BMS COLLEGE OF ENGINEERING

(An Autonomous Institute, Affiliated to VTU, Belagavi)

DEPARTMENT OF MACHINE LEARNING

PROBABILITY AND STATISTICS FOR MACHINE LEARNING

(Course Code: 23AM3PCPSM)

SOLUTION MANUAL

Lab In-charge

1. Dr. Gowrishankar
2. Dr. Monika Puttaramaiah
3. Prof. Amogh Pramod Kulkarni

Academic Year: 2023-24 (Session: Dec 2023 – Mar 2024)

BMS COLLEGE OF ENGINEERING

(Autonomous Institute, Affiliated to VTU)

VISION

Promoting Prosperity of mankind by augmenting Human Resource Capital through Quality Technical Education & Training.

MISSION

Accomplish Excellence in the field of Technical Education through Education, Research and Service needs of society.

DEPARTMENT OF MACHINE LEARNING

VISION

To achieve excellent standards of quality education in the field of Artificial Intelligence and Machine Learning.

MISSION

- ✓ To nurture the students with strong fundamentals for a successful carrier in the field of artificial intelligence and machine learning.
- ✓ To motivate the students for post-graduation and research.
- ✓ To create impact in the society with continuous research and innovations.

PREFACE

This laboratory manual is prepared by the Department of Machine Learning for Probability and Statistics for Machine Learning (23AM3PCPSM). This lab manual can be used as instructional book for students, staff and instructors to assist in generating various probability distributions and finding its distributions functions with the help of statistical module – scipy.stats in Python. In this manual, the programs are as per syllabus prescribed.

INSTRUCTIONS TO THE STUDENTS

Do's

- ✓ Learn the topics taught in the instruction class and come well prepared to the laboratory session.
- ✓ Update observation & record regularly and get it evaluated by the respective faculty.
- ✓ Practice additional concepts taught in the instruction class in every lab.
- ✓ Be obedient and disciplined during the stay in campus
- ✓ Maintain cleanliness inside the laboratory
- ✓ Damages observed in the laboratory to be informed to the concerned staff immediately.

Don'ts

- ✗ Usage of cell phones or any other electronic gadgets inside the laboratory.
- ✗ Eat or drink in the laboratory.
- ✗ Damage the department belongings.
- ✗ Meddle with the software programs that are harmful to the laboratory systems.

SYLLABUS

<table>
<tr><td>Course Title</td><td colspan="5">PROBABILITY AND STATISTICS FOR MACHINE LEARNING</td></tr>
<tr><td>Course Code</td><td>23AM3PCPSM</td><td>Credits</td><td>3</td><td>L-T-P</td><td>2-0-1</td></tr>
<tr><td>CIE</td><td>50 Marks</td><td>SEE</td><td colspan="3">100 Marks (50% Weightage)</td></tr>
<tr><td>Contact Hours/Week</td><td>2</td><td colspan="3">Total Lab Hours</td><td>12</td></tr>
<tr><td colspan="6"></td></tr>
<tr><td colspan="5">UNIT - 1</td><td>6 Hrs</td></tr>
<tr><td colspan="6">Introduction: What is Probability, Uncertainty in Machine Learning, Why Probability for Machine Learning, Joint and Marginal Probability, Conditional Probability, Intuition for Joint, Marginal and Conditional, Examples of Calculating Probability.</td></tr>
<tr><td colspan="5">UNIT - 2</td><td>5 Hrs</td></tr>
<tr><td colspan="6">Bayesian Probability: Introduction to Bayes Theorem and Modelling Hypothesis, Density Estimation, Maximum a Posteriori, Bayes Optimal Classifier, Develop a Naïve Bayes Classifier, Prior and Conditional Probabilities of Naïve Bayes.</td></tr>
<tr><td colspan="5">UNIT - 3</td><td>4 Hrs</td></tr>
<tr><td colspan="6">Discrete Random Variables: Distribution of a Random Variable, Types of Random Variables, Joint and Marginal distribution, Independence of random variables, Expectation and Variance, function, properties, standard deviation, Covariance and Correlation, Properties of Discrete Random Variables, Bernoulli Distribution, Binomial Distribution, Geometric Distribution and Poisson Distribution.</td></tr>
<tr><td colspan="5">UNIT - 4</td><td>6 Hrs</td></tr>
<tr><td colspan="6">Continuous Random Variables: Probability Density, Uniform Distribution, Exponential Distribution and Pareto Distribution, Normal Distribution, Central Limit Theorem.</td></tr>
<tr><td colspan="5">UNIT - 5</td><td>5 Hrs</td></tr>
<tr><td colspan="6">Introduction to Statistics: Population and Sample, Parameters and Statistics, Descriptive Statistics, Mean, Median, Quantiles, Percentiles, Quartiles, Interquartile range, Variance and Standard deviation, Standard Errors of Estimates</td></tr>
<tr><td colspan="6">Text Books:
1. Probability and Statistics for Computer Scientists, Michael Baron, CRC press, 2019.
2. Probability for Machine Learning Discover how to harness uncertainty with Python, Jason Brownlee.</td></tr>
<tr><td colspan="6">Reference Books:
1. Probability, Statistics, Queuing theory and Computer Science Applications, Kishore S Trivedi, 2nd Edition, Willey Publishers, 2008.</td></tr>
</table>

Course Outcomes:

CO1	Analyze the real-time challenges based on distribution of data, predict future estimations using the concept of probability and acquire skills to better handle the present situation.
CO2	Apply statistical knowledge to understand the uncertainty in daily applications and formulate automated solutions.
CO3	Analyze the relationship between the features extracted from samples and apply the learnt algorithms to handle data efficiently.

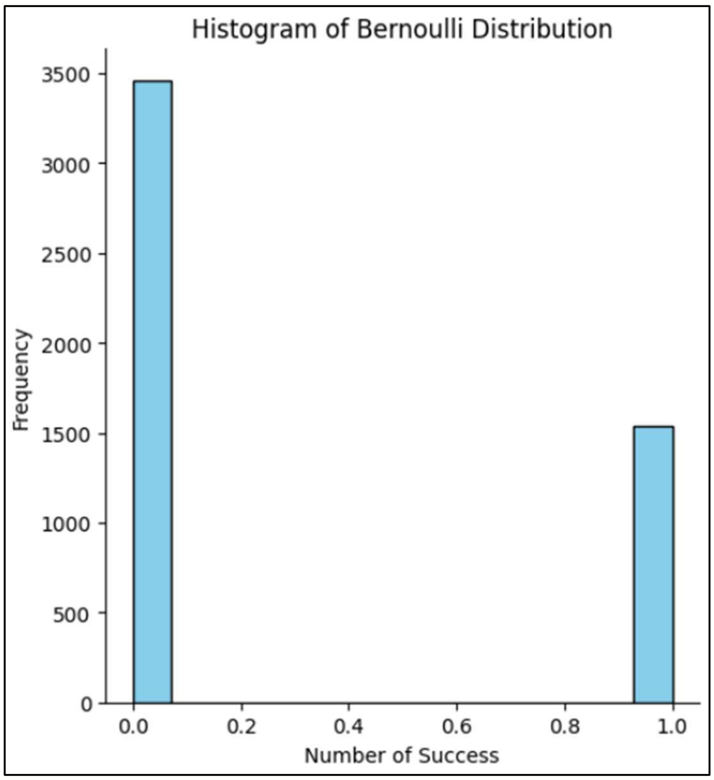
INDEX

Part-A

1. Generating Bernoulli Distribution in Python

```
from scipy.stats import bernoulli
bernoulli_data = bernoulli.rvs(size = 5000, p =0.3) #size -  number of random numbers to generate

viz_bernoulli = sns.displot(bernoulli_data,
                             kde =False,
                             color="skyblue", alpha = 1)
viz_bernoulli.set(xlabel='Number of Success', ylabel='Frequency', title = "Histogram of Bernoulli Distribution")
```

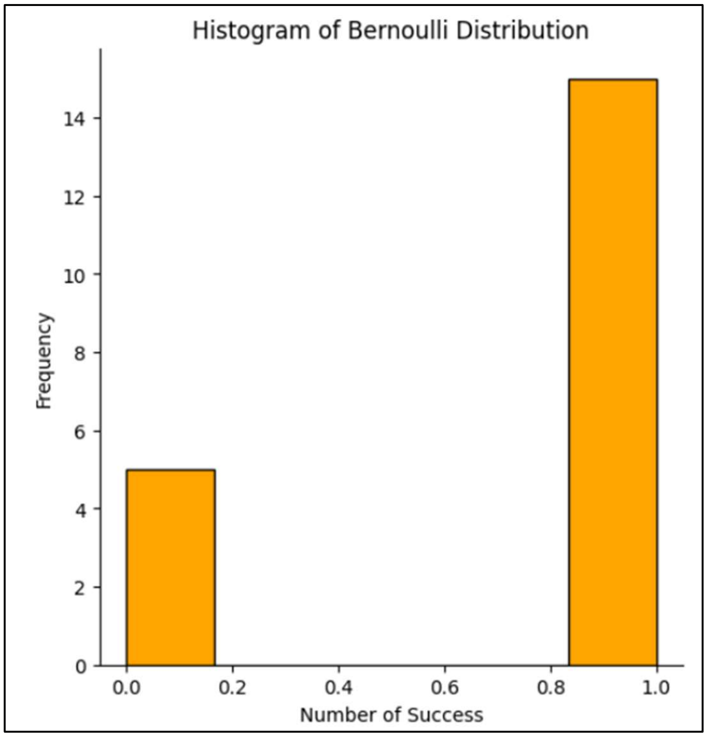
OUTPUT:



If the probability of finding a defective item in a certain type of good is 0.7. If X = 1 indicates probability of a defective item and X = 0 indicates a non- defective item. Find the random variate for 20 times. Also find the mean and variance.

```
defectives = bernoulli.rvs(size = 20, p = 0.7)
viz_defectives = sns.displot(defectives,
                             kde = False,
                             color="orange", alpha = 1)
viz_defectives.set(xlabel='Number  of  Success',  ylabel='Frequency',
title ="Histogram of Bernoulli Distribution")

mean =defectives.mean()
variance = defectives.var()
print(f"The average number of defectives is: , {mean:.2f}")
print(f"The variance observed in the defectives is: , {variance:.2f}")
```

OUTPUT:

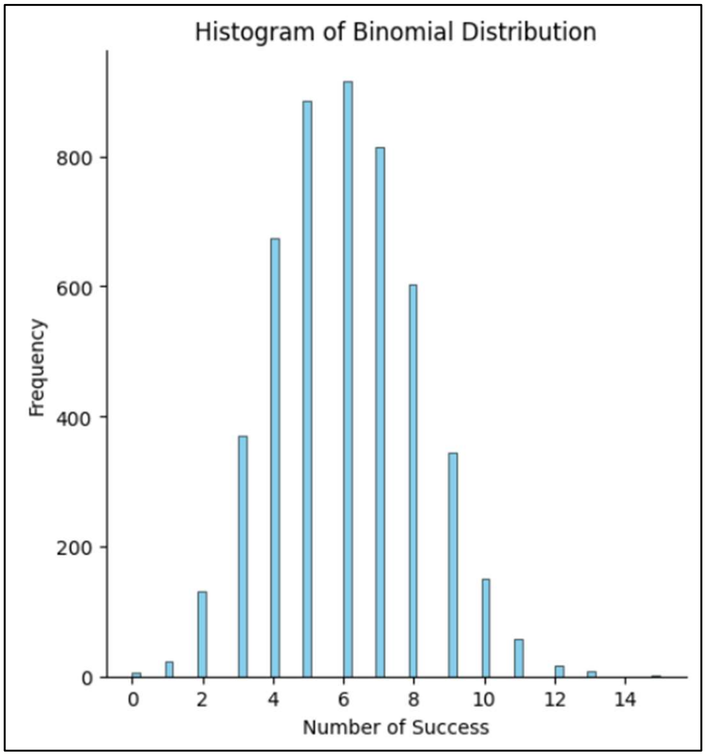



```
The average number of defectives is:  0.75
The variance observed in the defectives is:  0.19
```

2. Generating Binomial Distribution in Python

```
from scipy.stats import binom
binomial_data = binom.rvs(n =  20, p=0.3, size = 5000)
viz_binomial = sns.displot(binomial_data,
                             kde =False,
                             color="skyblue", alpha = 1)
viz_binomial.set(xlabel='Number of Success', ylabel='Frequency', title
= "Histogram of Binomial Distribution")
```

OUTPUT:



For a binomial distribution with n = 20 trials and p =0.6,

i. **What is the probability of getting greater 15 successes?**

ii. **What is the probability of getting exactly 8 successes?**

```
n = 20
p = 0.6
prob_greater_15 = 1 - binom.cdf(15, n,p)
print(f"The probability of getting greater than 15 successes is: {prob_greater_15:.4f}") prob_exactly_8 = binom.pmf(8,n,p)
print(f"The probability of getting exactly 8 successes is: {prob_exactly_8:.4f}")
```

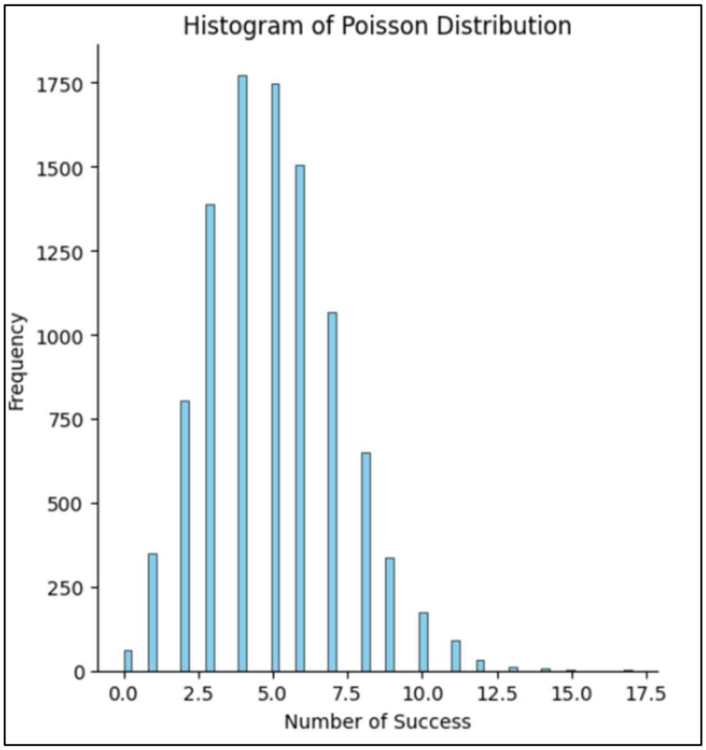
OUTPUT:

```
The probability of getting greater than 15 successes is: 0.0510
The probability of getting exactly 8 successes is: 0.0355
```

3. Generating Poisson Distribution in Python

```
from scipy.stats import poisson
poisson_data = poisson.rvs(mu = 5, size = 10000 )
viz_poisson = sns.displot(poisson_data,
                             kde =False,
                             color="skyblue", alpha = 1)
viz_poisson.set(xlabel='Number of Success', ylabel='Frequency', title = "Histogram of Poisson Distribution")
```

OUTPUT:



According to a recent poll by the Pew Internet Project, users between the ages of 14 and 17 send an average of 50 text messages each day. Let X= the number of texts that a user aged 14 to 17 sends per day. The discrete random variable X takes on the values x=0,1,2…. The random variable X has a Poisson distribution: X�P(50). Find the probability that the user sends less than 500 messages each day.

```
lambda_val = 50
prob_less_than_500 = poisson.cdf(499, lambda_val)
print(f"The probability that the user sends less than 500 messages
each day is: {prob_less_than_500:.4f}")
```

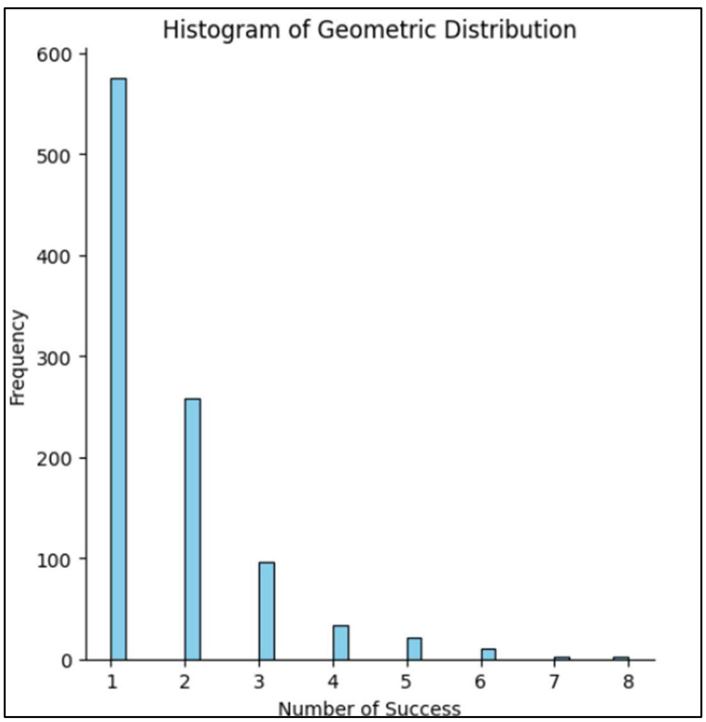
OUTPUT:

```
The probability that the user sends less than 500 messages each day is:
1.0000
```

4. Generating Geometric Distribution in Python

```
from scipy.stats import geom
geometric_data = geom.rvs(p=0.6, size = 1000)
viz_geometric = sns.displot(geometric_data,
                            kde =False,
                            color ="skyblue", alpha=1)
viz_geometric.set(xlabel = "Number of Success", ylabel = "Frequency")
```

OUTPUT:



The lifetime risk of developing cancer is 1/78. If X is the number of people you ask until one says he/she has cancer. If X~G(1/78). What is the probability that we need to ask 10 people before one says he/she has cancer?

```
p = 1/78
k = 10
prob_equal_10 = geom.pmf(k, p)

print(f"The probability of needing to ask 10 people before one says
he/she has cancer is: {prob_equal_10:.4f}")
```

OUTPUT:

```
The probability of needing to ask 10 people before one says he/she has
cancer is: 0.0114
```

5. How many people are required so that any two people in the group have the same birthday with at least a 50-50 chance?

```
# define maximum group size
n = 30
# number of days in the year
days = 365
# calculate probability for different group sizes
p = 1.0
for i in range(1, n):
  av = days - i
  p *= av / days
   print('n=%d,  %d/%d,  p=%.3f  1-p=%.3f'  %  (i+1,  av,  days,  p*100,  (1-
p)*100))
```

OUTPUT:

```
n=2, 364/365, p=99.726 1-p=0.274
n=3, 363/365, p=99.180 1-p=0.820
n=4, 362/365, p=98.364 1-p=1.636
n=5, 361/365, p=97.286 1-p=2.714
n=6, 360/365, p=95.954 1-p=4.046
n=7, 359/365, p=94.376 1-p=5.624
n=8, 358/365, p=92.566 1-p=7.434
n=9, 357/365, p=90.538 1-p=9.462
n=10, 356/365, p=88.305 1-p=11.695
n=11, 355/365, p=85.886 1-p=14.114
n=12, 354/365, p=83.298 1-p=16.702
n=13, 353/365, p=80.559 1-p=19.441

n=14, 352/365, p=77.690 1-p=22.310
```

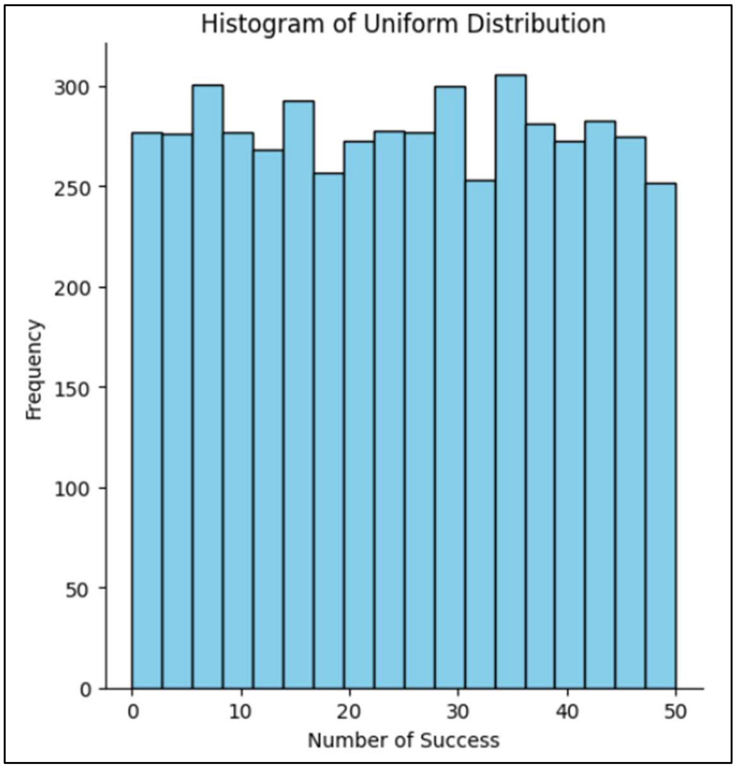
```
n=15, 351/365, p=74.710 1-p=25.290
n=16, 350/365, p=71.640 1-p=28.360
n=17, 349/365, p=68.499 1-p=31.501
n=18, 348/365, p=65.309 1-p=34.691
n=19, 347/365, p=62.088 1-p=37.912
n=20, 346/365, p=58.856 1-p=41.144
n=21, 345/365, p=55.631 1-p=44.369
n=22, 344/365, p=52.430 1-p=47.570
n=23, 343/365, p=49.270 1-p=50.730
n=24, 342/365, p=46.166 1-p=53.834
n=25, 341/365, p=43.130 1-p=56.870
n=26, 340/365, p=40.176 1-p=59.824
n=27, 339/365, p=37.314 1-p=62.686
n=28, 338/365, p=34.554 1-p=65.446
n=29, 337/365, p=31.903 1-p=68.097
n=30, 336/365, p=29.368 1-p=70.632
```

PART B

6. Generating Uniform Distribution in Python

```
from scipy.stats import uniform
uniform_data = uniform.rvs(size = 5000, loc = 0, scale = 50)
viz_uniform = sns.displot(uniform_data,
                          kde =False,
                          color ="skyblue",
                          alpha = 1)
viz_uniform.set(xlabel  = "Number of Success", ylabel="Frequency")
```

OUTPUT:



The waiting time (in minutes) for a bus at a certain bus stop follows a uniform distribution between 5 and 15 minutes. Let X be the random variable representing the waiting time.

i. **Find the probability that a randomly selected person waits less than 10 minutes for the bus.**
ii. **Calculate the expected value (mean) and variance of the waiting time.**

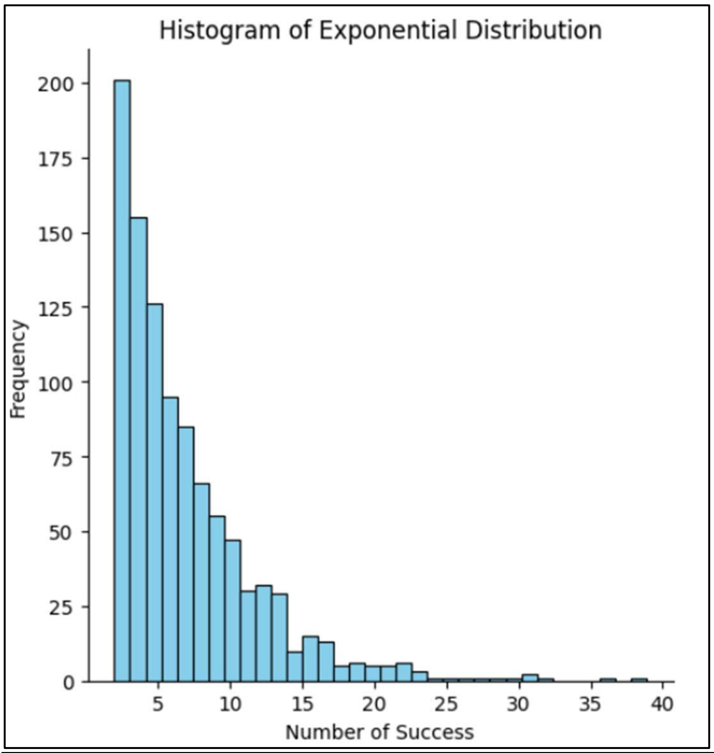
```
a = 5
b = 15
prob_less_than_10 = uniform.cdf(10, loc=a, scale = b-a)
print(f" i. Probability of waiting less than 10 minutes:
{prob_less_than_10:.2f}")
mean_waiting_time = uniform.mean(loc=a, scale=b-a)
variance_waiting_time = uniform.var(loc=a, scale=b-a)
print(f" ii. Mean waiting time: {mean_waiting_time:.2f}")
print(f"   Variance of waiting time: {variance_waiting_time:.2f}")
```

OUTPUT:

```
i.    Probability of waiting less than 10 minutes: 0.50
ii.   Mean waiting time: 10.00
      Variance of waiting time: 8.33
```

7. Generating Exponential Distribution in Python

```
from scipy.stats import expon
expo_data = expon.rvs(loc = 2, scale = 5, size = 1000)
viz_expon = sns.displot(expo_data,
                        kde =False,
                        alpha =1,
                        color="skyblue")
viz_expon.set(xlabel ="Number of Success", ylabel = "Frequency")
```



If the average waiting time for a bus is 15 minutes, what is the probability that the waiting time is less than 10 minutes?

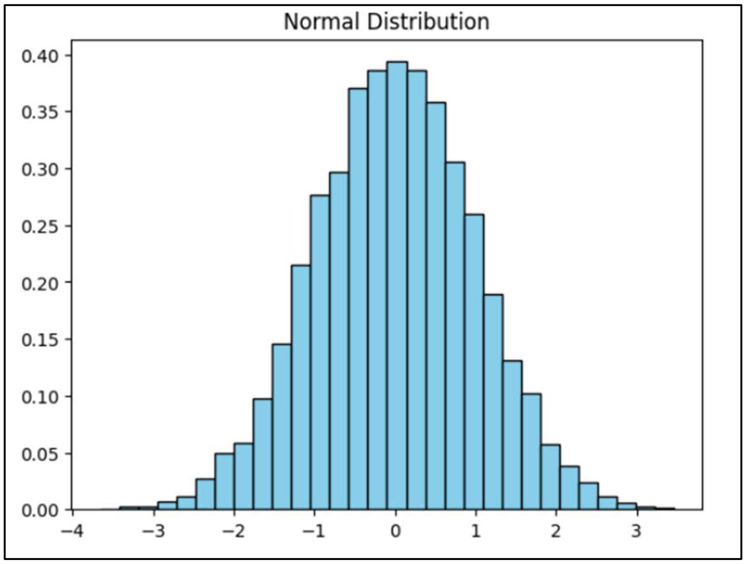
```
mean_waiting_time = 15
prob_less_than_10 = expon.cdf(10, scale=15)
print(f"The probability of waiting less than 10 minutes is: {prob_less_than_10:.4f}")
```

OUTPUT:

```
The probability of waiting less than 10 minutes is: 0.4866
```

8. Generating Normal Distribution in Python

```
from scipy.stats import norm
data_normal = norm.rvs(size=10000,loc=0,scale=1)
plt.hist(data_normal,density=True, edgecolor="black", bins =30,
color="skyblue")
```



In a normal distribution with mean =10 and standard deviation = 2. what is the probability that a randomly selected value is greater than 12?

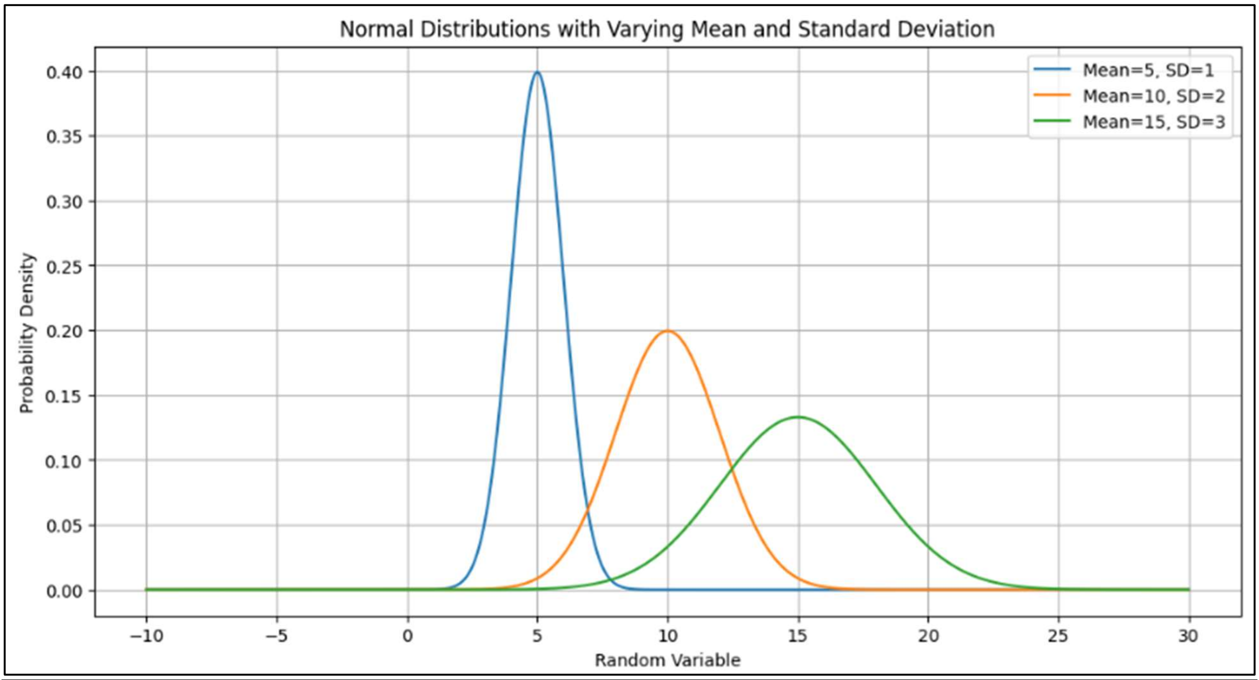
```
mean_value = 10
std_deviation = 2

prob_greater_12 = 1 - norm.cdf(12, loc=mean_value, scale=std_deviation)

print(f"The probability of a randomly selected value being greater than
12 is: {prob_greater_12:.4f}")
```

OUTPUT:

```
The probability of a randomly selected value being greater than 12 is:
0.1587
```

9. How does the parameters mean and standard deviation affect the normal distribution?

```
mean_values = [5, 10, 15]
std_deviation_values = [1, 2, 3]
x_values = np.linspace(-10, 30, 1000)


plt.figure(figsize=(12, 6))

for mean, std_dev in zip(mean_values, std_deviation_values):
    pdf_values = norm.pdf(x_values, loc=mean, scale=std_dev)
    plt.plot(x_values, pdf_values, label=f'Mean={mean}, SD={std_dev}')

plt.title('Normal  Distributions  with  Varying  Mean  and  Standard
Deviation')
plt.xlabel('Random Variable')
plt.ylabel('Probability Density')
plt.legend()
plt.grid(True)
plt.show()
```



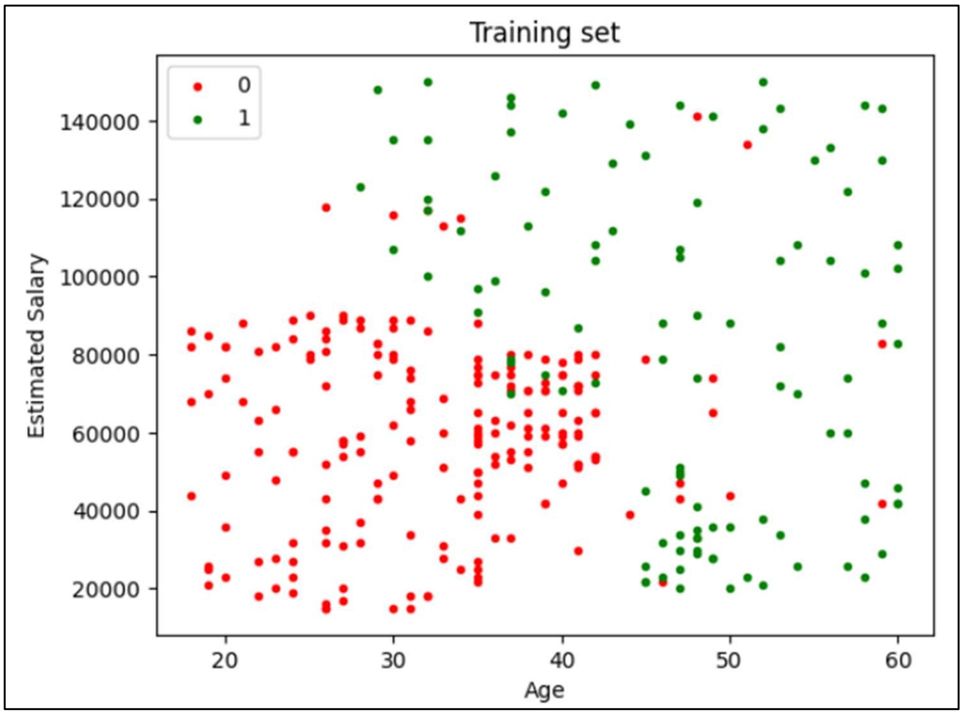
10. Implementing a Naïve Bayes Classifier using the "Social Network Adds" Dataset

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import time
from sklearn.metrics import confusion_matrix

#reading dataset
Data=pd.read_csv('Social_Network_Ads.csv')
print(Data.head(10))
Data.describe()

#training and testing set size
train_size=int(0.75*Data.shape[0])
test_size=int(0.25*Data.shape[0])
print("Training set size : "+ str(train_size))
print("Testing set size : "+str(test_size))

Data=Data.sample(frac=1)
X=Data.iloc[:,[2, 3]].values
y=Data.iloc[:,4].values
X=X.astype(float)

#training set split
X_train=X[0:train_size,:]
y_train=y[0:train_size]
#testing set split
X_test=X[train_size:,:]
y_test=y[train_size:]

#visualize the training set
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
    c = ListedColormap(('red', 'green'))(i), label = j,marker='.')
plt.title('Training set')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
cm2=confusion_matrix(y_test,y_pred)
print("Confusion Matrix:", cm2)
```

OUTPUT:

```
Training set size : 300
Testing set size : 100
```




```
Confusion Matrix:
[[56  6]
 [ 6 32]]
```