CSL7670 : Fundamentals of Machine Learning

Lab Report

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):

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.stats import norm
from scipy.optimize import curve_fit

#dataset import
df = pd.read_excel("marks.xlsx")

#1a Compute a histogram
hist, bins, _ = plt.hist(df, bins=5, density=True, alpha=0.6, color='b',
    ↪ label='Histogram')

plt.legend()
plt.show()
```

```
print(df.shape)


# In[ ]:
```

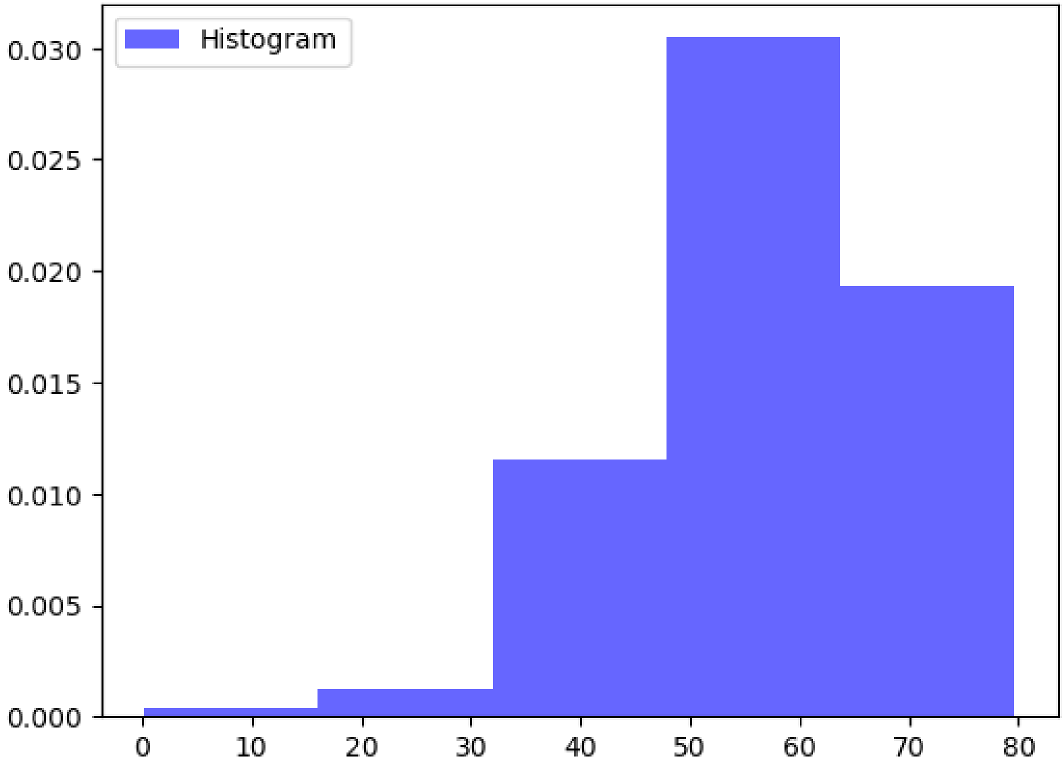

Figure 1.1: The histogram.

Solution 1(b):

```
#!/usr/bin/env python
# coding: utf-8

# In[6]:


import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.stats import norm
from scipy.optimize import curve_fit

#dataset import
df = pd.read_excel("marks.xlsx")


#1a Compute a histogram
# hist, bins, _ = plt.hist(df, bins=5, density=True, alpha=0.6, color='b',
    ↪ label='Histogram')

```

```
# 1b
# I took the row numbers from teh dataset for total length of teh dataset
total_count = df.shape[0]

# Bin width calculation
bin_widths = np.diff(bins)

# Normalizing the histogram .
normalized_hist = hist / (total_count * bin_widths)

# Plot the normalized histogram
plt.bar(bins[:-1], normalized_hist, width=bin_widths, alpha=0.8, color='r'
    , label='Normalized Histogram')

# Add labels and legend
plt.xlabel('The Values')
plt.ylabel('Probability Density')
plt.legend()
plt.show()


# In[ ]:
```

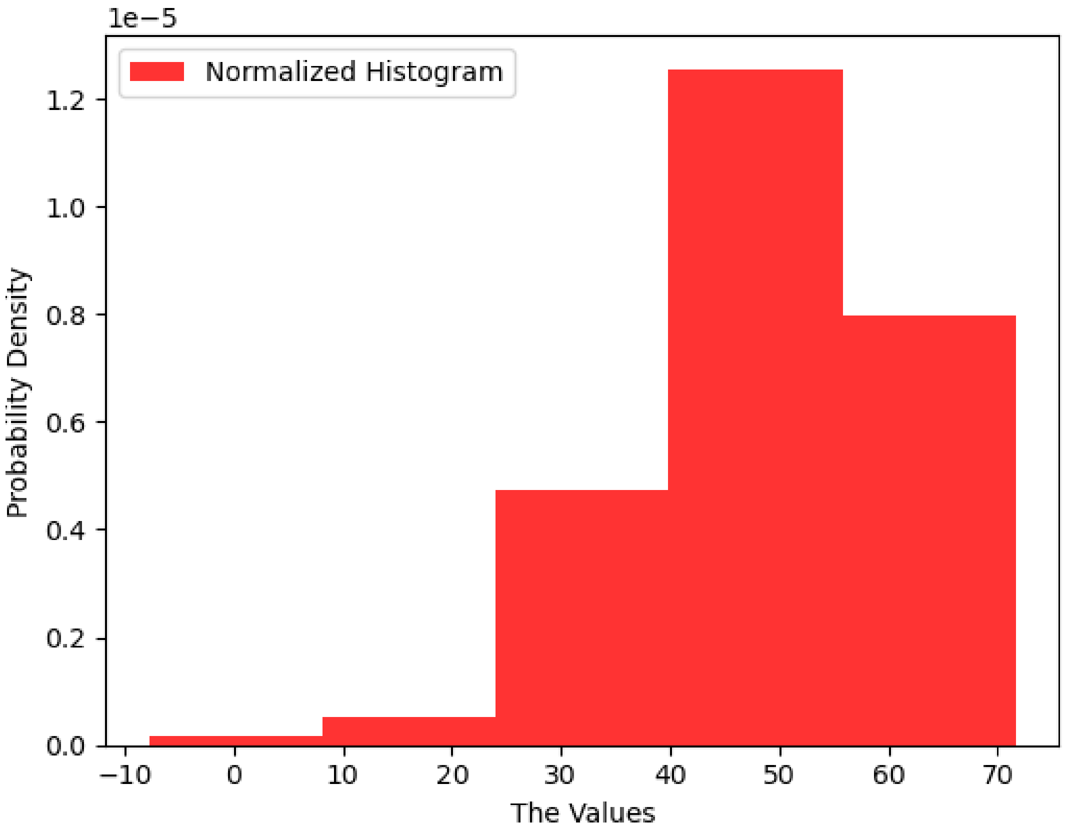

Figure 1.2: The probability distribution after normalizing histogram.

Solution 1(c):

```
#!/usr/bin/env python
# coding: utf-8

# In[5]:


import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.stats import norm
from scipy.optimize import curve_fit

#dataset import
df = pd.read_excel("marks.xlsx")


#1a Compute a histogram
# hist, bins, _ = plt.hist(df, bins=5, density=True, alpha=0.6, color='b', label='Histogram')


# 1b
# I took the row numbers from teh dataset for total length of teh dataset
total_count = df.shape[0]

# Bin width calculation
bin_widths = np.diff(bins)

# Normalizing the histogram .
normalized_hist = hist / (total_count * bin_widths)

# Plot the normalized histogram
# plt.bar(bins[:-1], normalized_hist, width=bin_widths, alpha=0.8, color='r', label='Normalized Histogram')

# Add labels and legend
plt.xlabel('The Values')
plt.ylabel('Probability Density')
plt.legend()


# 1c) probability of people with 75+ marks
plt.hist(df, bins=20, density=False, alpha=0.6, color='b', label='Histogram')


# In[6]:


count_num = int(input("Enter counts of marks with 75+:\t"))
probabilty_75 = count_num/total_count
print(f"Probability of marks greater than 75+:{probabilty_75}")

```

```
# In[ ]:
```

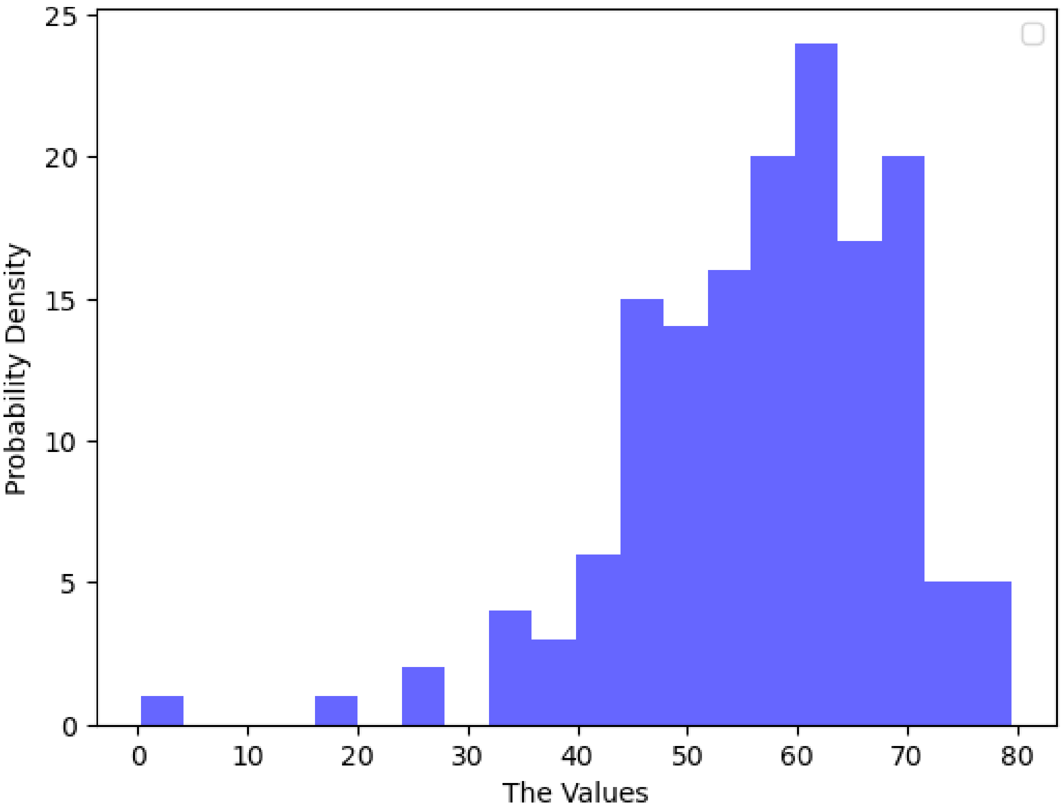

Figure 1.3: The probability of students getting more than 75 marks can be calculated from here onwards.

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

Solution 1(d):

```
#!/usr/bin/env python
# coding: utf-8

# In[22]:


import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.stats import norm
from scipy.optimize import curve_fit

#dataset import
df = pd.read_excel("marks.xlsx")


# In[63]:


# 1d)
# ---------------------------------------------------------------------------
# NameError                                 Traceback (most recent call last)
# Cell In[29], line 12
#        8 #Normalizing the marks
#       10 amplitude_estimate = 1  # Initial estimate for the amplitude
# ---> 12 params, covariance = curve_fit(gaussian, np.arange(len(df)), data, p0=[mean_estimate, stddev_estimate, amplitude_estimate],maxfev=100000)
#       14 # Extract the parameters of the fitted Gaussian
#       15 mu, sigma, A = params

# NameError: name 'gaussian' is not defined

# Hence we will create a function by the name of gaussian first and pass it in the curve_fit.
# ****************************************
# Create a histogram
hist, bins, _ = plt.hist(df, bins=20, density=True, alpha=0.6, color='g', label='Histogram')

# Calculate the total count of data points
total_count = len(df)

# Calculate the bin widths
bin_widths = np.diff(bins)

```

```
# Normalize the histogram by dividing bin counts by total count and bin width
normalized_hist = hist / (total_count * bin_widths)

# Plot the normalized histogram
plt.bar(bins[:-1], normalized_hist, width=bin_widths, alpha=0.6, color='b', label='Normalized Histogram')

# Add labels and legend
plt.xlabel('Value')
plt.ylabel('Probability Density')
plt.legend()
# ******************************************

df1 = df.to_numpy()
df1 = df1.flatten()
def gaussian(x, mu , sigma , A):
    return A*np.exp(-(x-mu)**2/(2*sigma**2))

# Fit the data to the Gaussian distribution
mean_estimate = np.mean(df1)
stddev_estimate = np.std(df1)
# making the data zero mean and std=1
df1 = (df1 - mean_estimate)/stddev_estimate


#Normalizing the marks

amplitude_estimate = 1  # Initial estimate for the amplitude

params, covariance = curve_fit(gaussian, np.arange(len(df1)), df1, p0=[mean_estimate, stddev_estimate, amplitude_estimate],maxfev=100000)

# Extract the parameters of the fitted Gaussian
mu, sigma, A = params

# Generate the x values for the plot
x = np.linspace(-20, 20, 1000)

# Calculate the fitted Gaussian curve
fit = gaussian(x, mu, sigma, A)

# plt.hist(df, bins=5, density=True, alpha=0.6, color='b', label='Histogram')


# hist, bins, _ = plt.hist(df, bins=10, density=True, alpha=0.6, color='g', label='Histogram')
plt.plot(x, fit, 'r-', label='Fitted Gaussian')

# Add labels and legend
plt.xlabel('Marks')
plt.ylabel('Probability Density')
plt.legend()
```

```
plt.grid()

# Show the plot
plt.show()

# Print the estimated parameters of the Gaussian distribution
print("Estimated␣Mean:", mu)
print("Estimated␣Standard␣Deviation:", sigma)



# In[62]:


import seaborn as sns
sns.distplot(df,bins = 20, kde=True)


# 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 guassian fit is wrong in this case.

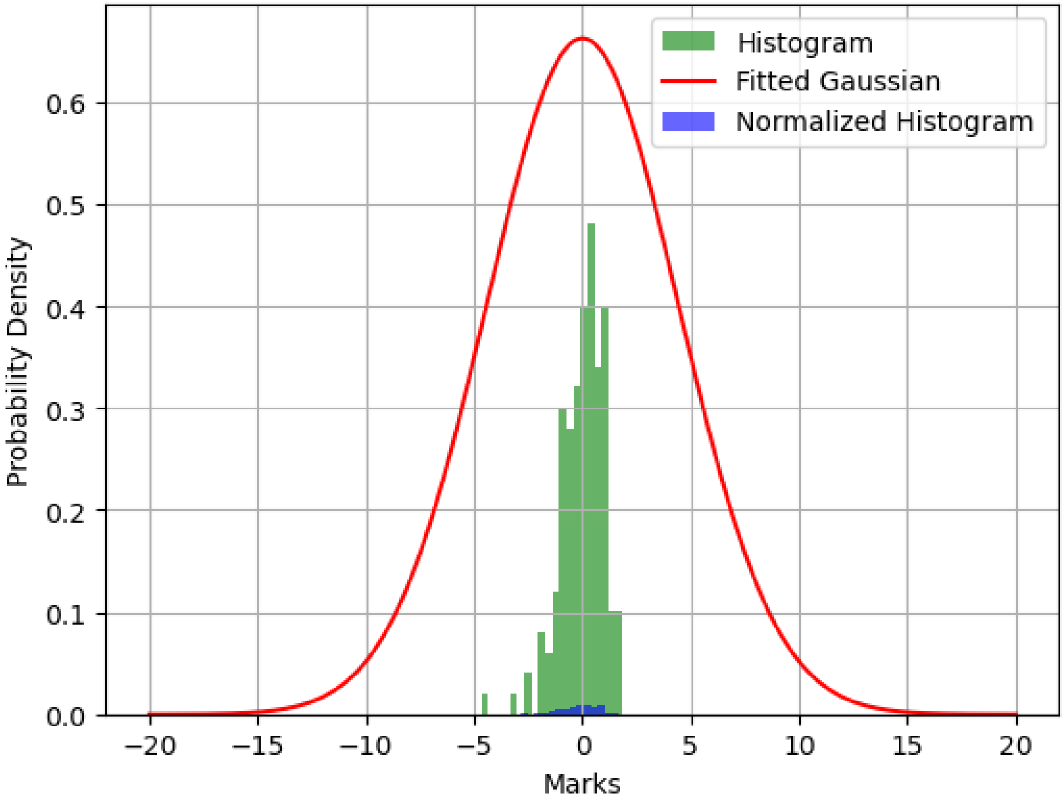

Figure 1.4: The Gaussian fit based on code

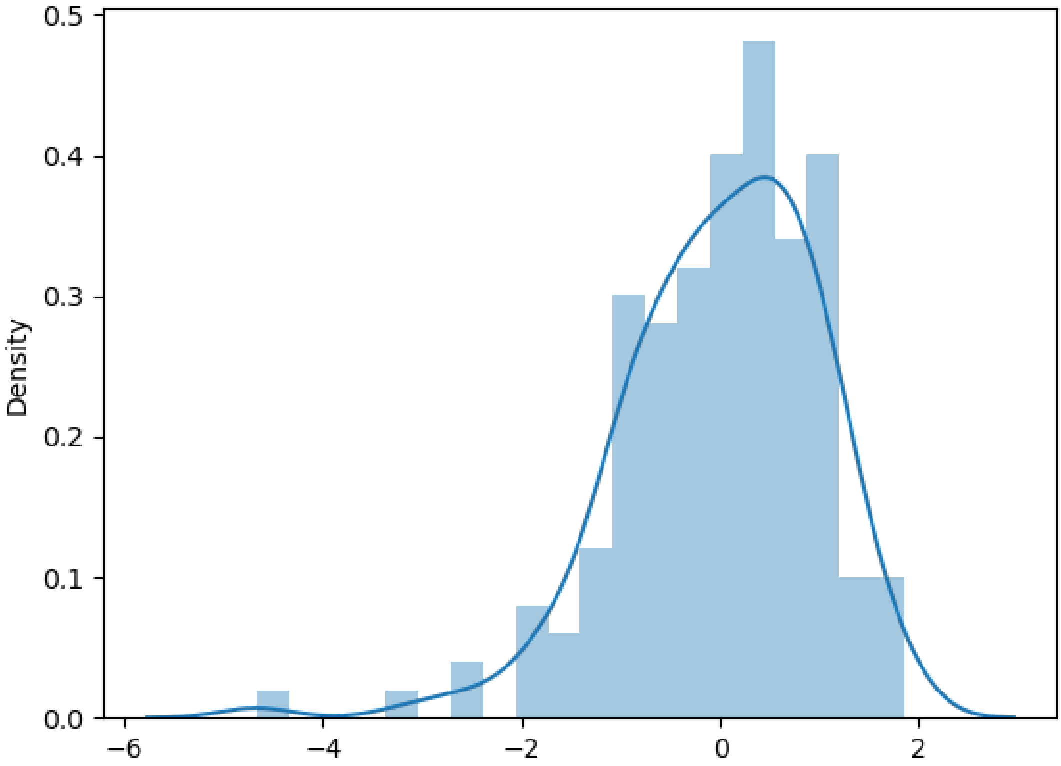

Figure 1.5: The Original fit is a bit left skewed type, not normalized plot, which is done by seaborn plot.

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):

```
#!/usr/bin/env python
# coding: utf-8

# In[17]:


# 1(A) N(0, 1)
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
import statistics


x_axis = np.arange(-20, 20, 0.01)

mean = 0 #CALCULATING THE MEAN
sd = 1   # CALCULATING THE STANDARD DEVIATION
plt.plot(x_axis, norm.pdf(x_axis, mean, sd))
plt.xlabel("THE SPREAD ALONG X-AXIS")
plt.ylabel("THE DISTRIBUTION")
plt.grid()
plt.show()



# In[51]:


# 1(B) U[0,5]

import numpy as np
import matplotlib.pyplot as plt

values = np.random.uniform(0,5)
plt.hist(values)
plt.title('Uniform Distribution')
plt.ylabel('The Density Values')
plt.xlabel('Spread of the Values')
plt.show()


# In[12]:


```

```
# 1(C) N (0, 0.8)

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
import statistics

# Plot between -10 and 10 with .001 steps.
x_axis = np.arange(-20, 20, 0.01)

mean = 0 #CALCULATING THE MEAN
sd = 0.8   # CALCULATING THE STANDARD DEVIATION
plt.plot(x_axis, norm.pdf(x_axis, mean, sd))
plt.xlabel("THE␣SPREAD␣ALONG␣X-AXIS")
plt.ylabel("THE␣DISTRIBUTION")
plt.grid()
plt.show()



# In[9]:


# 1(D) N (0, 0.6)
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
import statistics

# Plot between -10 and 10 with .001 steps.
x_axis = np.arange(-20, 20, 0.01)

mean = 0 #CALCULATING THE MEAN
sd = 0.6   # CALCULATING THE STANDARD DEVIATION
plt.plot(x_axis, norm.pdf(x_axis, mean, sd))
plt.xlabel("THE␣SPREAD␣ALONG␣X-AXIS")
plt.ylabel("THE␣DISTRIBUTION")
plt.grid()
plt.show()



# In[ ]:
```

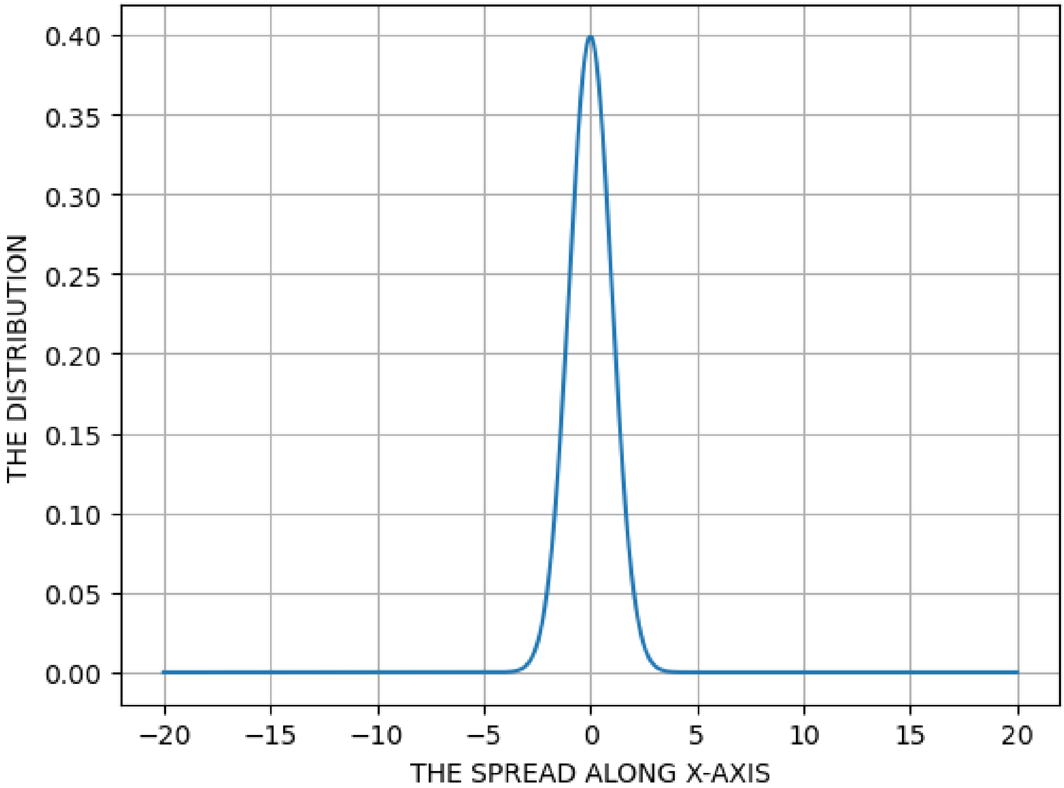

Figure 1.6: Plot of N(0, 1).

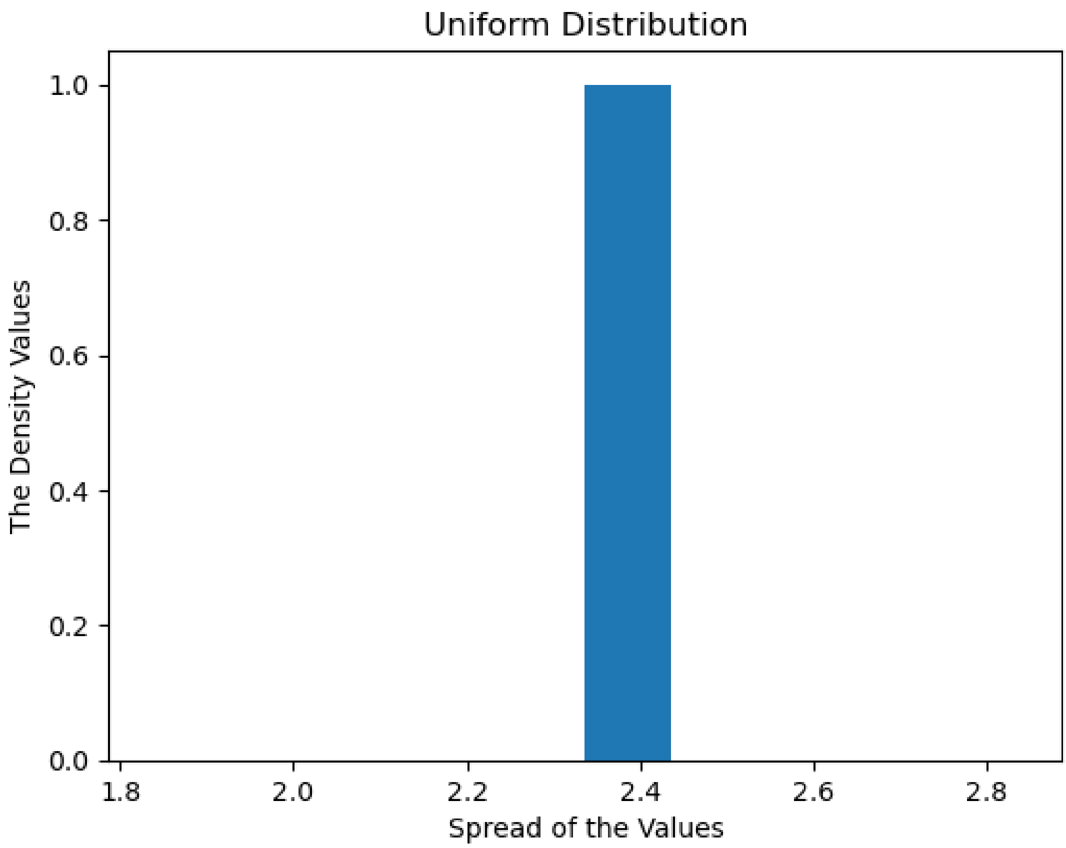

Figure 1.7: Plot of U[0,5].

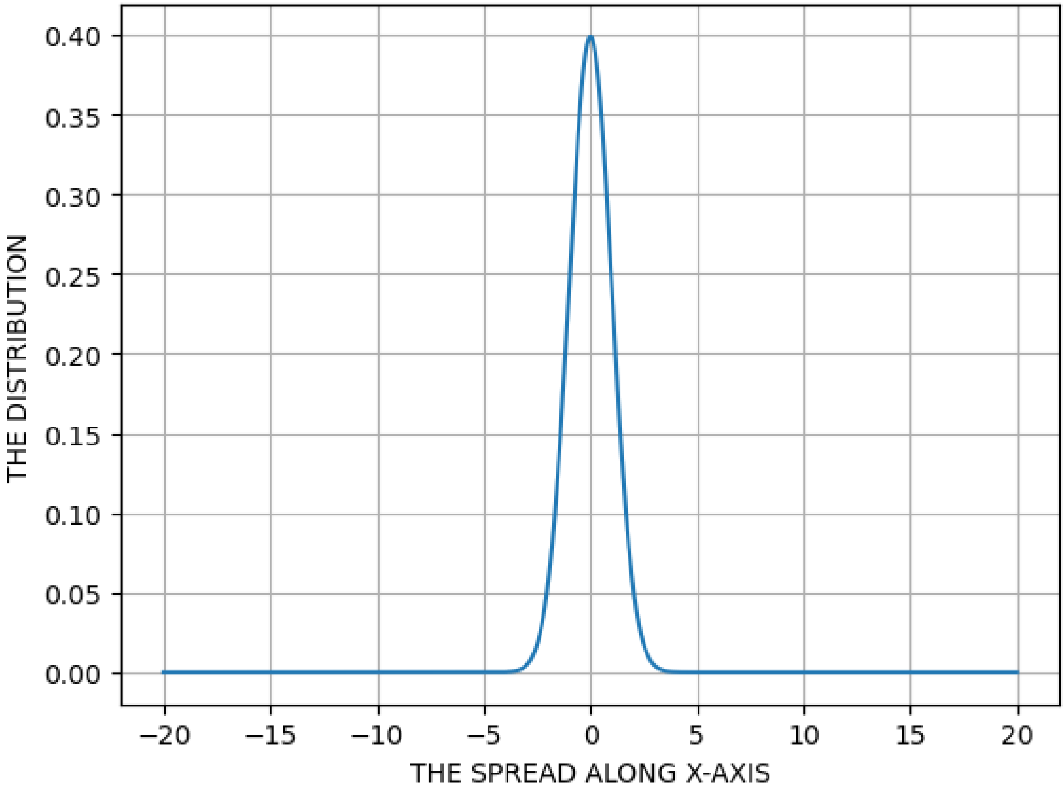

Figure 1.8: Plot of N(0, 0.8).

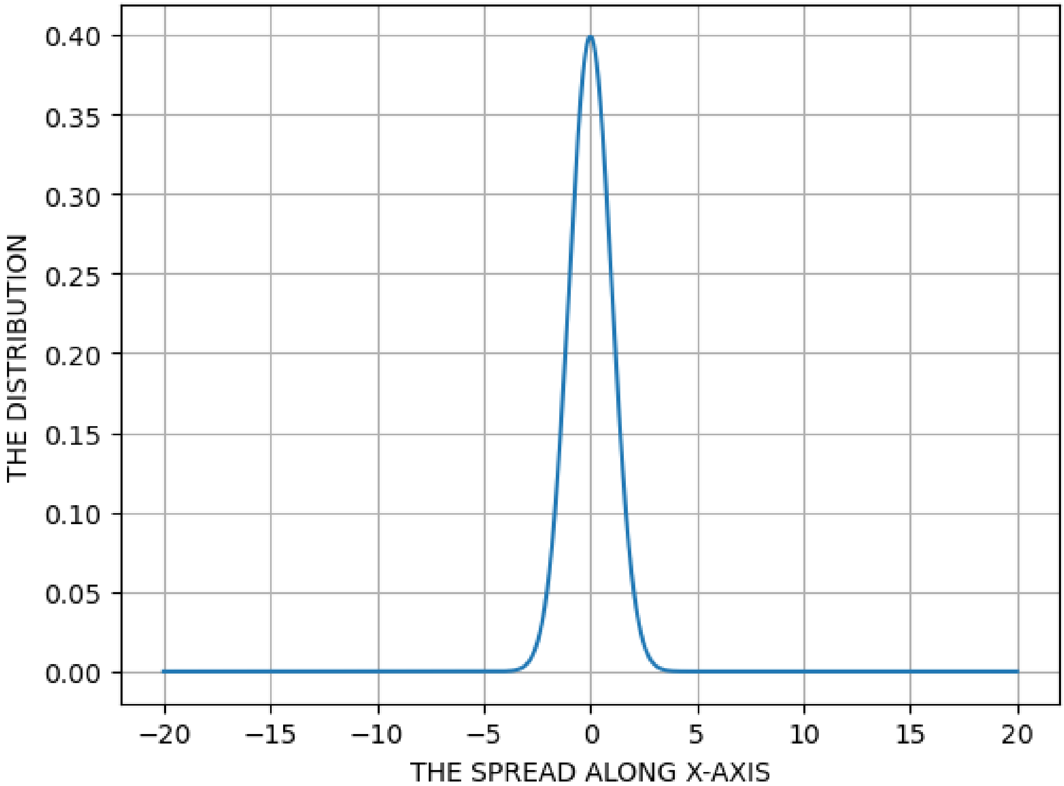

Figure 1.9: Plot of N (0, 0.6).