BM20BTECH11001-Lab 9

October 31, 2021

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import math
import scipy.special
from scipy.integrate import quad
```

[2]:
```
x = np.linspace(-10, 11, 10001)
```

0.0.1 Gaussian distribution

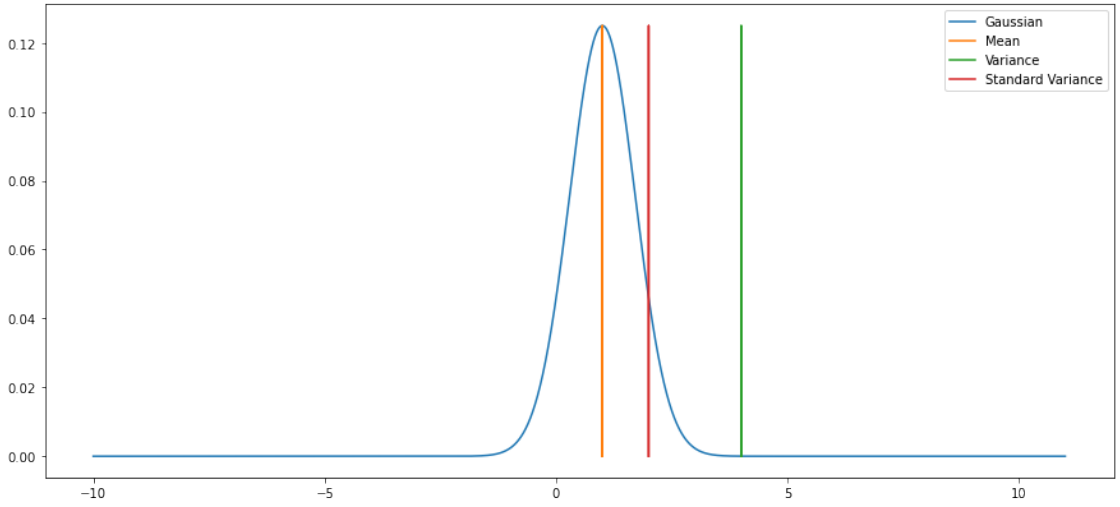
[3]:
```
mean = 1
sigma = 2
gaussian = [np.e**-((t-mean)**2)/(2*sigma**2) for t in x]
```

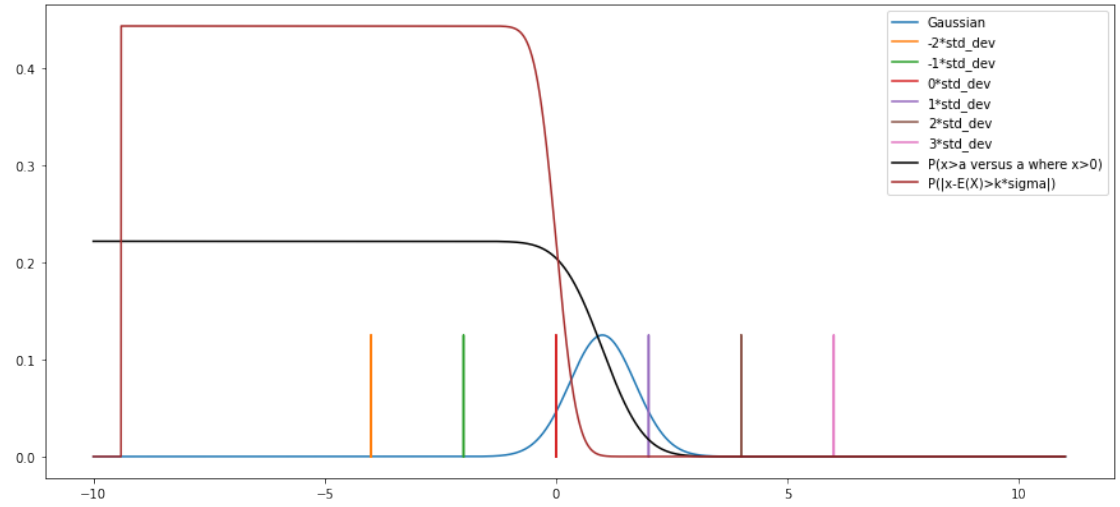
[4]:
```
fig, ax = plt.subplots(2, figsize=(15,15))
ax[0].plot(x, gaussian, label='Gaussian');
y = [1 for t in range(0, len(gaussian))]
ax[0].plot(y, gaussian, label="Mean");
variance = [4 for t in x]
ax[0].plot(variance, gaussian, label="Variance")
std = [2 for t in x]
ax[0].plot(std, gaussian, label="Standard Variance")
std_dev = 2
ax[1].plot(x, gaussian, label="Gaussian")

for k in range(-2, 4):
    distances = [k*std_dev for a in range(0, len(gaussian))]
    ax[1].plot(distances, gaussian, label = str(k)+"*std_dev")
a = np.linspace(1, 1000, 1000)

def integrand(x):
    return np.e**-((x-mean)**2)/(2*sigma**2)
I = [quad(integrand, t, np.inf)[0] for t in x]
ax[1].plot(x, I, color="black",label="P(x>a versus a where x>0)");
```

```
I2 = [quad(integrand, mean+(t*sigma), np.inf)[0] + quad(integrand, -np.inf,␣
 ↪mean-(t*sigma))[0] for t in x]
ax[1].plot(x, I2, color='brown', label="P(|x-E(X)>k*sigma|)");
leg_1 = ax[0].legend()
leg_2 = ax[1].legend()
```



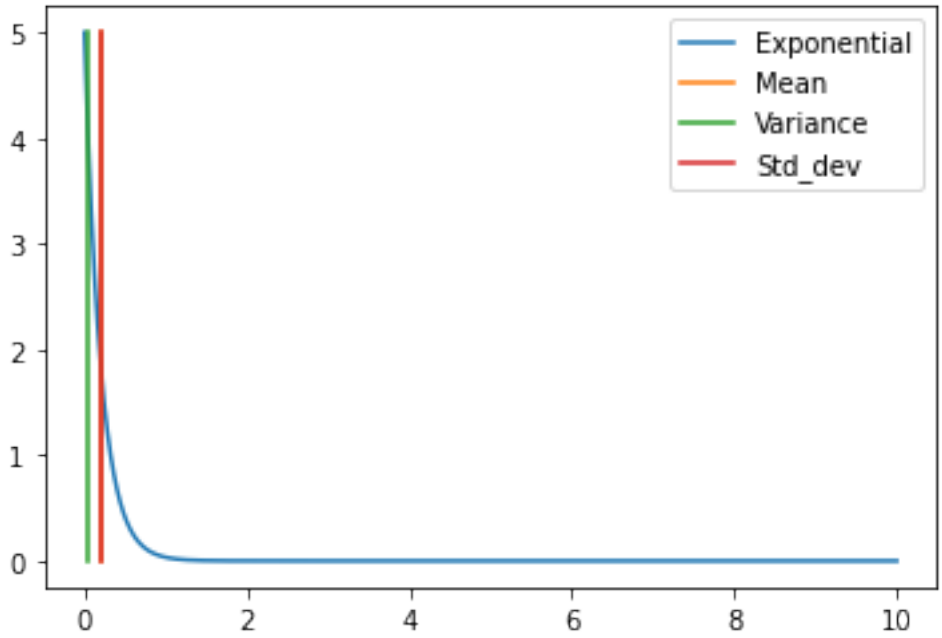


0.0.2 Exponential distribution

[5]:
```
mean = 1/5
sigma = 1/5
```

[6]:
```
x = np.linspace(0,10, 1001)
t = 5
exponential = [t*np.e**(-t*k) for k in x]
```

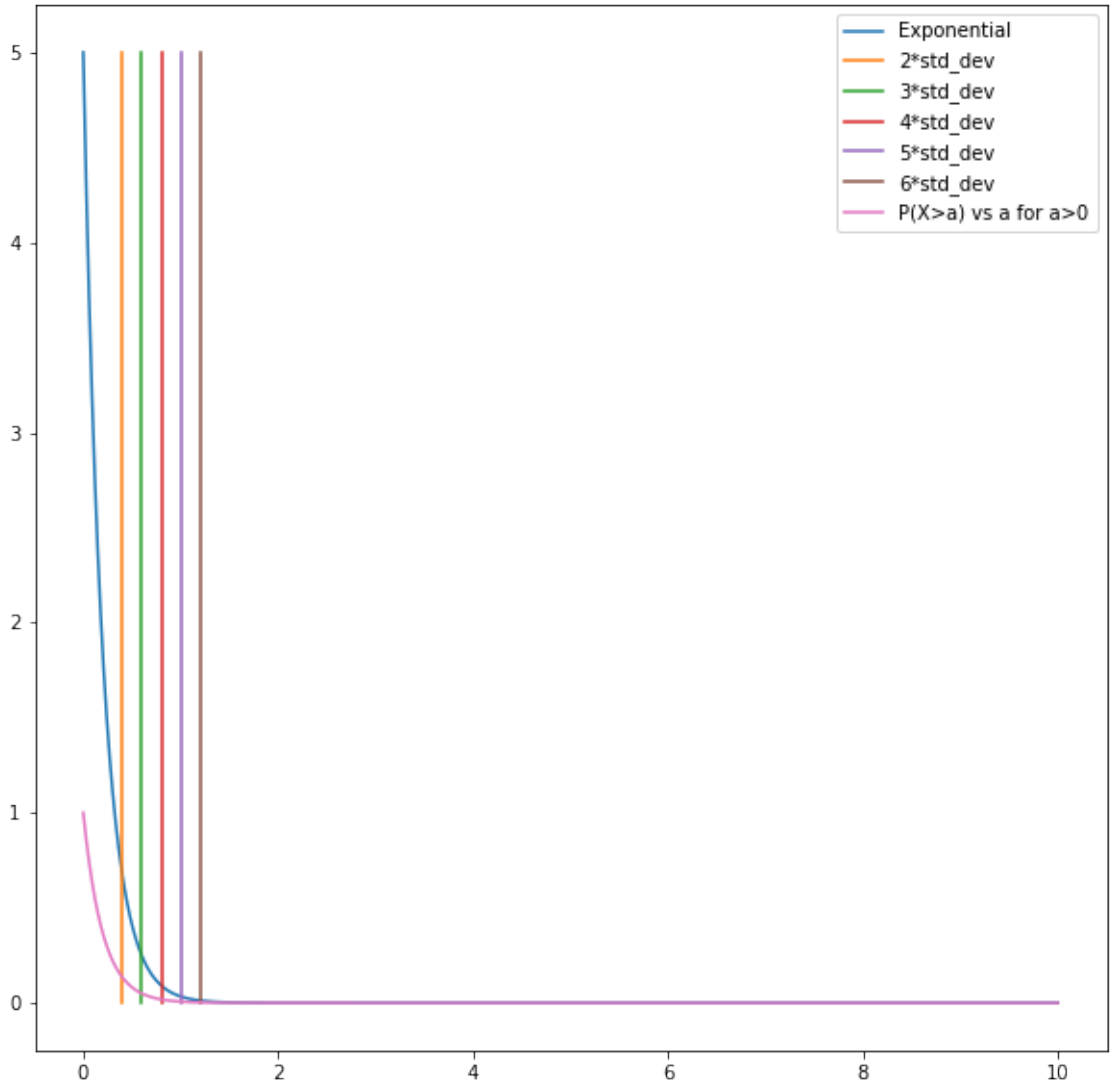
```
mean_array = [mean for a in x]
variance = [sigma**2 for a in x]
std_dev = [sigma for a in x]
plt.plot(x, exponential, label='Exponential')
plt.plot(mean_array, exponential, label="Mean")
plt.plot(variance, exponential, label="Variance")
plt.plot(std_dev, exponential, label="Std_dev");
leg = plt.legend()
```



[7]:
```
fig, ax = plt.subplots(1, figsize=(10, 10))
ax.plot(x, exponential, label="Exponential")
for k in range(2, 7):
    distances = [k*sigma for a in range(0, len(exponential))]
    ax.plot(distances, exponential, label=str(k)+"*std_dev")
a = np.linspace(1, 1000, 1000)

def integrand(x):
    return t*np.e**(-t*x)
I = [quad(integrand, b, np.inf)[0] for b in x]


ax.plot(x, I, label="P(X>a) vs a for a>0");
leg1 = ax.legend()
```



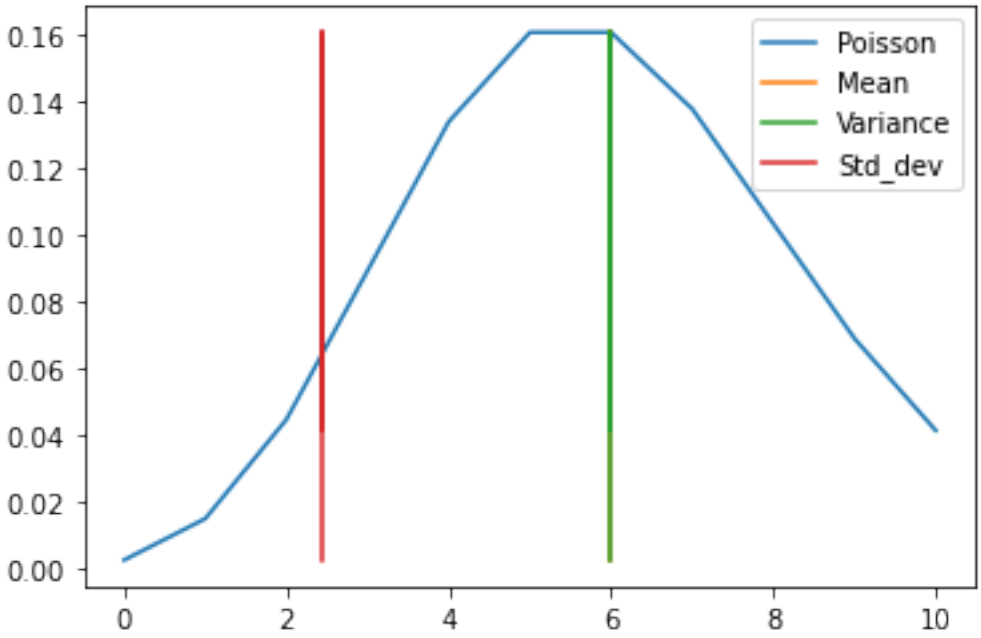
0.0.3 Poisson distribution

[8]:
```
x = np.linspace(0,10, 11)
Lambda = 6
poisson = [((np.e**(-Lambda))*(Lambda**t)/math.factorial(t)) for t in x]
```

[9]:
```
mean = Lambda
std_dev = Lambda**(1/2)
variance = Lambda
```

[10]:
```
mean_array = [mean for a in x]
variance_arr = [variance for a in x]
std_dev_arr = [std_dev for a in x]
```

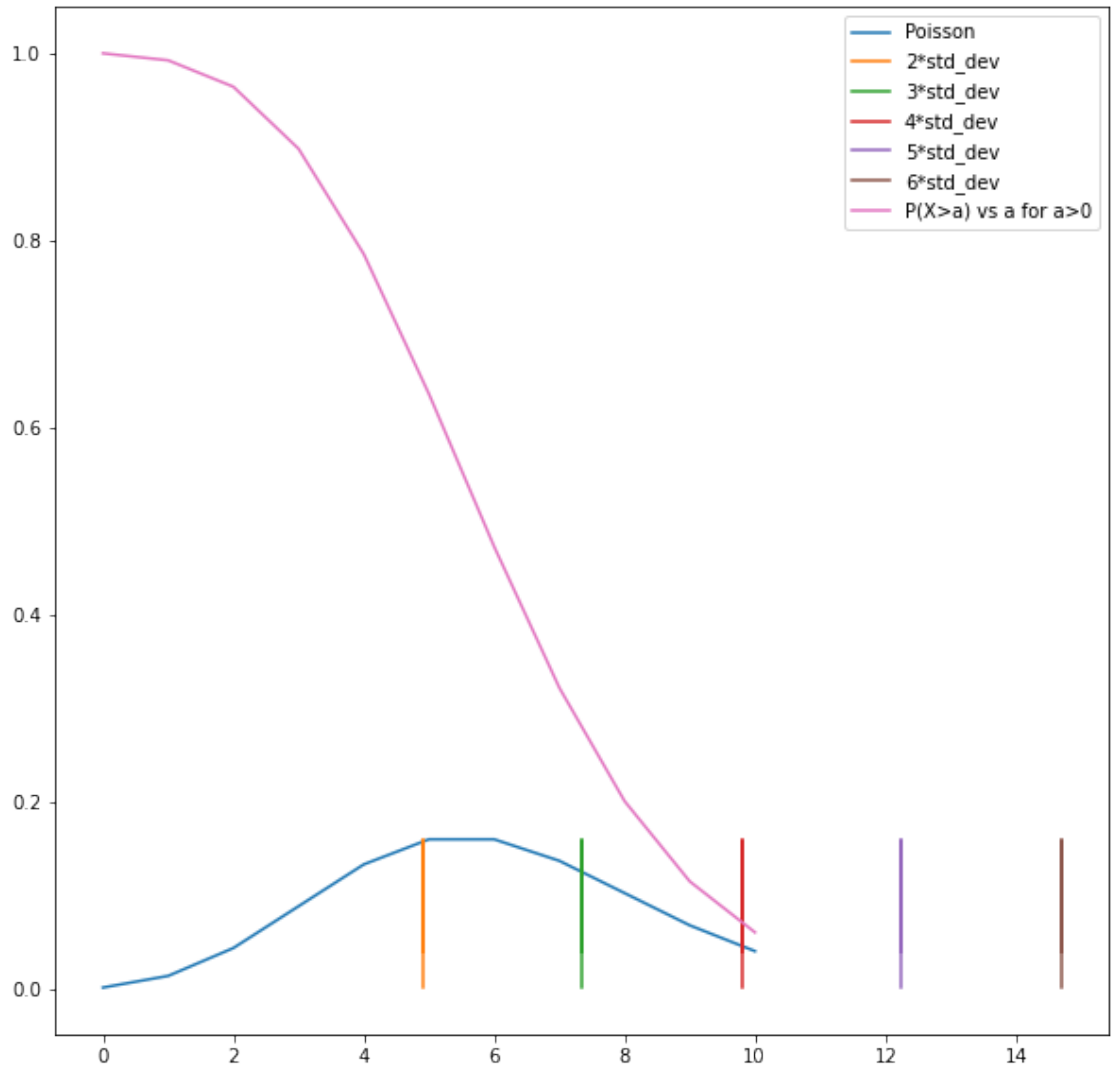
```
plt.plot(x, poisson, label="Poisson")
plt.plot(mean_array, poisson, label="Mean")
plt.plot(variance_arr, poisson, label="Variance")
plt.plot(std_dev_arr, poisson, label="Std_dev");
leg = plt.legend()
```



[11]:
```
fig, ax = plt.subplots(1, figsize=(10, 10))
ax.plot(x, poisson, label="Poisson")
for k in range(2, 7):
    distances = [k*std_dev for a in range(0, len(poisson))]
    ax.plot(distances, poisson, label=str(k)+"*std_dev")
a = np.linspace(1, 1000, 1000)

def integrand(x):
    return (np.e**(-Lambda))*(Lambda**x)/math.gamma(x+1)
I = [quad(integrand, b, 100)[0] for b in x]

ax.plot(x, I, label="P(X>a) vs a for a>0");
leg = ax.legend()
```

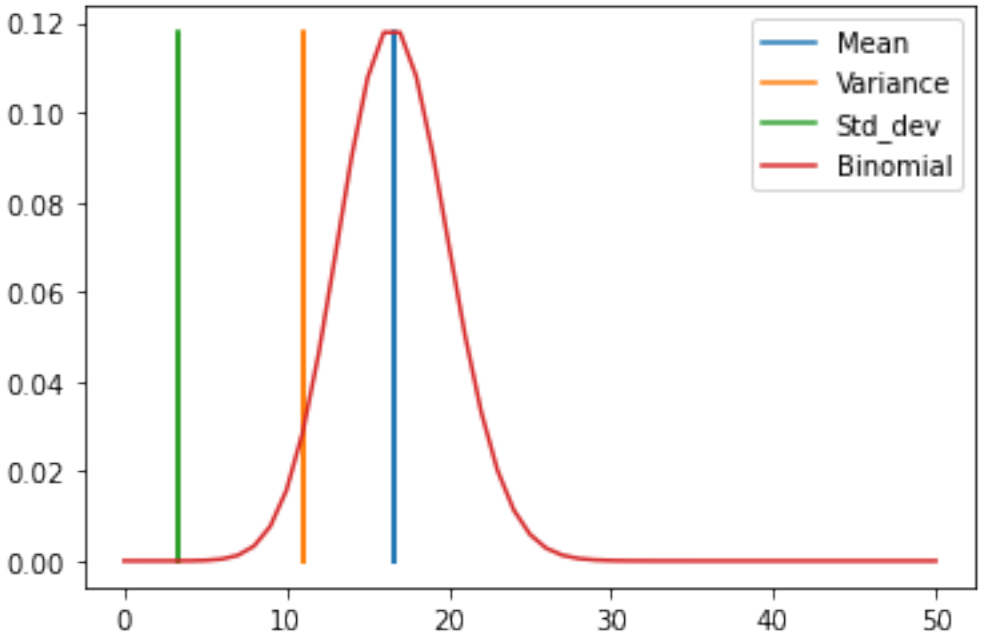


0.0.4 Binomial distribution

[12]:
```
x = np.linspace(0,50, 51)
def combination(n, x):
    return
p = 1/3
binomial = [(math.factorial(50)/((math.factorial(t))*(math.
 ↪factorial(50-t))))*(p**t)*((1-p)**(50-t)) for t in x]
```

[13]:
```
mean = 50*p
variance = 50*p*(1-p)
std_dev = variance**0.5
mean_array = [mean for a in x]
variance_arr = [variance for a in x]
```

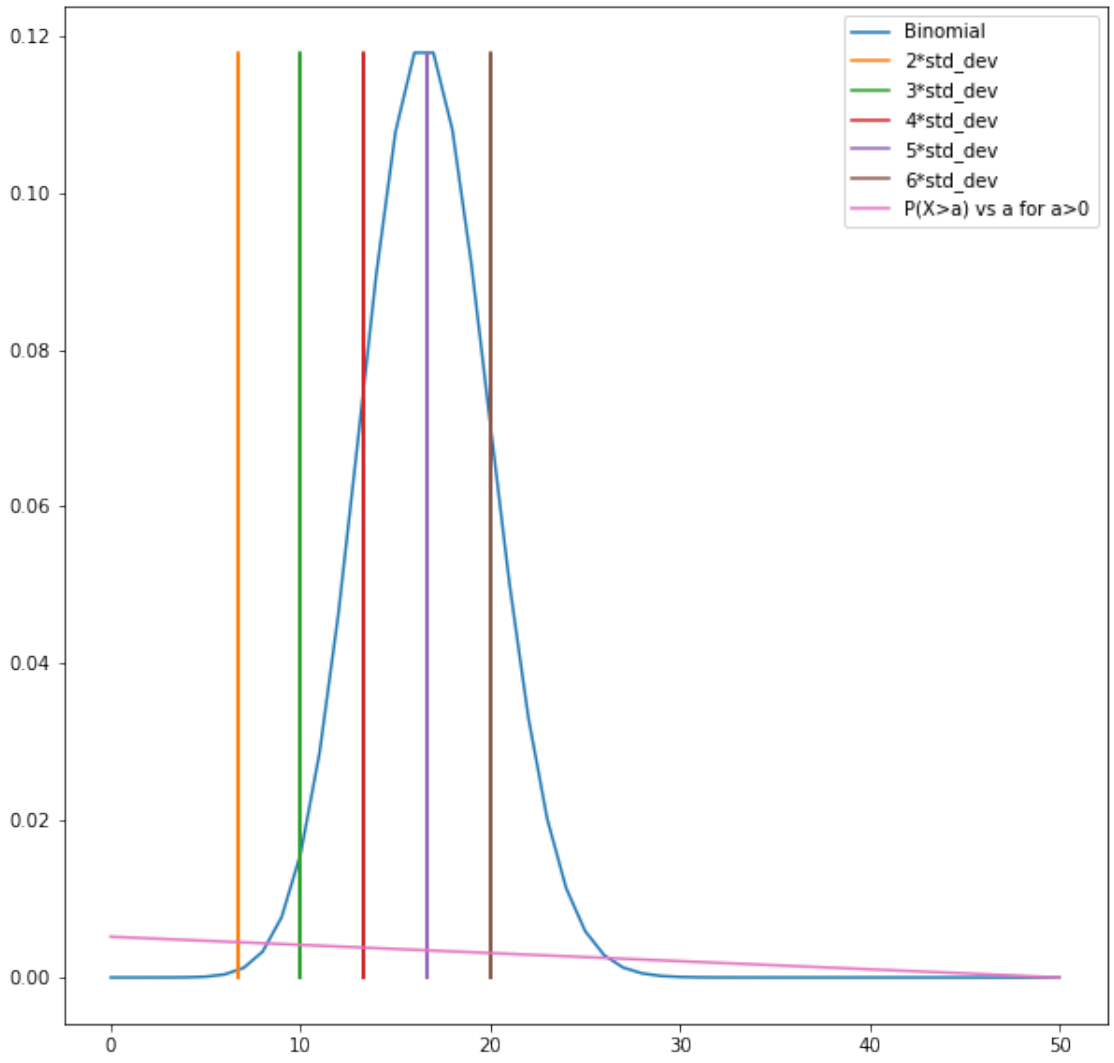
```
std_dev_arr = [std_dev for a in x]
fig, ax = plt.subplots(1)
ax.plot(mean_array, binomial, label="Mean")
ax.plot(variance_arr, binomial, label="Variance")
ax.plot(std_dev_arr, binomial, label="Std_dev");
ax.plot(x, binomial, label="Binomial");
leg = ax.legend()
```



[14]:
```
fig, ax = plt.subplots(1, figsize=(10, 10))
ax.plot(x, binomial, label="Binomial")
for k in range(2, 7):
    distances = [k*std_dev for a in range(0, len(binomial))]
    ax.plot(distances, binomial, label=str(k)+"*std_dev")
a = np.linspace(1, 1000, 1000)

def integrand(x):
    return (math.factorial(50)/((math.factorial(t))*(math.
 ↪factorial(50-t))))*(p**t)*((1-p)**(50-t))
I = [quad(integrand, b, 50)[0] for b in x]

ax.plot(x, I, label="P(X>a) vs a for a>0");
leg = ax.legend()
```

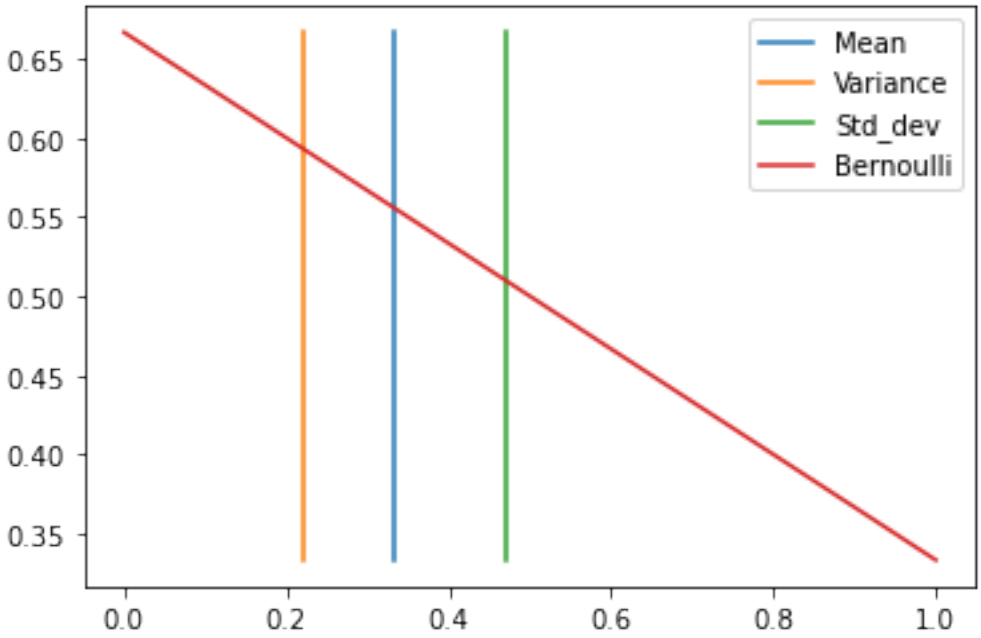


0.0.5 Bernoulli distribution

[15]:
```
x = np.linspace(0,1, 2)
p = 1/3
bernoulli = [(p**t)*((1-p)**(1-t)) for t in x]
```

[16]:
```
mean = p
variance = p*(1-p)
std_dev = variance**0.5
mean_array = [mean for a in x]
variance_arr = [variance for a in x]
std_dev_arr = [std_dev for a in x]
fig, ax = plt.subplots(1)
ax.plot(mean_array, bernoulli, label="Mean")
```

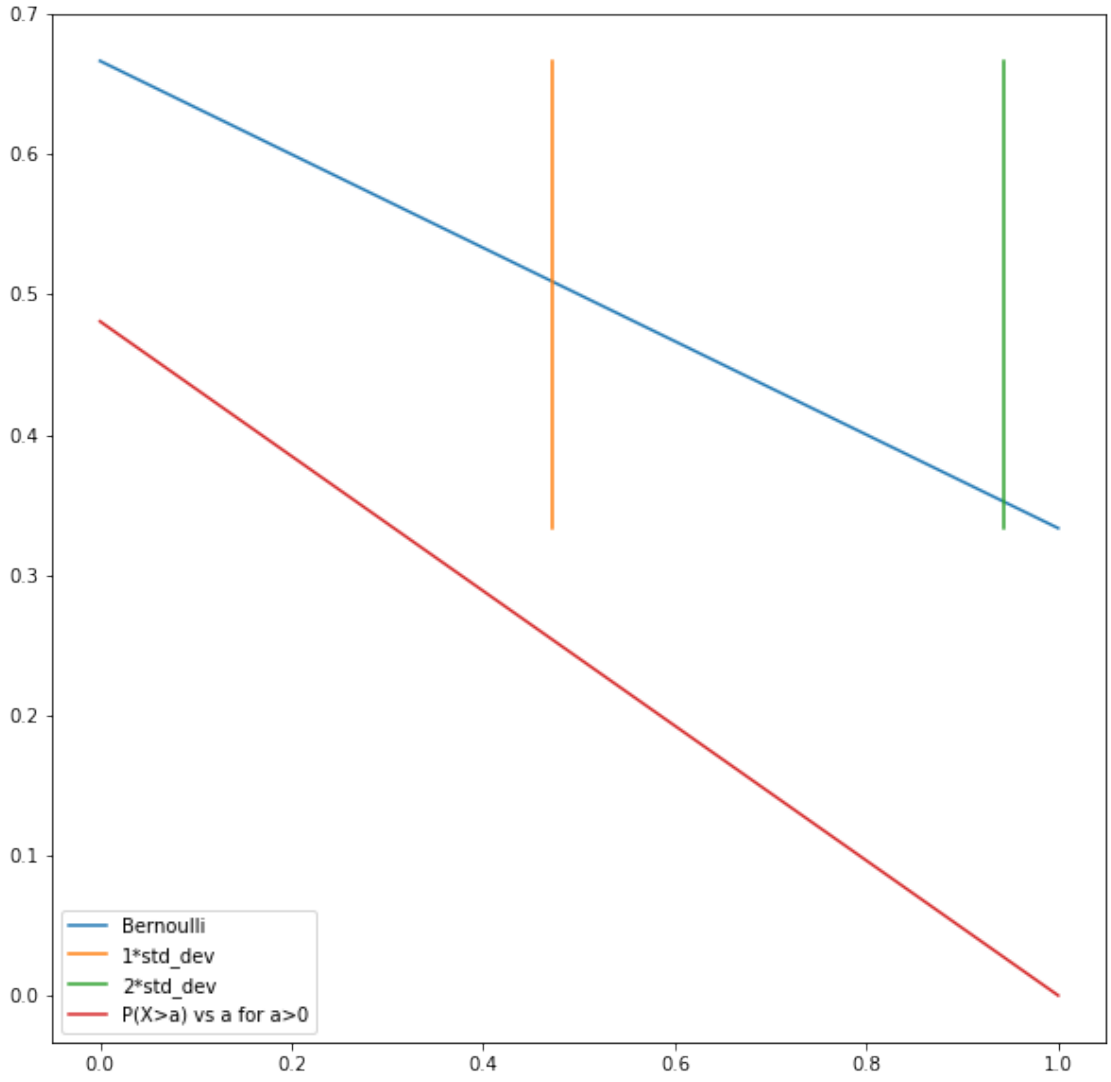
```
ax.plot(variance_arr, bernoulli, label="Variance")
ax.plot(std_dev_arr, bernoulli, label="Std_dev");
ax.plot(x, bernoulli, label="Bernoulli");
leg = ax.legend()
```



[17]:
```
fig, ax = plt.subplots(1, figsize=(10, 10))
ax.plot(x, bernoulli, label="Bernoulli")
for k in range(1, 3):
    distances = [k*std_dev for a in range(0, len(bernoulli))]
    ax.plot(distances, bernoulli, label=str(k)+"*std_dev")
a = np.linspace(1, 1000, 1000)

def integrand(x):
    return (p**x)*((1-p)**(1-x))
I = [quad(integrand, b, 1)[0] for b in x]

ax.plot(x, I, label="P(X>a) vs a for a>0");
leg = ax.legend()
```


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Bernoulli
1*std_dev
2*std_dev
P(X>a) vs a for a>0