THE SPREAD ALONG X-AXIS")
81 plt.ylabel("THE DISTRIBUTION")
82 plt.grid()
83 plt.show()
84
85
86
87 # In[ ]:
```

Figure 1.6: Plot of $N(0, 1)$.

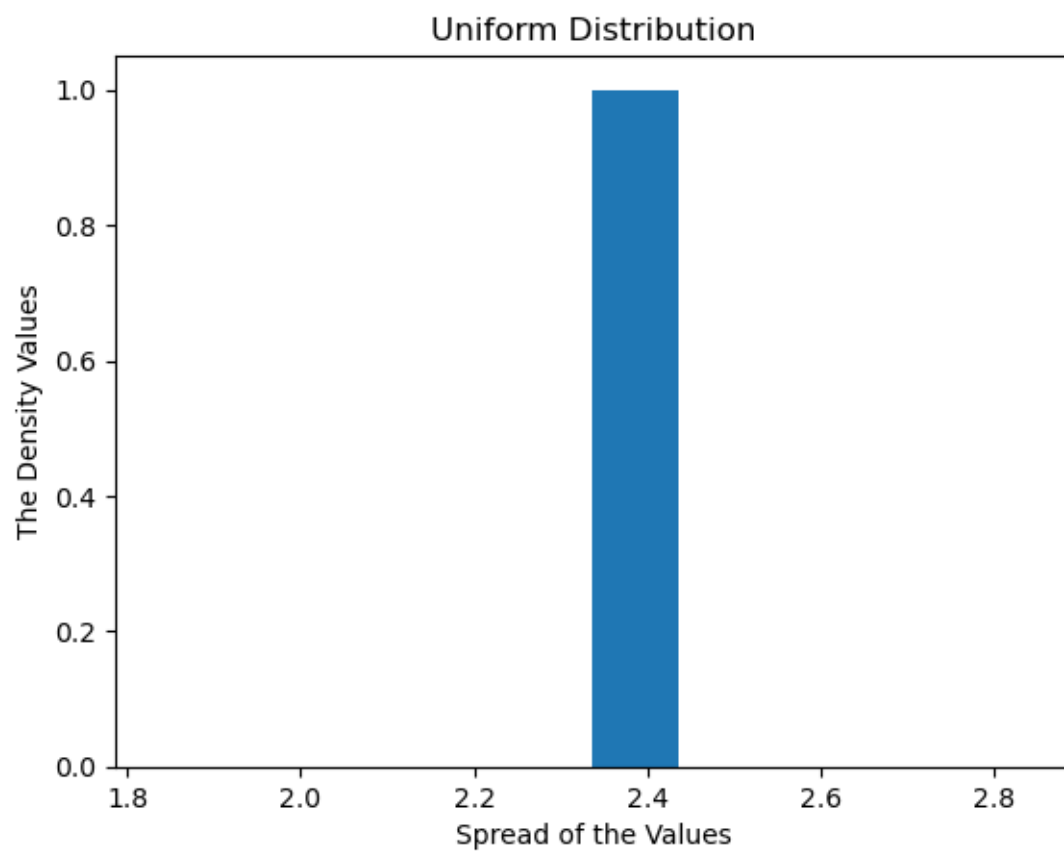
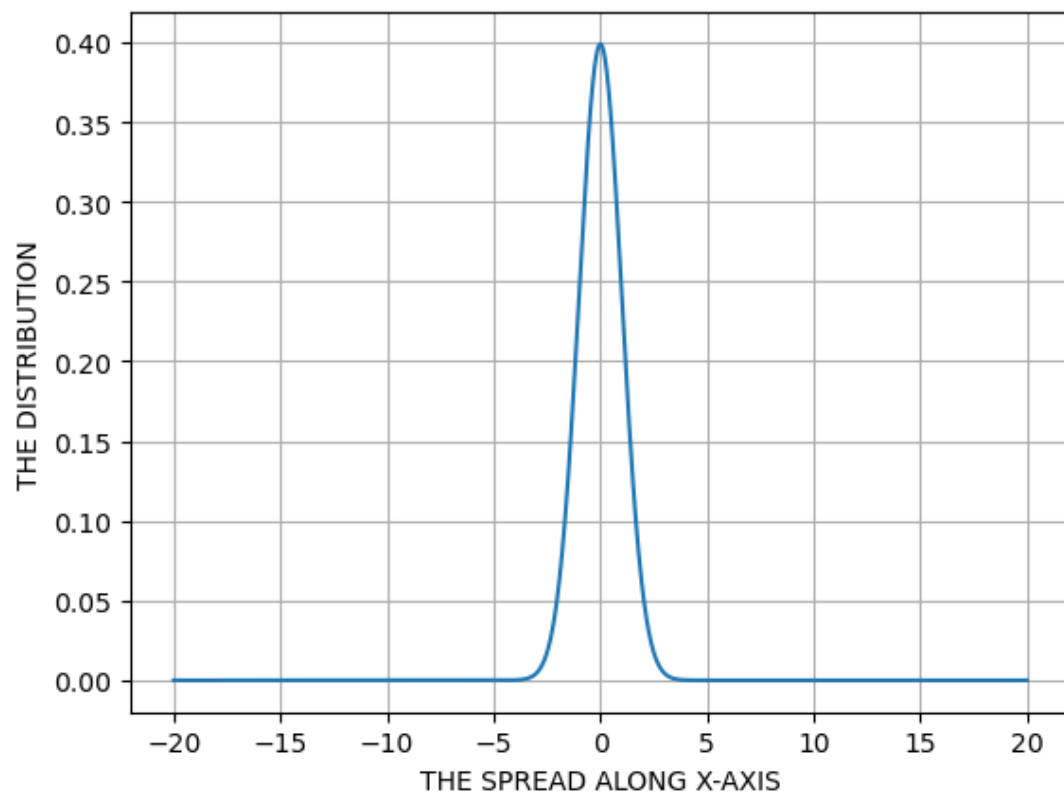


Figure 1.7: Plot of $U[0,5]$.

Figure 1.8: Plot of $N(0, 0.8)$.

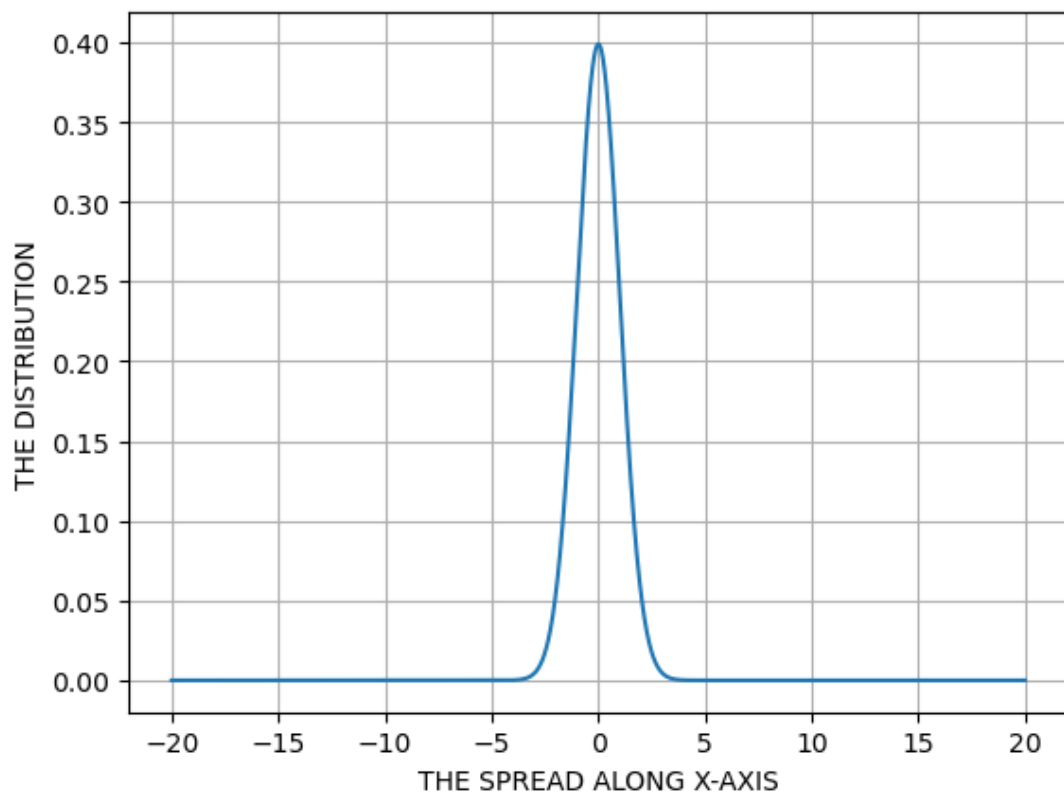


Figure 1.9: Plot of $N(0, 0.6)$.