$$PMF(x) = P(X = x) = \binom{n}{x} \times p^x \times (1-p)^{n-x} \tag{3.3}$$

where

$$\binom{n}{x} = \frac{n!}{x!(n-x)!}$$

The CDF of a binomial distribution (probability that the number of success will be x *or less than* x out of n trials) is given by

$$CDF(x) = P(X \leq x) = \sum_{k=0}^{x} \binom{n}{k} \times p^k \times (1-p)^{n-k} \tag{3.4}$$

In Python, the *scipy.stats.binom* class provides methods to work with binomial distribution.

3.4.1 | Example of Binomial Distribution

Fashion Trends Online (FTO) is an e-commerce company that sells women apparel. It is observed that 10% of their customers return the items purchased by them for many reasons (such as size, color, and material mismatch). On a specific day, 20 customers purchased items from FTO. Calculate:

1. Probability that exactly 5 customers will return the items.
2. Probability that a maximum of 5 customers will return the items.
3. Probability that more than 5 customers will return the items purchased by them.
4. Average number of customers who are likely to return the items and the variance and the standard deviation of the number of returns.

We solve each of these as follows:

1. Probability that exactly 5 customers will return the items.

The function *stats.binom.pmf()* calculates PMF for binomial distribution and takes three parameters:

(a) Expected number of successful trials (5)
(b) Total number of trials (20)
(c) The probability of success (0.1)

Note: The values in the bracket indicate the value of the parameters.

```
from scipy import stats
stats.binom.pmf(5, 20, 0.1)
```

The corresponding probability is 0.03192, that is, the probability that exactly 5 customers will return the items is approximately 3%.

To visualize how the PMF varies with increasing number of successful trials, we will create a list of all possible number of successes (0 to 20) and corresponding PMF values and draw a bar plot as shown in Figure 3.1.

```
# range(0,21) returns all values from 0 to 20 (excluding 21)
pmf_df = pd.DataFrame({'success': range(0,21),
                       'pmf': list(stats.binom.pmf(range(0,21),
                       20, 0.1))})
# Creating a bar plot with number of success as x and pmf as y
sn.barplot(x = pmf_df.success, y = pmf_df.pmf)
plt.ylabel('pmf')
plt.xlabel('Number of items returned');
```

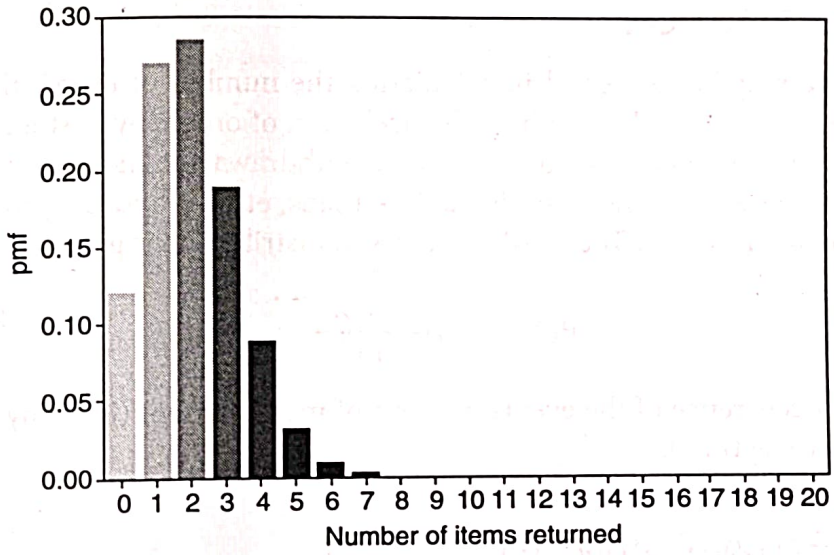

FIGURE 3.1 Binomial distribution.

2. Probability that a maximum of 5 customers will return the items.

The class *stats.binom.cdf()* computes the CDF for binomial distribution. In this case the cumulative distribution function returns the probability that a maximum of 5 customers will return items.

```
stats.binom.cdf(5, 20, 0.1)
```

The corresponding probability value is 0.9887.

3. Probability that more than 5 customers will return the items purchased by them.

Total probability of any number of customers returning items (including 0) is always equal to 1.0. So, the probability that more than 5 customers will return the items can be computed by subtracting the probability of a maximum of 5 customers will return items from 1.0. In other words, the probability that more than 5 customers will return the items can be obtained by computing CDF of 5 and then subtracting it from 1.0.

```
1 - stats.binom.cdf(5, 20, 0.1)
```

The corresponding probability value is 0.0112.

4. **Average number of customers who are likely to return the items and the variance and the standard deviation of the number of returns.**
   (a) Average of a binomial distribution is given by $n * p$
   (b) Variance of the binomial distribution is given by $n * p * (1 - p)$

```
mean, var = stats.binom.stats(20, 0.1)
print("Average: ", mean , " Variance:", var)
```

```
Average:  2.0  Variance: 1.8
```

3.5 | POISSON DISTRIBUTION

In many situations, we may be interested in calculating the number of events that may occur over a period of time or space. For example, number of cancellation of orders by customers at an e-commerce portal, number of customer complaints, number of cash withdrawals at an ATM, number of typographical errors in a book, number of potholes on Bangalore roads, etc. To find the probability of number of events, we use Poisson distribution. The PMF of a Poisson distribution is given by

$$P(X=k)=\frac{e^{-\lambda}\times\lambda^{k}}{k!} \tag{3.5}$$

where λ is the rate of occurrence of the events per unit of measurement (in many situations the unit of measurement is likely to be time).

3.5.1 | Example of Poisson Distribution

The number of calls arriving at a call center follows a Poisson distribution at 10 calls per hour.

1. Calculate the probability that the number of calls will be maximum 5.
2. Calculate the probability that the number of calls over a 3-hour period will exceed 30.

We solve each of these as follows:

1. Calculate the probability that a maximum of 5 calls will arrive at the call center.

As the number of calls arriving at the center follows Poisson distribution, we can use *stats.poisson.cdf* to calculate the probability value. It takes the following two parameters:

(a) First parameter: Number of events (in this case, 5 calls) for which the probability needs to be calculated.
(b) Second parameter: The average numbers of events (i.e., 10 calls per hour).

```
stats.poisson.cdf(5, 10)
```

```
0.06708
```

The corresponding probability is 0.067.

2. Calculate the probability that the number of calls over a 3-hour period will exceed 30.

Since the average calls per hour is 10 ($\lambda = 10$), and we are interested in finding the calls over 3 hours, the mean number of calls over 3 hours is $\lambda t = 30$. Probability that the number of calls will be more than 30 is given by

```
1 - stats.poisson.cdf(30, 30)
```

```
0.45164
```

The corresponding probability is 0.451.

To visualize the Poisson distribution for the average calls per hour as 10, we can plot PMF for all possible number of calls the call center can receive ranging from 0 to 30. We will create a DataFrame which will contain the number of calls ranging from 0 to 30 in one column named *success* and the corresponding PMFs in another column named *pmf*. The plotting is done using *barplot* in *seaborn* library.

```
# Range(0,30) returns all values from 0 to 30 (excluding 30)
pmf_df = pd.DataFrame({'success': range(0,30), 'pmf': list(stats.
                       poisson.pmf(range(0,30), 10))})
# Creating a barplot with number of calls as x and pmf as y
sn.barplot(x = pmf_df.success, y = pmf_df.pmf);
plt.xlabel('Number of Calls Received);
```

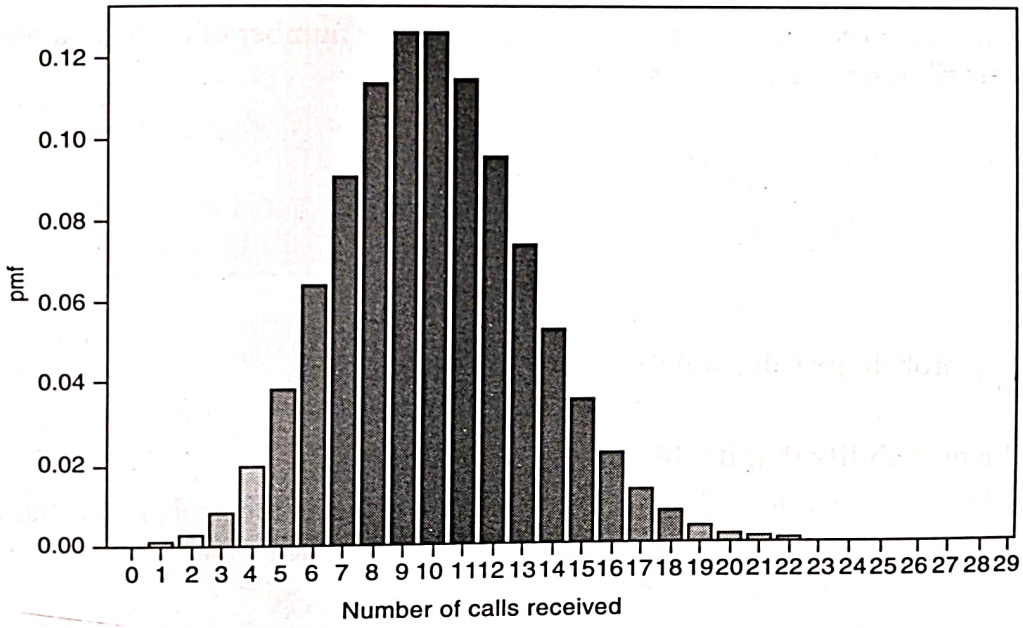

FIGURE 3.2 Poisson distribution.

The corresponding Poisson distribution plot is shown in Figure 3.2.

3.6 | EXPONENTIAL DISTRIBUTION

Exponential distribution is a single parameter continuous distribution that is traditionally used for modeling time-to-failure of electronic components. The exponential distribution represents a process in which events occur continuously and independently at a constant average rate.

The probability density function is given by

$$f(x) = \lambda e^{-\lambda x}, x \geq 0$$

where

1. The parameter λ is the scale parameter and represents the rate of occurrence of the event.
2. Mean of exponential distribution is given by $1/\lambda$.

3.6.1 | Example of Exponential Distribution

The time-to-failure of an avionic system follows an exponential distribution with a mean time between failures (MTBF) of 1000 hours. Calculate

1. The probability that the system will fail before 1000 hours.
2. The probability that it will not fail up to 2000 hours.
3. The time by which 10% of the system will fail (i.e., calculate P10 life).

We solve each of these as follows: Since time-to-failure is 1000 hours, so λ is 1/1000.

1. Calculate the probability that the system will fail before 1000 hours.

Cumulative distribution up to value 1000 for the exponential distribution will give the probability that the system will fail before 1000 hours. *stats.expon.cdf()* takes the number of hours and mean and scale of the exponential distribution as parameters to calculate CDF.

```
stats.expon.cdf(1000,
                loc = 1/1000,
                scale = 1000)
```

```
0.6321
```

The corresponding probability value is 0.6321.

2. Calculate the probability that it will not fail up to 2000 hours.

Probability that the system will not fail up to 2000 hours is same as the probability that the system will fail only beyond 2000 hours. This can be obtained by subtracting the probability that the system will fail up to 2000 hours from total probability (i.e. 1.0).

```
1 - stats.expon.cdf(2000,
                    loc = 1/1000,
                    scale = 1000)
```

```
0.1353
```

The corresponding probability value 0.1353.

3. Calculate the time by which 10% of the system will fail (i.e., calculate P10 life).

This can be calculated by ppf (percent point function) and is an inverse of CDF. *stats.expon.ppf* takes the percent point value and the mean and scale of the exponential distribution.

```
stats.expon.ppf(.1,
                loc = 1/1000,
                scale = 1000)
```

```
105.3615
```

That is, by 105.36 hours, 10% of systems will fail.

We can visualize the exponential distribution by plotting the PDF function against different time-to-failure hours. We will create a list of time-to-failure ranging from 100 to 5000 and then calculate and plot the PDF against those.

```
pdf_df = pd.DataFrame({'success': range(0,5000, 100),
                       'pdf':
                       list(stats.expon.pdf(range(0, 5000, 100),
                                            loc = 1/1000,
                                            scale = 1000))})

plt.figure(figsize=(10,4))
sn.barplot(x = pdf_df.success, y = pdf_df.pdf)
plt.xticks(rotation=90);
plt.xlabel('Time to failure');
```

The corresponding exponential distribution is shown in Figure 3.3.

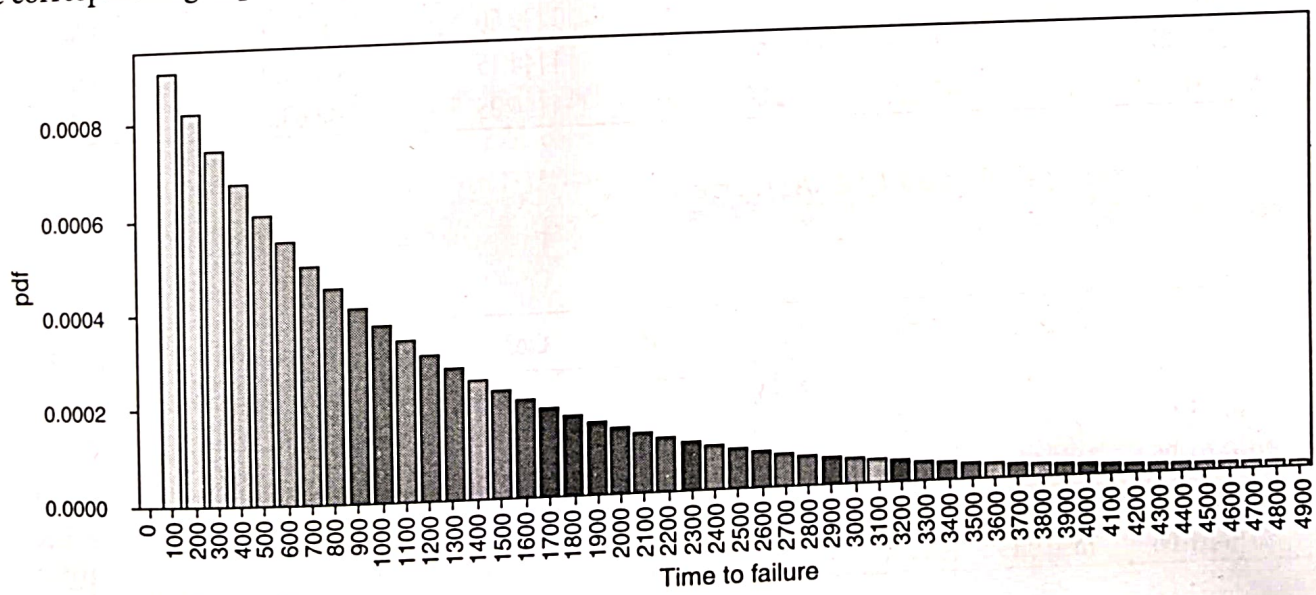

FIGURE 3.3 Exponential distribution.

3.7 | NORMAL DISTRIBUTION

Normal distribution, also known as *Gaussian distribution*, is one of the most popular continuous distribution in the field of analytics especially due to its use in multiple contexts. Normal distribution is observed across many naturally occurring measures such as age, salary, sales volume, birth weight, height, etc. It is also popularly known as bell curve (as it is shaped like a bell).

3.7.1 | Example of Normal Distribution

To understand normal distribution and its application, we will use daily returns of stocks traded in BSE (Bombay Stock Exchange). Imagine a scenario where an investor wants to understand the risks and returns associated with various stocks before investing in them. For this analysis, we will evaluate two stocks: *BEML* and *GLAXO*. The daily trading data (open and close price) for each stock is taken for the period starting from 2010 to 2016 from BSE site (*www.bseindia.com*).

First, we will load and prepare the data before getting back to the application of normal distribution.

```
import pandas as pd
import numpy as np
import warnings

beml_df = pd.read_csv('BEML.csv')
beml_df[0:5]
```

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2010-01-04	1121.0	1151.00	1121.00	1134.0	1135.60	101651.0	1157.18
1	2010-01-05	1146.8	1149.00	1128.75	1135.0	1134.60	59504.0	676.47
2	2010-01-06	1140.0	1164.25	1130.05	1137.0	1139.60	128908.0	1482.84
3	2010-01-07	1142.0	1159.40	1119.20	1141.0	1144.15	117871.0	1352.98
4	2010-01-08	1156.0	1172.00	1140.00	1141.2	1144.05	170063.0	1971.42

```
glaxo_df = pd.read_csv('GLAXO.csv')
glaxo_df[0:5]
```

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2010-01-04	1613.00	1629.10	1602.00	1629.0	1625.65	9365.0	151.74
1	2010-01-05	1639.95	1639.95	1611.05	1620.0	1616.80	38148.0	622.58
2	2010-01-06	1618.00	1644.00	1617.00	1639.0	1638.50	36519.0	595.09
3	2010-01-07	1645.00	1654.00	1636.00	1648.0	1648.70	12809.0	211.00
4	2010-01-08	1650.00	1650.00	1626.55	1640.0	1639.80	28035.0	459.11

The dataset contains daily Open and Close price along with daily High and Low prices, Total Trade Quantity, and Turnover (Lacs). Our discussion will involve only close price. The daily returns of a stock are calculated as the change in close prices with respect to the close price of yesterday.

Since our analysis will involve only daily close prices, so we will select *Date* and *Close* columns from the DataFrames.

```
beml_df = beml_df[['Date', 'Close']]
glaxo_df = glaxo_df[['Date', 'Close']]
```

Visualizing the daily close prices will show how stock prices have moved over time. To show the trend of close price, the rows should be ordered by time. The DataFrames have a date column, so we can create a *DatetimeIndex* index from this column *Date*. It will ensure that the rows are sorted by time in ascending order.

```
glaxo_df = glaxo_df.set_index(pd.DatetimeIndex(glaxo_df['Date']))
beml_df = beml_df.set_index(pd.DatetimeIndex(beml_df['Date']))
```

Let us display the first 5 records after the DataFrame is sorted by time to ensure that it is done correctly.

```
glaxo_df.head(5)
```

Date	Date	Close
2010-01-04	2010-01-04	1625.65
2010-01-05	2010-01-05	1616.80
2010-01-06	2010-01-06	1638.50
2010-01-07	2010-01-07	1648.70
2010-01-08	2010-01-08	1639.80

Now plot the trend of close prices of GLAXO stock using *plot()* method of matplotlib, which takes *glaxo_df.Close* as a parameter. The trend is shown in Figure 3.4.

```
import matplotlib.pyplot as plt
import seaborn as sn
%matplotlib inline

plt.plot(glaxo_df.Close);
plt.xlabel('Time');
plt.ylabel('Close Price');
```

Now plot BEML stock close price trend. The trend is shown in Figure 3.5.

```
plt.plot(beml_df.Close);
plt.xlabel('Time');
plt.ylabel('Close');
```

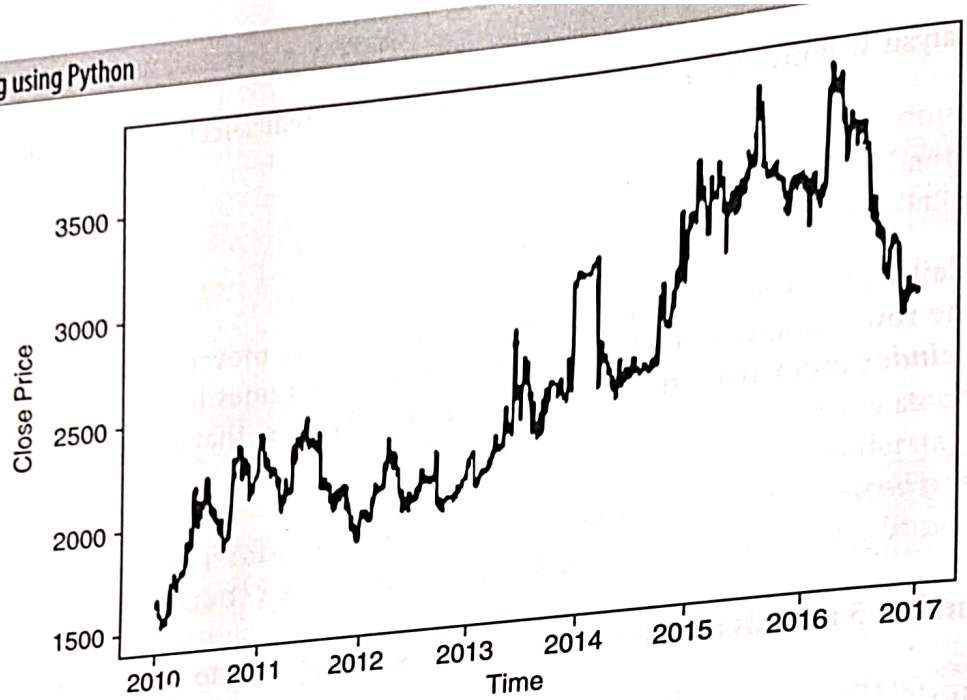

FIGURE 3.4 Close price trends of GLAXO stock.

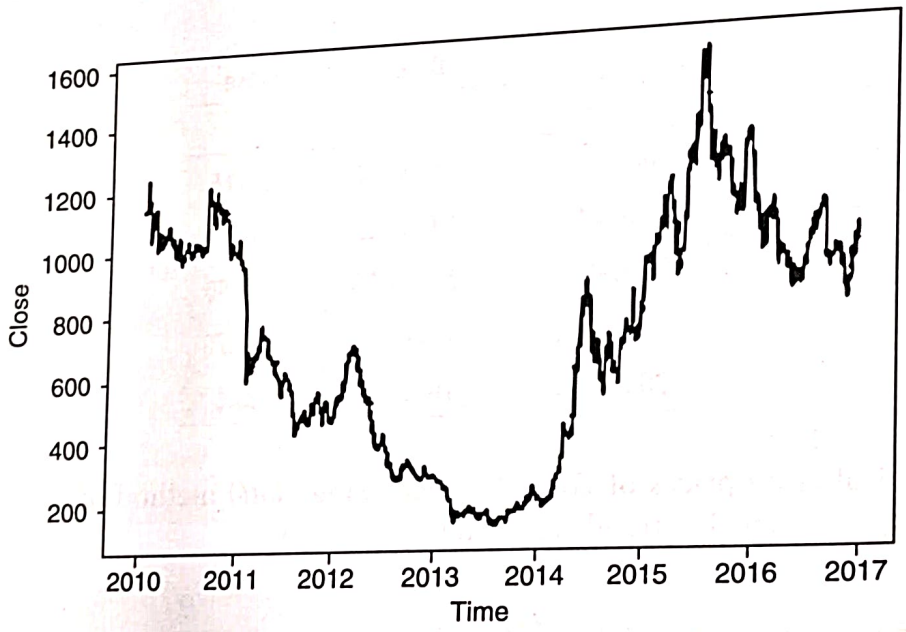

FIGURE 3.5 Close price trends of BEML stock.

It can be observed that there is an upward trend in the close price of Glaxo during 2010–2017 period. However, BEML had a downward trend during 2010–2013, followed by an upward trend since 2014 and then again a price correction from mid of 2015 (Figure 3.5).

What if a short-term (intraday) investor is interested in understanding the following characteristics about these stocks:

1. What is the expected daily rate of return of these stocks?
2. Which stocks have higher risk or volatility as far as daily returns are concerned?
3. What is the expected range of return for 95% confidence interval?

4. Which stock has higher probability of making a daily return of 2% or more?
5. Which stock has higher probability of making a loss (risk) of 2% or more?

To answer the above questions, we must find out the behavior of daily returns (we will refer to this as *gain* hence forward) on these stocks. The *gain* can be calculated as a percentage change in *close price*, from the previous day's *close price*.

$$gain = \frac{ClosePrice_t - ClosePrice_{t-1}}{ClosePrice_{t-1}}$$

The method *pct_change()* in Pandas will give the percentage change in a column value shifted by a period, which is passed as a parameter to *periods*. periods = 1 indicates the value change since last row, that is, the previous day.

```
glaxo_df['gain'] = glaxo_df.Close.pct_change(periods = 1)
beml_df['gain'] = beml_df.Close.pct_change(periods = 1)
glaxo_df.head(5)
```

Date	Date	Close	Gain
2010-01-04	2010-01-04	1625.65	NaN
2010-01-05	2010-01-05	1616.80	−0.005444
2010-01-06	2010-01-06	1638.50	0.013422
2010-01-07	2010-01-07	1648.70	0.006225
2010-01-08	2010-01-08	1639.80	−0.005398

The first day gain is shown as *NAN*, as there is no previous day for it to calculate *gain*. We can drop this record using the *dropna()* method.

```
glaxo_df = glaxo_df.dropna()
beml_df = beml_df.dropna()
```

Now, plot *gain* against time (Figure 3.6).

```
plt.figure(figsize = (8, 6));
plt.plot(glaxo_df.index, glaxo_df.gain);
plt.xlabel('Time');
plt.ylabel('gain');
```

The plot in Figure 3.6 shows that the daily gain is highly random and fluctuates around 0.00. The gain remains mostly between 0.05 and −0.05. However, very high gain close to 0.20 has been observed once and similarly, high loss of around 0.08 has been observed once. We can draw distribution plot of gain of both BEML and Glaxo stocks to gain better insight (see Figure 3.7).

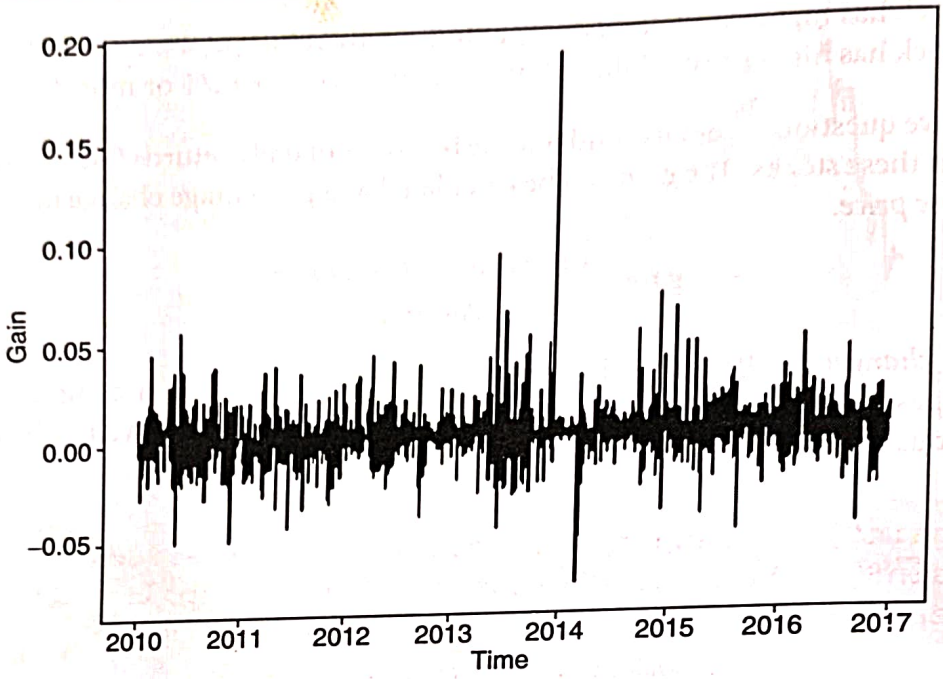

FIGURE 3.6 Daily gain of BEML stock.

```
sn.distplot(glaxo_df.gain, label = 'Glaxo');
sn.distplot(beml_df.gain, label = 'BEML');
plt.xlabel('gain');
plt.ylabel('Density');
plt.legend();
```

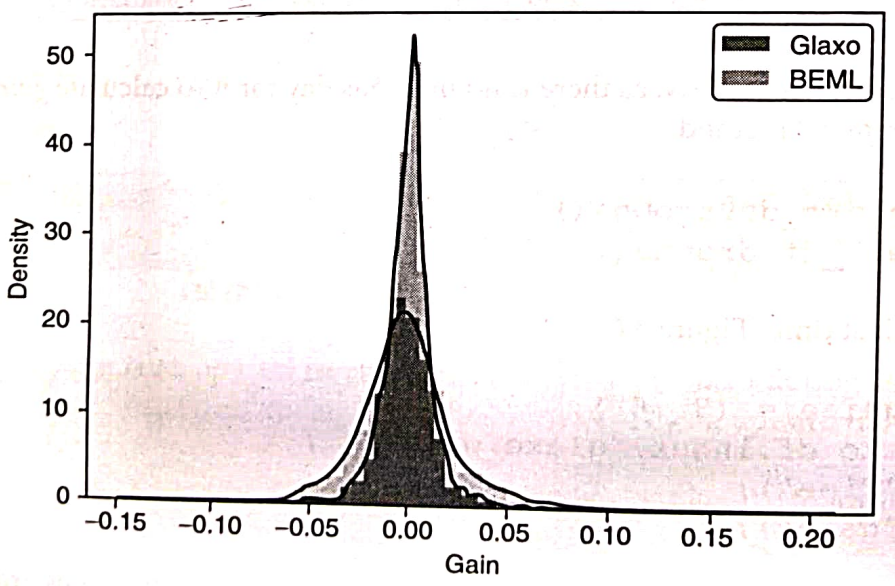

FIGURE 3.7 Distribution plot of daily gain of BEML and Glaxo stocks.

From the plot shown in Figure 3.7, *gain* seems to be normally distributed for both the stocks with a mean around 0.00. BEML seems to have a higher variance than Glaxo.

Note: This distribution has a long tail, but we will assume normal distribution for simplicity and discuss the example.

3.7.2 | Mean and Variance

The normal distribution is parameterized by two parameters: the mean of the distribution μ and the variance σ^2. The sample mean of a normal distribution is given by

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Variance is given by

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

The *standard deviation* is square root of variance and is denoted by σ.
Methods *mean()* and *std()* on DataFrame columns return mean and standard deviation, respectively. Mean and standard deviation for daily returns for Glaxo are:

```
print("Daily gain of Glaxo")
print("--------------------")
print("Mean: ", round(glaxo_df.gain.mean(), 4))
print("Standard Deviation: ", round(glaxo_df.gain.std(), 4))
```

```
Daily gain of Glaxo
--------------------
Mean:                0.0004
Standard Deviation:  0.0134
```

Mean and standard deviation for daily returns for BEML are:

```
print("Daily gain of BEML")
print("--------------------")
print("Mean: ", round(beml_df.gain.mean(), 4))
print("Standard Deviation: ", round(beml_df.gain.std(), 4))
```

```
Daily gain of BEML
--------------------
Mean:                0.0003
Standard Deviation:  0.0264
```

The *describe()* method of DataFrame returns the detailed statistical summary of a variables.

```
beml_df.gain.describe()
```

```
count      1738.000000
mean          0.000271
std           0.026431
min          -0.133940
25%          -0.013736
50%          -0.001541
75%           0.011985
max           0.198329
Name: gain, dtype: float64
```

The expected daily rate of return (gain) is around 0% for both stocks. Here variance or standard deviation of *gain* indicates risk. So, BEML stock has a higher risk as standard deviation of BEML is 2.64% whereas the standard deviation for Glaxo is 1.33%.