```
Author   -Shashank Chhoker
Github   -https://github.com/Shashank975
LinkedIn -https://www.linkedin.com/in/shashankchhoker07
Kaggle   -https://www.kaggle.com/shashankchhoker
```

Implementing Statistics Practically

Measure of Central Tendency

1. Mean
2. Median
3. Mode

The terms mean, median, mode, and range describe properties of statistical distributions. In statistics, a distribution is the set of all possible values for terms that represent defined events. The value of a term, when expressed as a variable, is called a random variable.

```
In Machine Learning (and in mathematics) there are often three values that interests us:

Mean - The average value.
Median - The mid point value.
Mode - The most common value.
```

In [1]:
```
#Importing Required Libraries
import numpy as np
import pandas as pd
import statistics
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stat
import pylab
```

In [2]:
```
ages=[23,24,32,45,12,43,67,45,32,56,32,120]
```

In [3]:
```
print(np.mean(ages))
```

```
44.25
```

In [4]:
```
print(np.median(ages))
```

```
37.5
```

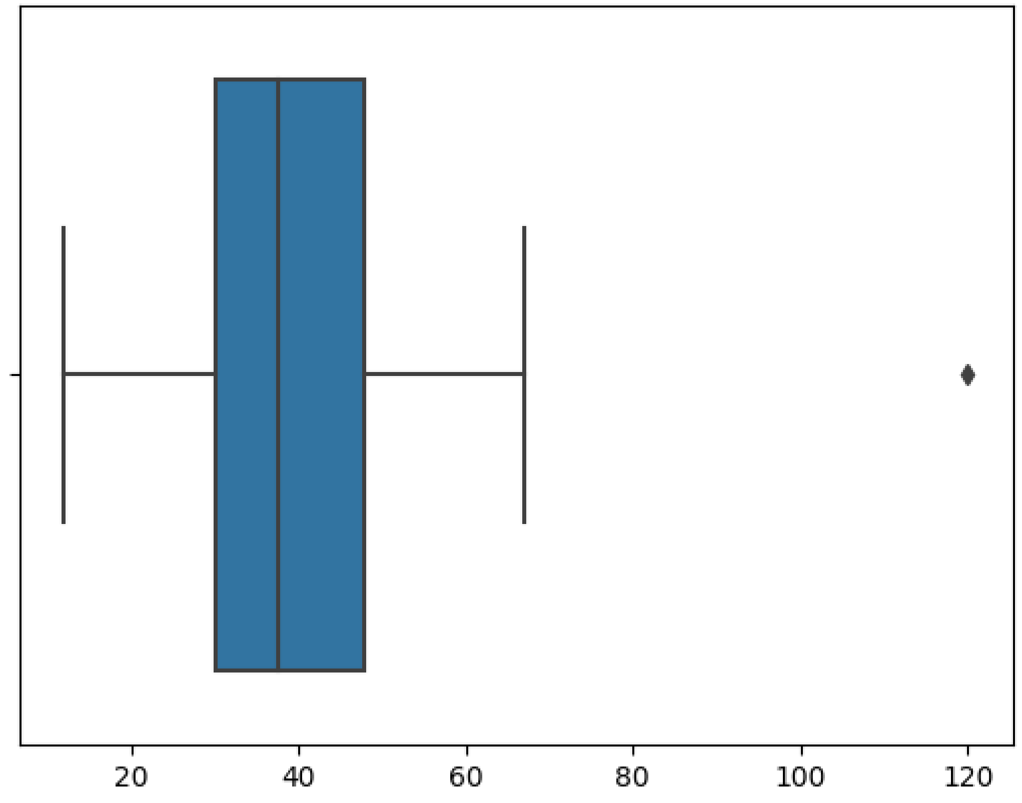
In [5]:
```
print(statistics.mode(ages))
print(statistics.mean(ages))
print(statistics.median(ages))
```

```
32
44.25
37.5
```

```
This code utilizes Seaborn to create a horizontal boxplot showcasing the age distribution and potential outliers
within the sample data.
--> we can easily Detect the outlier here .Which is at the point 120 on the Right side.
```

In [6]:
```
sns.boxplot(x=ages, orient="h")
```

Out[6]:
```
<Axes: >
```



-:5 Number Summary:-

```
-->Now By Using the 5 Number Summary we are finding the Outliers.
```

In [7]:
```python
q1,q3=np.percentile(ages,[25,75])
```

In [8]:
```python
print(q1)
```

```
30.0
```

In [9]:
```python
print(q3)
```

```
47.75
```

-->This code calculates the 25th and 75th percentiles, then computes the interquartile range (IQR). Using the IQR, it defines lower and upper bounds to identify outliers that fall outside these bounds. The identified outliers are printed. Replace the data list with your dataset.

In [10]:
```python
## To Check Outliers [Lower Fence - Higher Fence ]
```

In [11]:
```python
IQR=q3-q1
lower_fence=q1-1.5*(IQR)
higher_fence=q3+1.5*(IQR)
print("Lower Fence is =",lower_fence,"And Higher Fence is =",higher_fence)
print(" ")
print ("This tell us about the  outlier .The value which is Lower than 3.375 and value Higher than 74.375 is consider
```

```
Lower Fence is = 3.375 And Higher Fence is = 74.375

This tell us about the  outlier .The value which is Lower than 3.375 and value Higher than 74.375 is consider to be
the outliers.
```

-:Measure of Dispersion:-

-->Measures of dispersion, also known as measures of variability or spread, provide information about how the data points in a dataset are spread out from the central tendency (mean, median, etc.). These measures help to quantify the degree of variability or diversity within a dataset

-->Range: Difference between the maximum and minimum values, sensitive to outliers.

-->Interquartile Range (IQR): Range between the 75th and 25th percentiles, less affected by outliers.

-->Variance: Average of squared differences from the mean, providing overall spread.

-->Standard Deviation: Square root of variance, indicates average deviation from the mean.

-->Coefficient of Variation (CV): Ratio of standard deviation to mean, measures relative variability.

-->Mean Absolute Deviation (MAD): Average of absolute differences from the mean.

-->Percentiles: Values indicating specific data points within a distribution.

These measures help understand the spread and distribution of data points in a dataset.

1.Variance
2.Standard Deviation

In [12]:
```
statistics.variance(ages)
```
Out[12]:
```
795.2954545454545
```

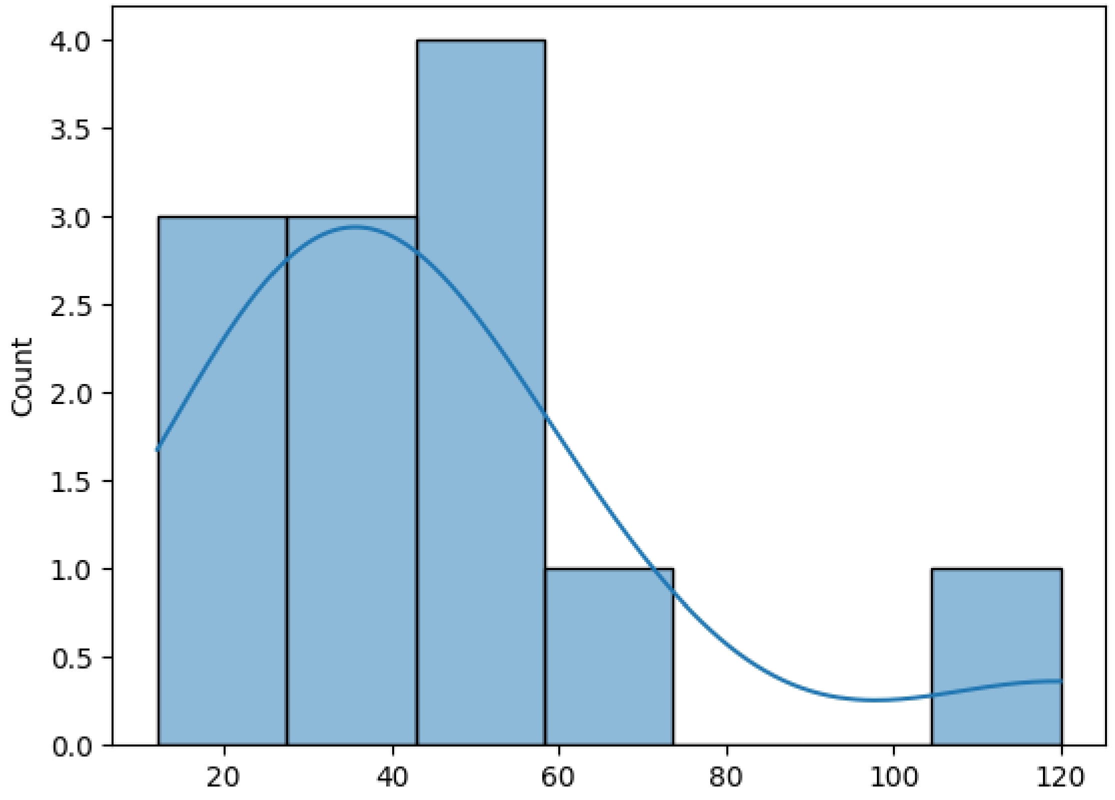
In [13]:
```
np.var(ages,axis=0)# It basically uses the Formula of Sample Variance.
```
Out[13]:
```
729.0208333333334
```

In [14]:
```
statistics.pvariance(ages) # Similer like --> np.var(ages,axis=0)
```
Out[14]:
```
729.0208333333334
```

-:Histogram and PDF:-

In [15]:
```
sns.histplot(ages,kde=True)#Probability Dinsity function
```

Out[15]:
```
<Axes: ylabel='Count'>
```



In [16]:
```
#practice on Dataset
#Here we loading A " IRIS" Dataset
```

In [17]:
```
df=sns.load_dataset('iris')
```

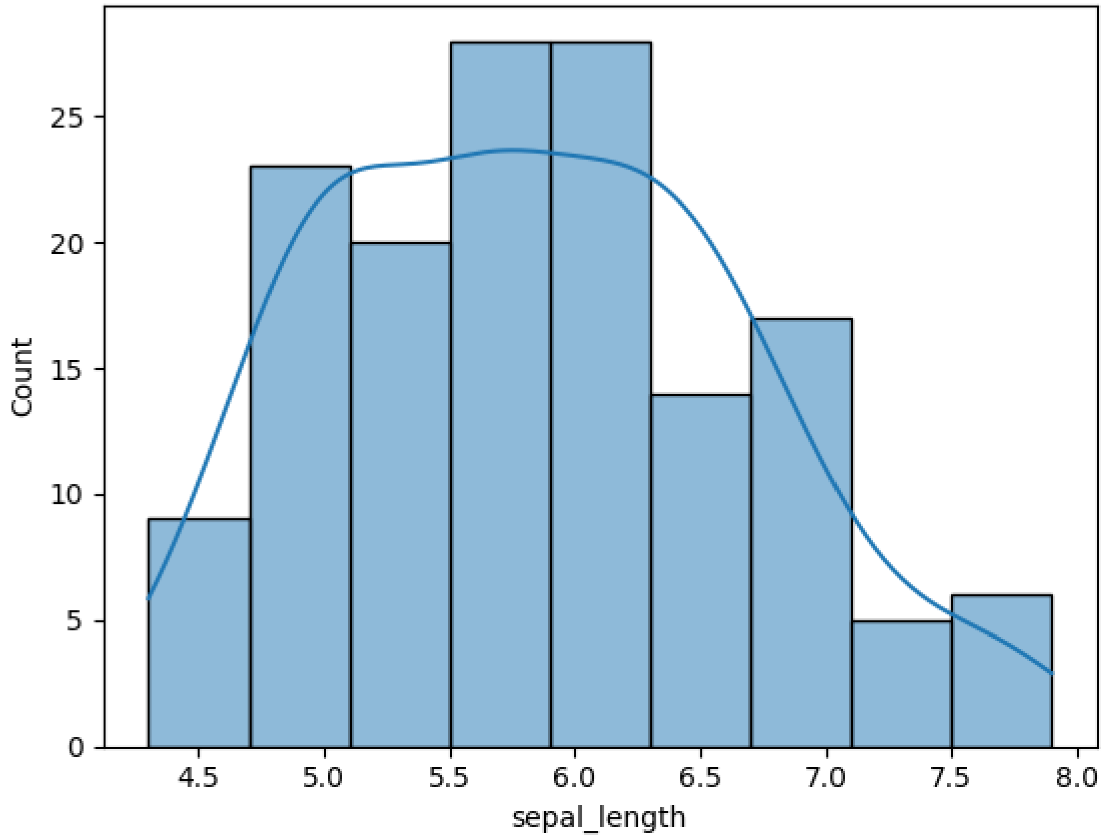
In [18]:
```
df.head(10)
```

Out[18]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa

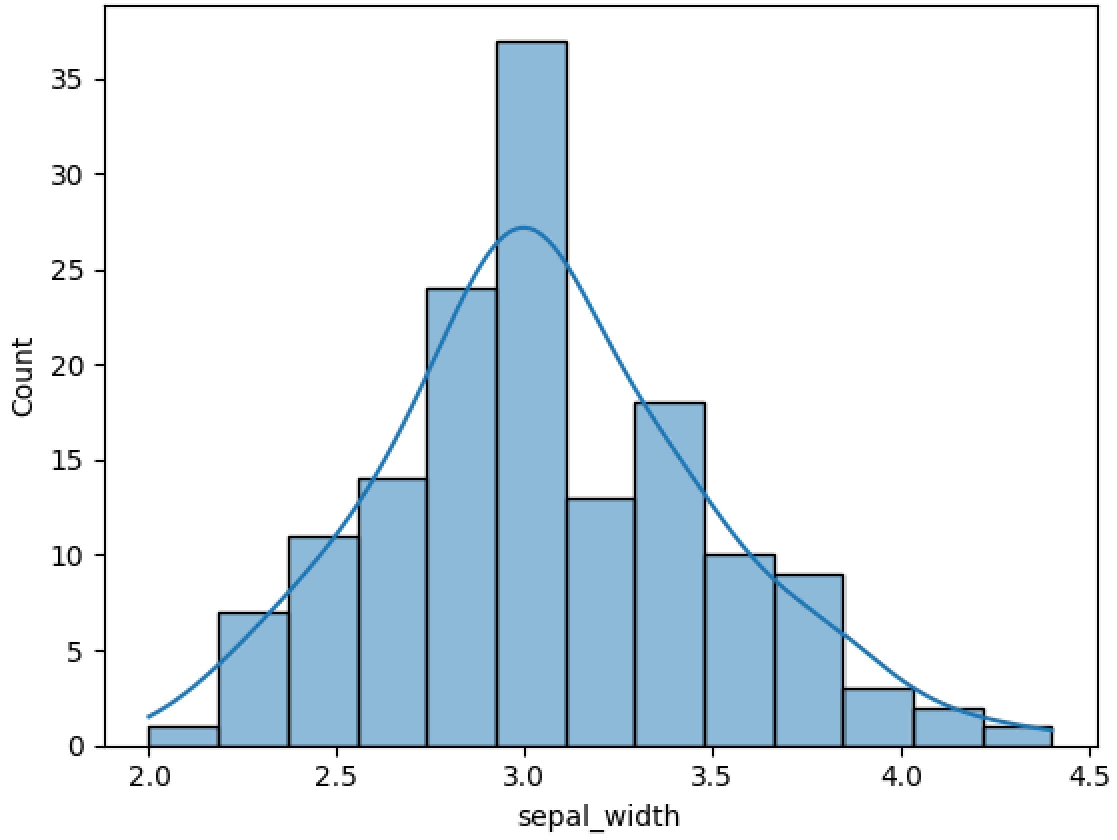
In [19]:
```
sns.histplot(df['sepal_length'],kde=True)
```

Out[19]:
```
<Axes: xlabel='sepal_length', ylabel='Count'>
```



```
In [20]: sns.histplot(df['sepal_width'],kde=True)# this graph is the fine Example of Normal Distribution .
```

```
Out[20]: <Axes: xlabel='sepal_width', ylabel='Count'>
```

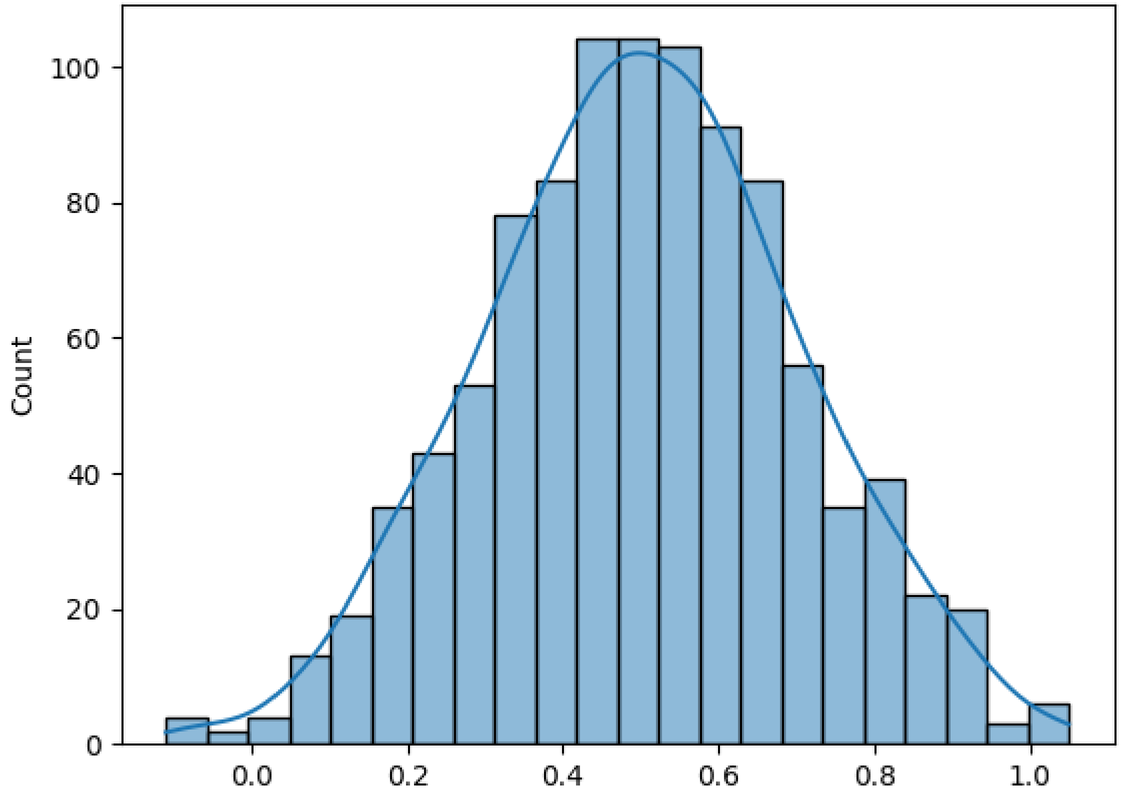


Create a Normal Distributed Dataset

-->A normal distribution, also called Gaussian distribution, is a symmetric bell-shaped pattern commonly seen in various natural and social phenomena. It's described by a mean and standard deviation. Many real-world measurements like heights and errors follow this pattern, making it crucial for statistical analysis and modeling.

In [21]:
```
s=np.random.normal(0.5,0.2,1000)
sns.histplot(s,kde=True)
```

Out[21]:
```
<Axes: ylabel='Count'>
```

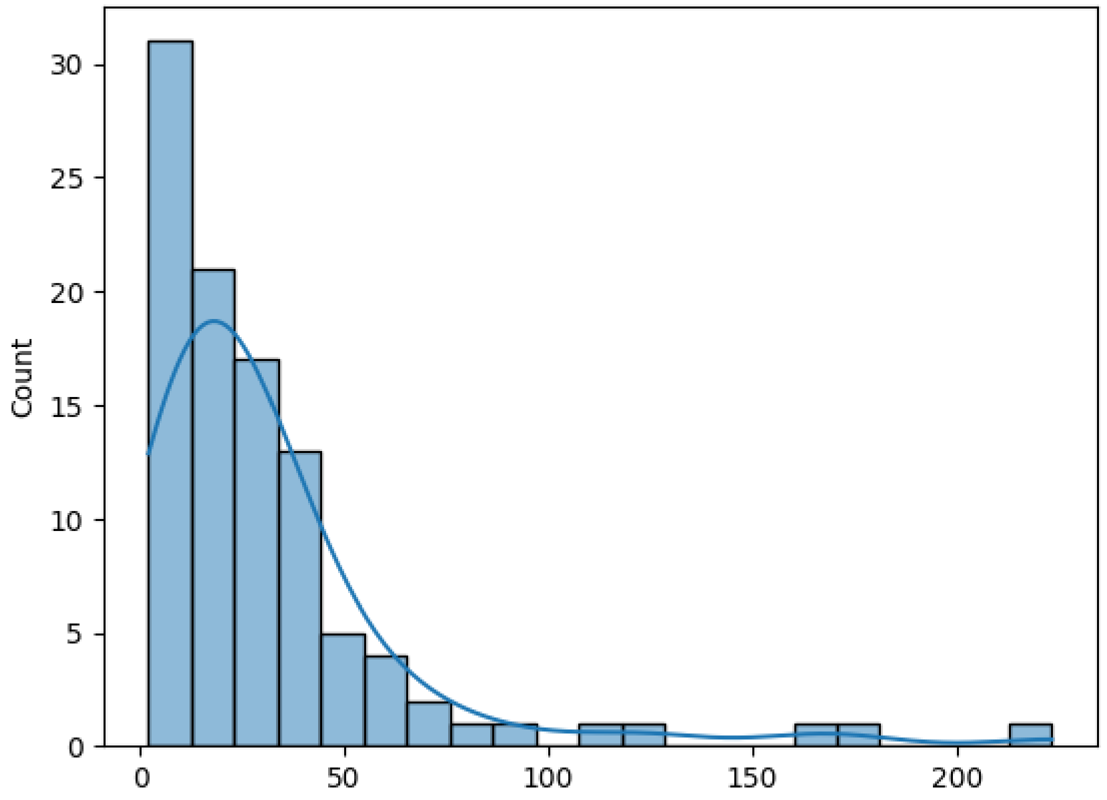


In [22]:
```
#Creating a Log Normal Distribution
```

```
-->A lognormal distribution describes data where the logarithm of the values follows a normal distribution. It's
used for positive, multiplicative quantities like stock prices or income, and results in a right-skewed
distribution.
```

In [23]:
```
mu,sigma =3.,1
s=np.random.lognormal(mu,sigma,100)
sns.histplot(s,kde=True)
```

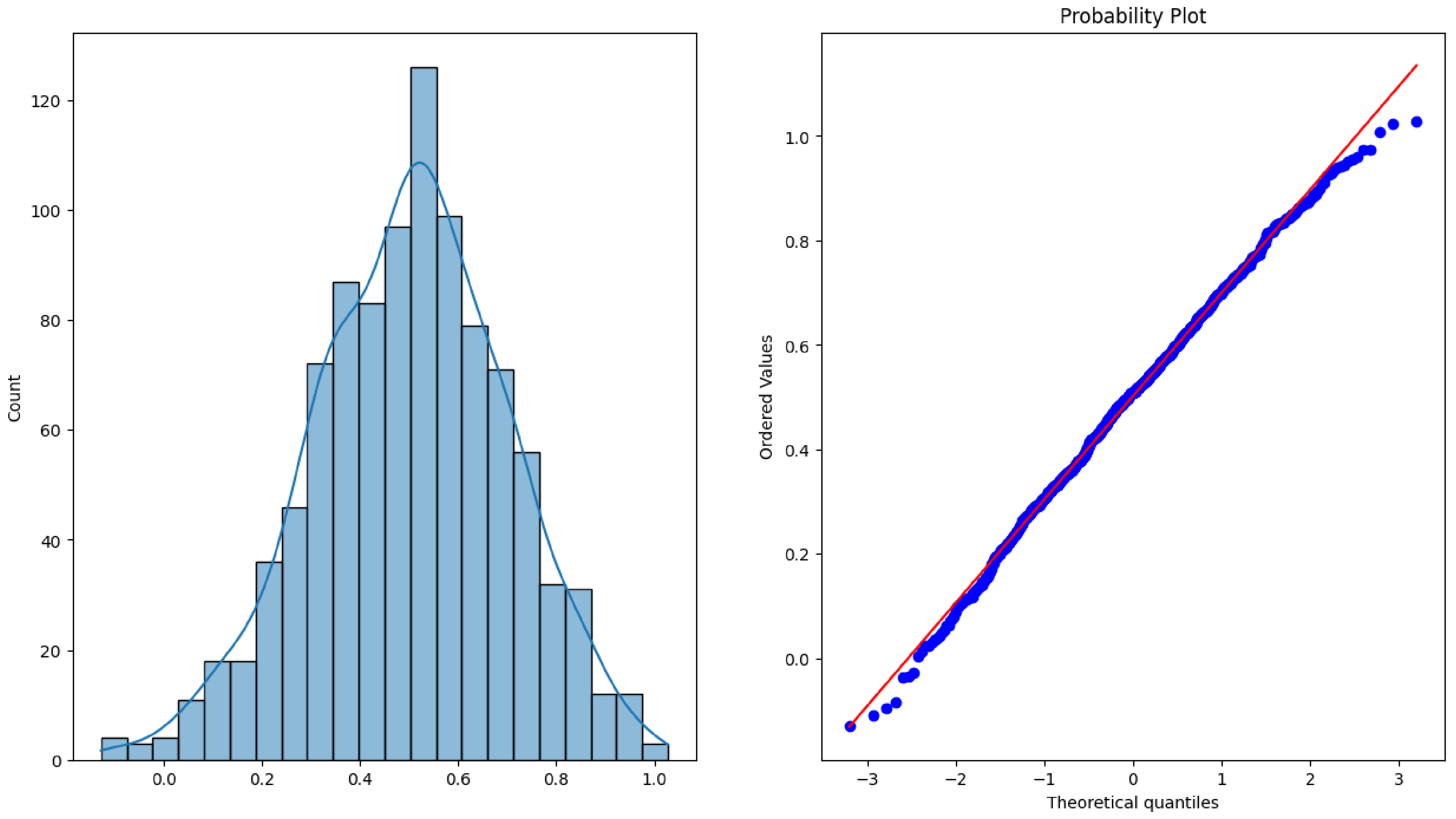
Out[23]:
```
<Axes: ylabel='Count'>
```



Check Weather a Distribution is Normal or Not?

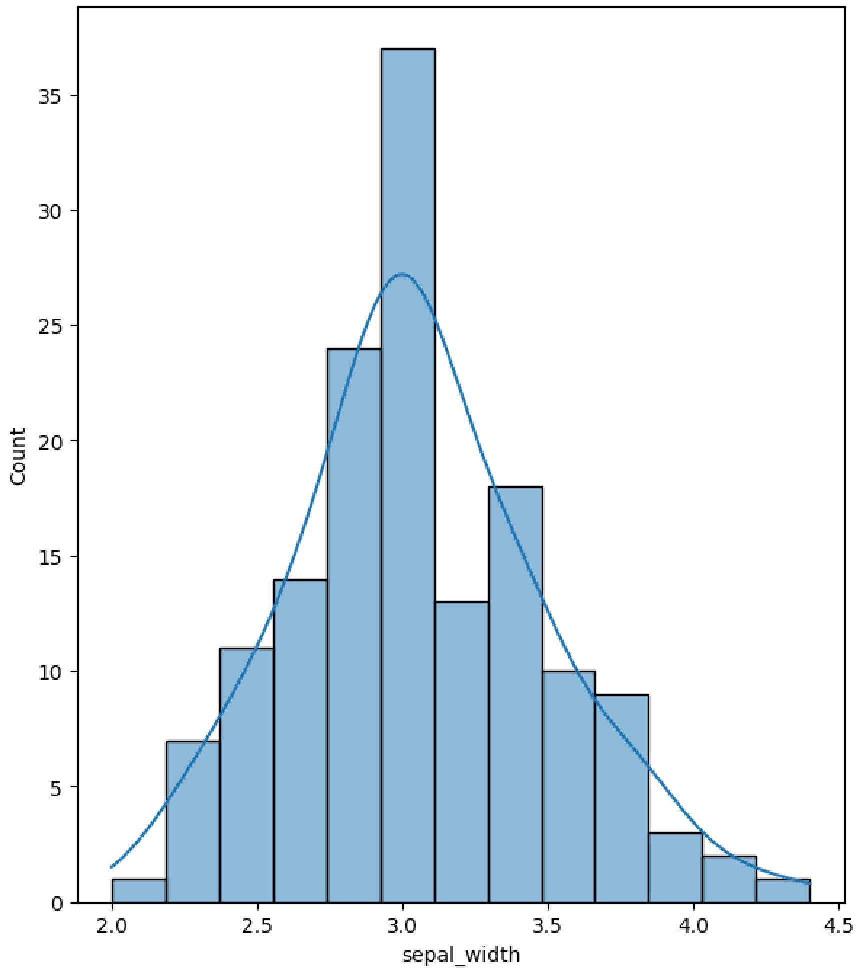
-->A Q-Q plot is a visual tool to check if data follows a specific theoretical distribution, like the normal distribution. If the points on the plot form a straight line, the data is likely normally distributed. Deviations suggest non-normality.

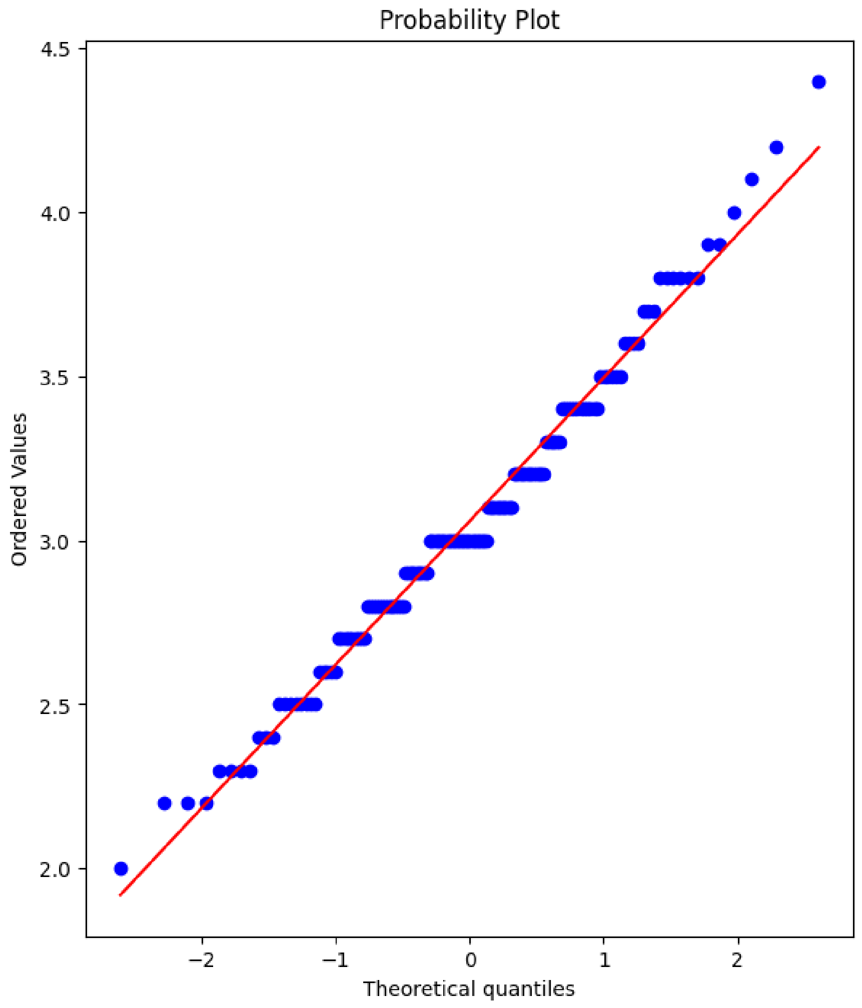
In [24]:
```
import matplotlib.pyplot as plt
import scipy.stats as stat
import pylab
def plot_data(sample):
    plt.figure(figsize=(15,8))
    plt.subplot(1,2,1)
    sns.histplot(sample,kde=True)
    plt.subplot(1,2,2)
    stat.probplot(sample,dist='norm',plot=pylab)
    plt.show

s=np.random.normal(0.5,0.2,1000)
plot_data(s)
```



In [25]:
```
# we can even check it for the Iris dataset
```

In [26]:
```
def plot_data(sample):
    plt.figure(figsize=(15,8))
    plt.subplot(1,2,1)
    sns.histplot(sample,kde=True)
    plt.subplot(1,2,2)
    stat.probplot(sample,dist='norm',plot=pylab)
    plt.show

#s=np.random.normal(0.5,0.2,1000)
plot_data(df['sepal_width'])
```

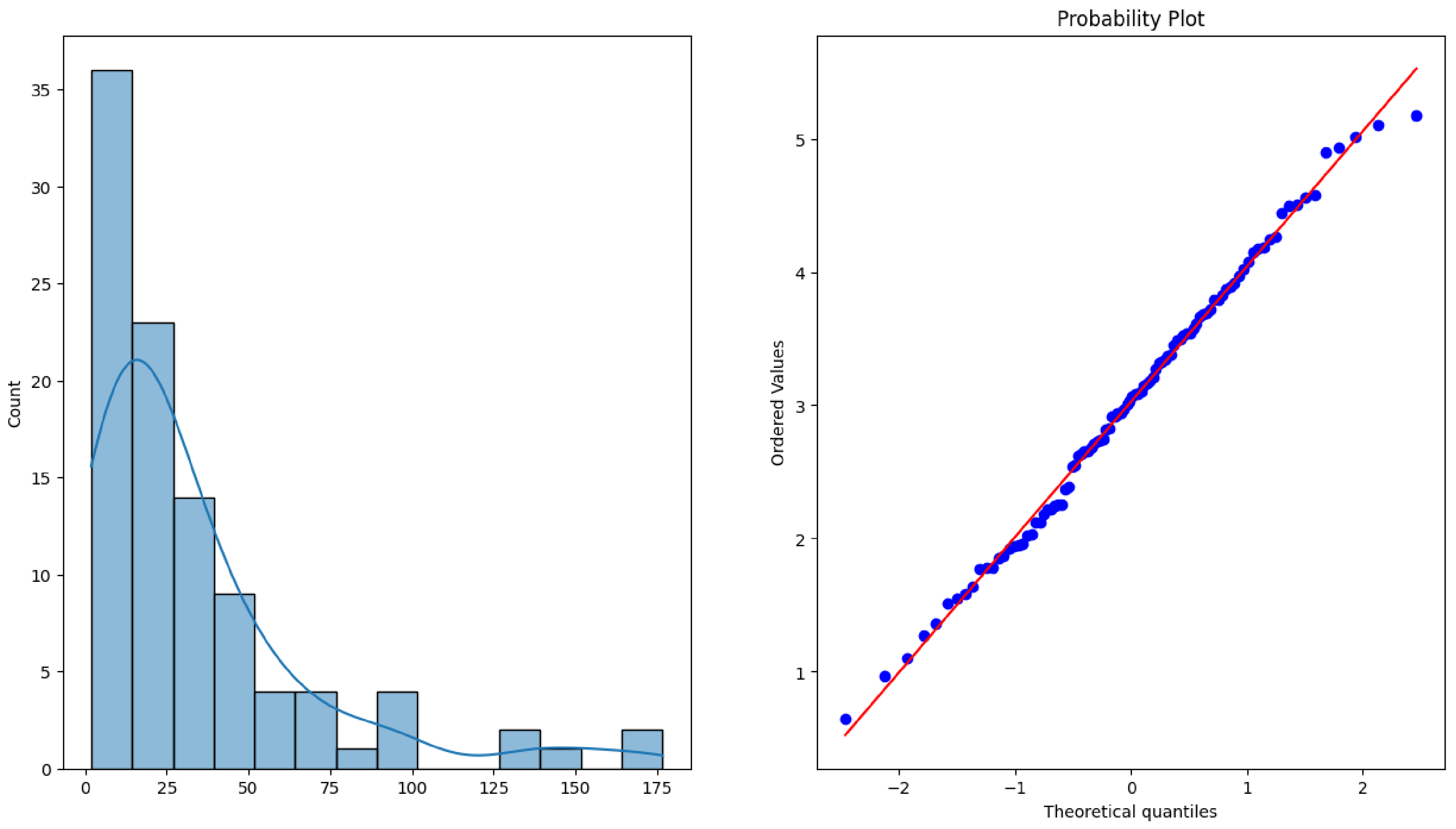




In [27]:
```
# In the same we can Check for the Lognormal Distributed data.
```

In [28]:
```
mu,sigma =3.,1
sample=np.random.lognormal(mu,sigma,100)
```

In [29]:
```
def plot_data_log(sample):
    plt.figure(figsize=(15,8))
    plt.subplot(1,2,1)
    sns.histplot(sample,kde=True)
    plt.subplot(1,2,2)
    stat.probplot(np.log(sample),dist='norm',plot=pylab)
    plt.show()

plot_data_log(sample)
```


Count
Probability Plot
Ordered Values
Theoretical quantiles