In this exercise we will fit the distributions for Normal, Exponential & Bernoulli distributions using random data and then we will find the accuracy of each distribution. ¶

First, we will import all the libraries which are needed.

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
from matplotlib import pyplot
import scipy.stats as stats
from scipy.stats import chisquare
import statsmodels.formula.api as smf
import random
from scipy.stats import norm
from scipy.stats import expon
from scipy.stats import bernoulli
from sklearn.metrics import mean_squared_error
```

1. Normal Distribution

Normal distribution is a function that represents the distribution of many random variables as a symmetrical bell-shaped graph.

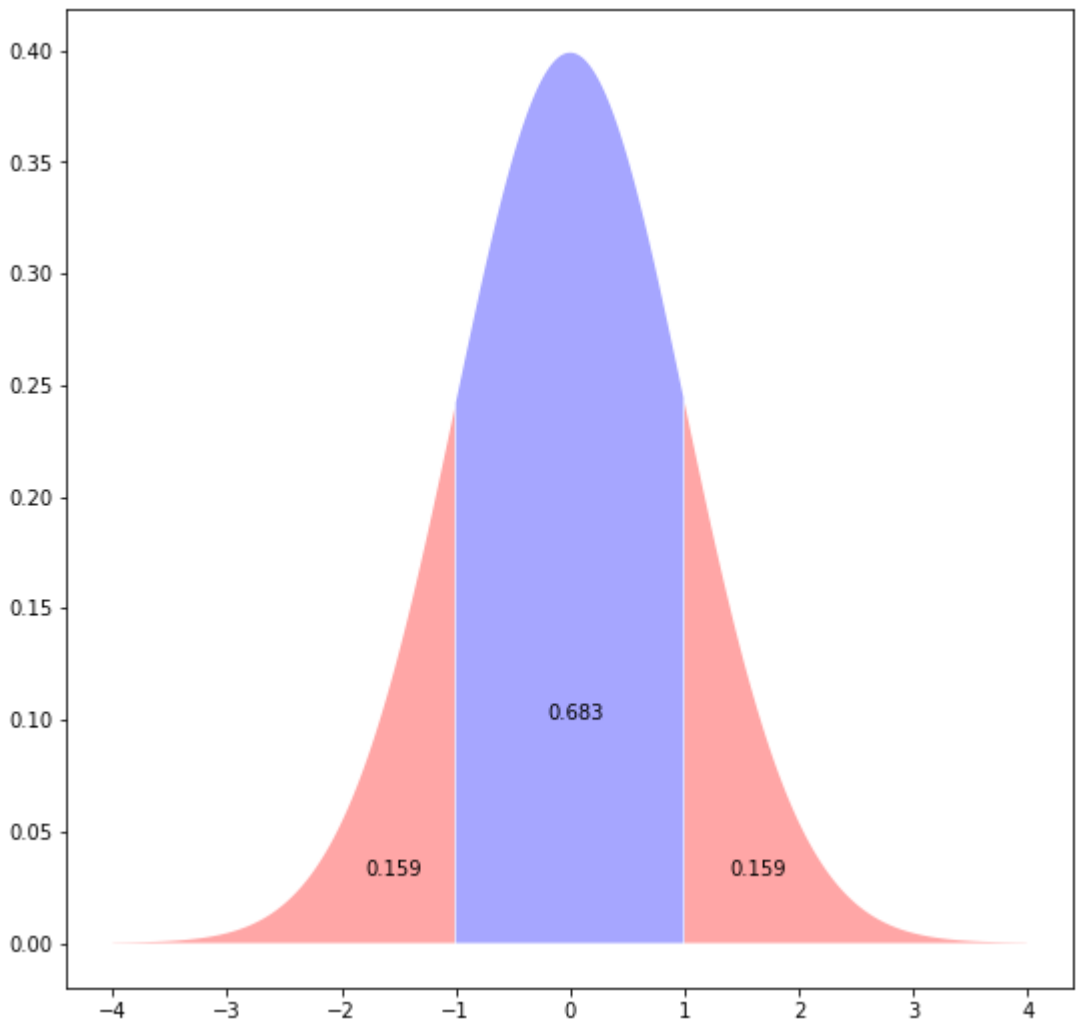
In [2]:

```
#we will set the three regions for our distribution

prob_under_minus1 = stats.norm.cdf(x= -1,
                                   loc = 0,
                                   scale= 1)

prob_over_1 = 1 - stats.norm.cdf(x= 1,
                                 loc = 0,
                                 scale= 1)

between_prob = 1-(prob_under_minus1+prob_over_1)

print(prob_under_minus1, prob_over_1, between_prob)
```

```
0.158655253931 0.158655253931 0.682689492137
```

In [3]:

```
# Now we will plot the bell shaped graph and color the regions accordingly

plt.rcParams["figure.figsize"] = (9,9)

plt.fill_between(x=np.arange(-4,-1,0.01),
                 y1= stats.norm.pdf(np.arange(-4,-1,0.01)) ,
                 facecolor='red',
                 alpha=0.35)

plt.fill_between(x=np.arange(1,4,0.01),
                 y1= stats.norm.pdf(np.arange(1,4,0.01)) ,
                 facecolor='red',
                 alpha=0.35)

plt.fill_between(x=np.arange(-1,1,0.01),
                 y1= stats.norm.pdf(np.arange(-1,1,0.01)) ,
                 facecolor='blue',
                 alpha=0.35)

plt.text(x=-1.8, y=0.03, s= round(prob_under_minus1,3))
plt.text(x=-0.2, y=0.1, s= round(between_prob,3))
plt.text(x=1.4, y=0.03, s= round(prob_over_1,3))
plt.show()
```



Now we will check the accuracy of our normal distribution

In [4]:

```
vals = norm.ppf([0.159, 0.683, 0.159])
np.allclose([0.159, 0.683, 0.159], norm.cdf(vals))
```

Out[4]:

```
True
```

2. Exponential Distribution

Exponential Distribution is the probability distribution that describes a process in which events occur continuously and independently at a constant average rate.

In [5]:

```
fig, ax = plt.subplots(1, 1)
```

In [6]:

```
mean, var, skew, kurt = expon.stats(moments='mvsk')
```

In [7]:

```
x = np.linspace(expon.ppf(0.01),
                expon.ppf(0.99), 100)
ax.plot(x, expon.pdf(x),
       'r-', lw=5, alpha=0.6, label='expon pdf')
```

Out[7]:

```
[<matplotlib.lines.Line2D at 0x10e835d30>]
```

In [8]:

```
rv = expon()
ax.plot(x, rv.pdf(x), 'k-', lw=2, label='frozen pdf')
```

Out[8]:

```
[<matplotlib.lines.Line2D at 0x106328710>]
```

In [9]:

```
#Accuracy of cdf and ppf

vals = expon.ppf([0.001, 0.5, 0.999])
np.allclose([0.001, 0.5, 0.999], expon.cdf(vals))
```

Out[9]:

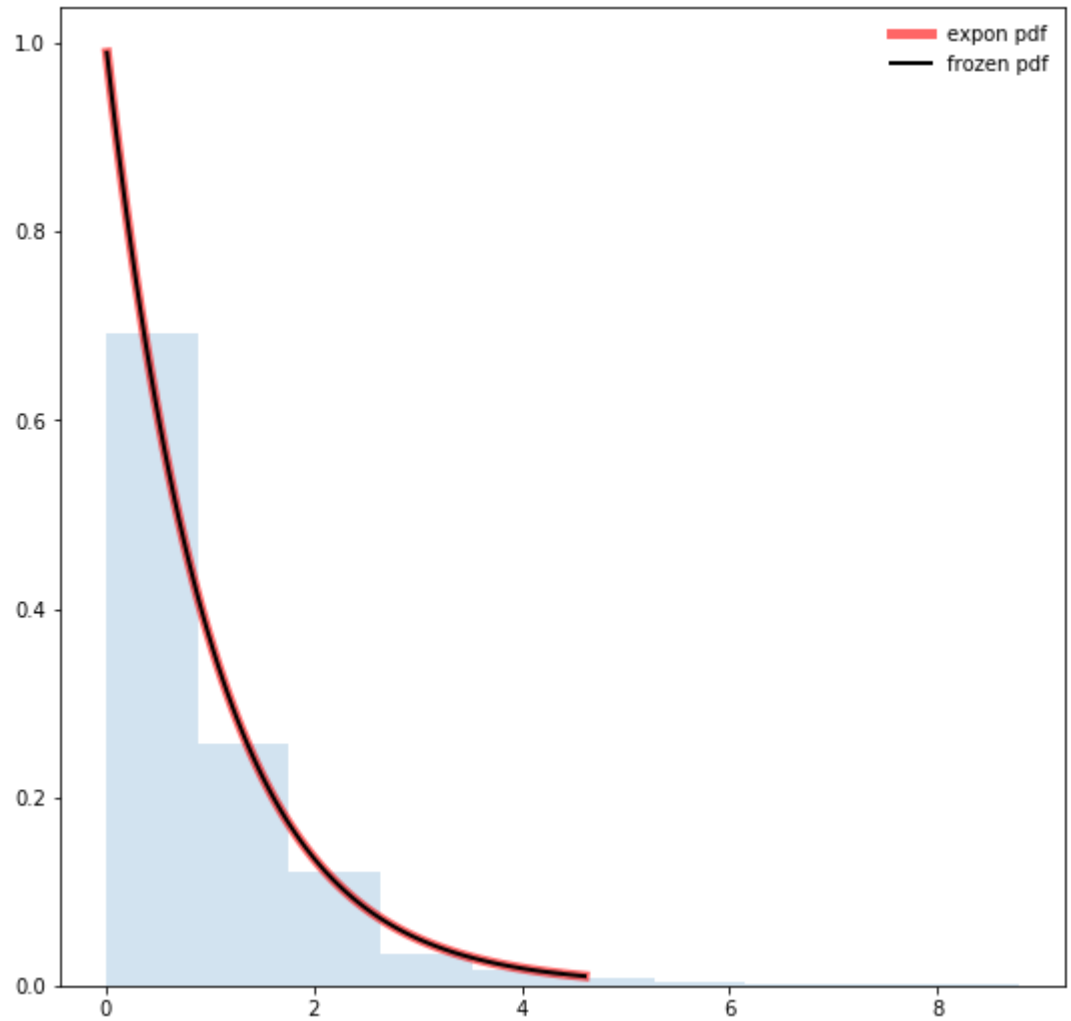
```
True
```

In [10]:

```
r = expon.rvs(size=1000)
```

In [11]:

```
#Now we will plot the exponential distribution

ax.hist(r, normed=True, histtype='stepfilled', alpha=0.2)
ax.legend(loc='best', frameon=False)
plt.show()
```



Now we will check the accuracy of our distribution

In [12]:

```
from sklearn.metrics import mean_squared_error
mean_squared_error(x, expon.cdf(x))
```

Out[12]:

3.5903807419483926

In [13]:

```
accuracy = 100 - mean_squared_error(x, expon.cdf(x))
accuracy
```

Out[13]:

96.409619258051606

3. Bernoulli Distribution

Bernoulli distribution is the probability distribution of a random variable which takes the value 1 with probability p and the value 0 with probability q=1-p.

In [14]:

```
fig, ax = plt.subplots(1, 1)
```

In [15]:

```
p = 0.3 #where p is a shape parameter
mean, var, skew, kurt = bernoulli.stats(p, moments='mvsk')
```

In [16]:

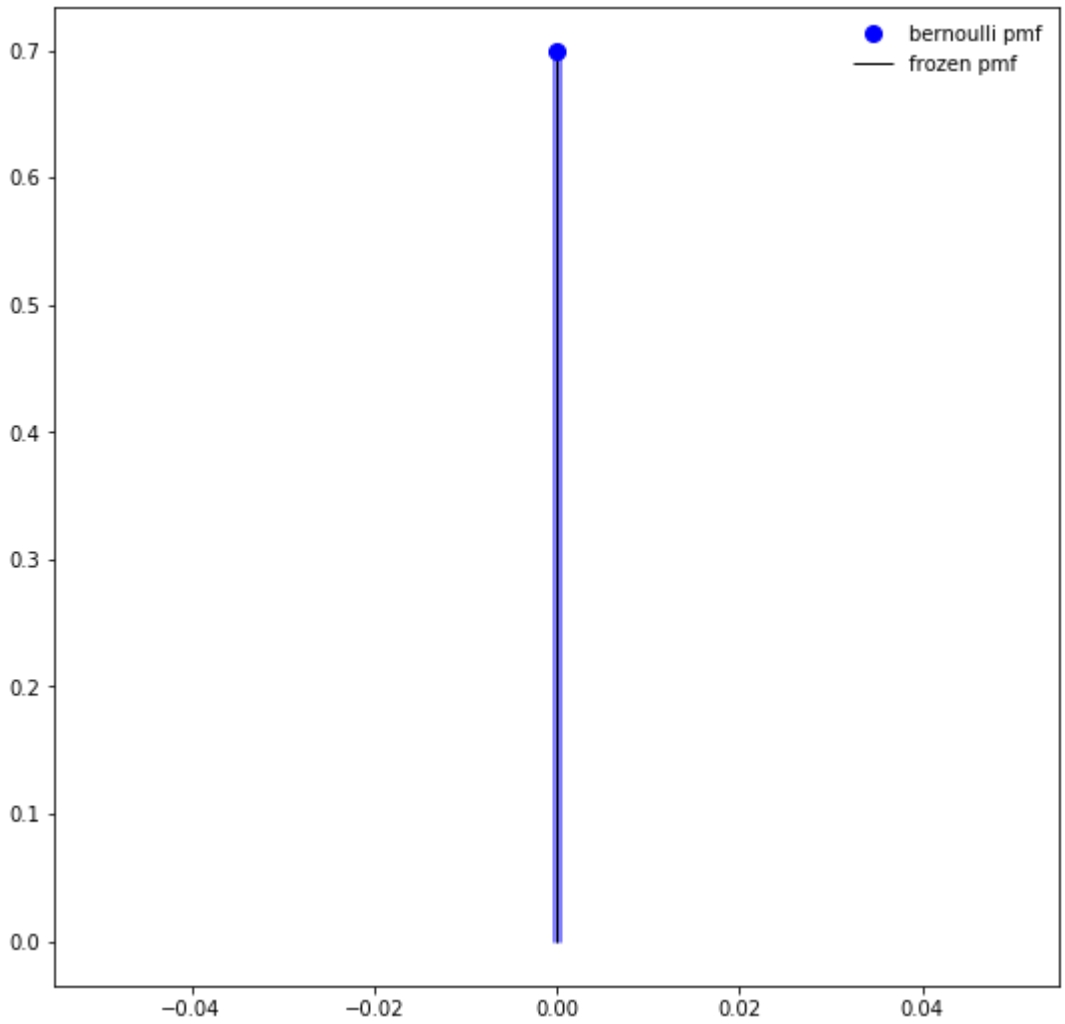
```
x = np.arange(bernoulli.ppf(0.01, p),
              bernoulli.ppf(0.99, p))
ax.plot(x, bernoulli.pmf(x, p), 'bo', ms=8, label='bernoulli pmf')
ax.vlines(x, 0, bernoulli.pmf(x, p), colors='b', lw=5, alpha=0.5)
```

Out[16]:

```
<matplotlib.collections.LineCollection at 0x10ea0f518>
```

In [17]:

```
rv = bernoulli(p)
ax.vlines(x, 0, rv.pmf(x), colors='k', linestyles='-', lw=1,
        label='frozen pmf')
ax.legend(loc='best', frameon=False)
plt.show()
```



In [18]:

```
prob = bernoulli.cdf(x, p)
np.allclose(x, bernoulli.ppf(prob, p))
```

Out[18]:

True

In [19]:

```
r = bernoulli.rvs(p, size=1000)
```

Now we will check the accuracy of our distribution