Applied Computational Statistics

Lab Manual

Department of Computer Science and Engineering
The NorthCap University, Gurugram

Applied Computational Statistics

Lab WorkBook

CSL227

Ms.Sonal Saurabh

Dr. Srishti

Name: | Roll No:

Department of Computer Science and Engineering

The NorthCap University, Gurugram- 122001, India

Session 2022-23

THE NORTHCAP UNIVERSITY

Published by:

Department of Computer Science & Engineering
School of Engineering and Technology
The NorthCap University Gurugram

• Laboratory Manual is for Internal Circulation only

Copying or facilitating copying of lab work comes under cheating and is considered as use of unfair means. Students indulging in copying or facilitating copying shall be awarded zero marks for that particular experiment. Frequent cases of copying may lead to disciplinary action. Attendance in lab classes is mandatory.

Labs are open up to 7 PM upon request. Students are encouraged to make full use of labs beyond normal lab hours.

PREFACE

Applied Computational Statistics Laboratory Manual is designed to meet the course and program requirements of NCU curriculum for B.Tech. fourth semester students of CSE branch. The concept of the lab work is to give brief practical experience for basic lab skills to students. It provides the space and scope for self-study so that students can come up with new and creative ideas.

The Lab manual is written on the basis of "teach yourself pattern" and expected that students who come with proper preparation should be able to perform the experiments without any difficulty. A brief introduction to each experiment with information about self-study material is provided. The laboratory exercises will help students to provide a hands-on each exercise that will help them to understand thoroughly. The students are expected to come thoroughly prepared for the lab. General disciplines, safety guidelines and report writing are also discussed.

The lab manual is a part of curriculum for the The NorthCap University, Gurugram. Teacher's copy of the experimental results and answer for the questions are available as sample guidelines.

We hope that lab manual would be useful to students of CSE branch and author requests the readers to kindly forward their suggestions / constructive criticism for further improvement of the work book.

Author expresses deep gratitude to Members, Governing Body-NCU for encouragement and motivation.

Authors
The NorthCap University
Gurugram, India

CONTENTS

1. INTRODUCTION

That 'learning is a continuous process' cannot be over emphasized. The theoretical knowledge gained during lecture sessions need to be strengthened through practical experimentation. Thus, practical makes an integral part of a learning process.

COURSE OBJECTIVES:

1. **Apply appropriate descriptive statistical and exploratory methods in the analysis of datasets.**
2. **Visualize the data for the various analysis.**
3. **Understand the probability mass function and various discrete distributions through application on real world examples.**
4. **Understand the probability density function and various continuous distributions through application on real world examples.**
5. **Understand and interpret statistical hypothesis test.**
6. **Translate real-world problems into probability models using Bayesian Statistics.**

2. LAB REQUIREMENTS

S.No.	Requirements	Details
1	**Software Requirements**	Python 3.x, Numpy, Pandas, Matplotlib, Seaborn, statistics, sci-kit learn
2	**Operating System**	Windows 7 onwards or Linux (32 or 64 bit)
3	**Hardware Requirements**	4 GB RAM (Recommended) 2.60 GHz (Recommended)
4	**Required Bandwidth**	NA

3. GENERAL INSTRUCTIONS

a. General discipline in the lab

- Students must turn up in time and contact concerned faculty for the experiment they are supposed to perform.
- Students will not be allowed to enter late in the lab.
- Students will not leave the class till the period is over.
- Students should come prepared for their experiment.
- Experimental results should be entered in the lab report format and certified/signed by concerned faculty/ lab Instructor.
- Students must get the connection of the hardware setup verified before switching on the power supply.
- Students should maintain silence while performing the experiments. If any necessity arises for discussion amongst them, they should discuss with a very low pitch without disturbing the adjacent groups.
- Violating the above code of conduct may attract disciplinary action.
- Damaging lab equipment or removing any component from the lab may invite penalties and strict disciplinary action.

b. Attendance

- Attendance in the lab class is compulsory.
- Students should not attend a different lab group/section other than the one assigned at the beginning of the session.
- On account of illness or some family problems, if a student misses his/her lab classes, he/she may be assigned a different group to make up the losses in consultation with the concerned faculty / lab instructor. Or he/she may work in the lab during spare/extra hours to complete the experiment. No attendance will be granted for such case**.**

c. Preparation and Performance

- Students should come to the lab thoroughly prepared on the experiments they are assigned to perform on that day. Brief introduction to each experiment with information about self -study reference is provided on LMS.
- Students must bring the lab report during each practical class with written records of the last experiments performed complete in all respect.

- Each student is required to write a complete report of the experiment he has performed and bring to lab class for evaluation in the next working lab. Sufficient space in work book is provided for independent writing of theory, observation, calculation and conclusion.
- Students should follow the Zero tolerance policy for copying / plagiarism. Zero marks will be awarded if found copied. If caught further, it will lead to disciplinary action.
- Refer **Annexure 1** for Lab Report Format

4. LIST OF EXPERIMENTS

Sr. No.	Title of the Experiment	Software used	Unit Covered	CO Covered	Time Required
1.	Introduction To Statistics using Python	Python	1	CO1	2 Hours
2.	Demonstration of descriptive statistics	Python (Jupyter)	1	CO1	2 hours
3.	Demonstrate Measure of Shape using Python	Python (Jupyter)	1	CO1	2 hours
4.	Data visualization for qualitative and quantitative data	Python (Jupyter)	1	CO1	2 hours
5.	Basics of Probability using Python	Python (Jupyter)	1	CO1	2 hours
6.	Write python program to implement laws of Probability	Python (Jupyter)	2	CO2	2 hours
7.	Implementation of Probability Mass Function for Discrete Random Variable using Python	Python (Jupyter)	3	CO3	2 hours
8.	Write a python program to find expectation and variance in discrete distribution	Python (Jupyter)	3	CO3	2 hours
9.	Write Python Program for Continuous Probability Distributions	Python (Jupyter)	3	CO4	2 hours
10	Write python code to find critical value in normal distribution	Python (Jupyter)	4	CO4	2 hours
11	Implement central limit theorem	Python (Jupyter)	5	CO5	2 hours
12	Write a python program to conduct a hypothesis test for unknown variance.	Python (Jupyter)	5	CO5	2 hours
13	Write a python program to conduct hypothesis testing for two populations	Python (Jupyter)	5	CO5	2 hours
Value Added Experiments					
14	Write a python program to bootstrap your data.	Python (Jupyter)	5	CO5	2 hours
15	Write a python program to find confidence interval for known variance.	Python (Jupyter)	5	CO5	2 hours
16	Write a python program to conduct a hypothesis test for known variance	Python (Jupyter)	5	CO5	2 hours

5. LIST OF FLIP EXPERIMENTS

Exp. No.	Title of the Experiment	Mapped CO
1.	Implement the ANOVA and ANCOVA analysis of real-world data	CO5
2.	Implement the Bayes Theorem in Python	CO2
3.	Competitions on Kaggle	CO1-6

6. LIST OF PROJECTS

Sr No.	Project Title	Mapped CO
1.	2008_Swing_States dataset (Available on Kaggle) contains US swing states election results. There are 221 observations in the dataset with 6 features each (state, county, total_votes, dem_votes, rep_votes, dem_share). What is the highest dem_share percentage for different states? Find the country with highest dem_share in each state? Also check for null values. What is the average total_votes for each state? Also calculate their respective median. Calculate the variability and skewness (whether positive or negative or zero) for total_votes for each state. Also calculate the z - score of total_votes for each states. Calculate Interquartile range of total_votes of state PA.	CO1
2.	Suicide Rates Overview 1985 to 2016 dataset (Available on Kaggle). This compiled dataset pulled from four other datasets linked by time and place, and was built to find signals correlated to increased suicide rates among different cohorts globally, across the socio-economic spectrum. Our aim is to make general analysis of suicide rates and we will explain these rates in detail. Given the data, after removing null values, analyse and perform the EDA. How many males and females are there in dataset? Also find out how many males and females are there based on minimum and maximum gdp per capita? How many males and females are there in the data categorise by age parameter? Count the number of people based on Age values and plot. (Hint: Use groupby on age). How many generations are there in the data? Also based on sex parameter? plot a five number summary for three parameters - population, gdp per capita and year. Plot a line graph for the statistics of data. *(Hint: plot on data.describe() where kind = area). Draw two graphs on same canvas for generation parameter. First graph should be **Pie Chart** and Second graph should be Count plot. year on x-axis and suicides per100k of pop on y-axis (scatter plot). year on x-axis and gdp per capita on y-axis (reg plot). year on x-axis and gdp per capita on y-axis (kde plot).	CO2

	year on x-axis and gdp per capita on y-axis (hex plot). Draw FacetGrid between gdp per capita and population. Visualise the graph between year and suicides using strip plot. Plot graph between each unique values of generation parameter. Also use violin plot between generation and population. Visualize the correlation between data.	
3.	Apply descriptive statistics on Insurance dataset.	CO1, CO2
4.	Do exploratory data analysis on University graduate admission dataset.	CO2
5.	Apply ANOVA (analysis of variance) on a given dataset.	CO3
6.	Apply ANCOVA (analysis of covariance) on a given dataset.	CO3
7.	Apply PCA (principal component analysis) on cancer dataset for dimensionality reduction.	CO4
8.	Do exploratory data analysis on IPL cricket dataset and calculate the chance of winning for each team	CO2
9.	Generate a pair of hypotheses for a given case study and statistically evaluate it	CO4, CO5
10.	Evaluate null hypothesis using p-value method	CO4, CO5

7. RUBRICS (Only for Lab components)

Marks Distribution	
Continuous Evaluation (25 Marks)	**Project Evaluations (20 Marks)**
Each experiment shall be evaluated for 5 marks and at the end of the semester proportional marks shall be awarded out of total 25.	Project shall be evaluated for 20 marks and at the end of the semester viva will be conducted related to the project.
Viva and Reporting (25 Marks)	
Following is the breakup of 25 marks for each **10 Marks**: Observation & conduct of experiment. Teacher may ask questions about experiment in mid-term viva. **10 Marks:** Observation & conduct of experiment. **5 Marks:** For report writing	

Annexure 1

CSL227

Lab Work Book

Faculty name: Dr. Sujata

Student Name: Piyush Gambhir

Roll No.: 21CSU349

Semester: Semester IV

Group: AL-3

Department of Computer Science and Engineering

The NorthCap University, Gurugram- 122001, India

Session 2022-23

INDEX

Student name: Piyush Grambhir
Roll No.: 21CSU369

INDEX

S.No.	Experiment	Date of Experiment	Date of Submission	Marks	Signature
1	Introduction to statistics using Python	20-01-23	20-01-23		20/1/23
2	Demonstration of descriptive statistical measures	03-02-23	03-02-23		3/2/23
3	Demonstrate measures of shape using Python	04-02-23	04-02-23		4/2/23.
4	Data visualization for qualitative and quantitative data	10-02-23	17-02-23		17/2/23
5	Basics of probability using Python	24-02-23	24-02-23		24/2/23
6	To implement laws of probability	03-03-23	03-03-23		3/3/23
7	Implementation of probability mass function for discrete random variable using Python.	17-03-23	17-03-23		17/3/23.
8	Write a python program to find expectation and variance in discrete distribution	17-03-23	17-03-23		17/3/23
9	Write Python Program for Continuous Probability Distributions	24-03-23	24-03-23		14/4/23
10	Write python code to find critical value in normal distribution	14-04-23	14-04-23		14/4/23
11	Write a python program to Implement central limit theorem	28-04-23	28-04-23		
12	Write a python program to conduct a hypothesis test for unknown variance.	28-04-23	28-04-23		28/4/23
	13. Write a python program to conduct hypothesis testing for two populations	28-04-23	28-04-23		

EXPERIMENT NO. 1

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL227-ACS-Lab_Manual (github.com)
Date: 20.01.2023
Faculty Signature:
Marks/Grade:

Objective(s): Familiarization with statistics libraries in Python
Outcome: Revision of the libraries Numpy, Pandas, Matplotlib, Seaborn and exploration of Statistics and Scipy.Statistics
Problem Statement: Introduction to statistics using Python
Background Study: Python has libraries like Statistics and SciPy. Statistics which contain functions for several descriptive and inferential statistics tasks which can be of help to the students.

Question Bank:

1. What is difference between differential and inferential statistics?
   Differential statistics deals with analyzing and describing data in a sample or population without making any generalizations to a larger population. It is concerned with describing the characteristics of a sample or population and includes measures such as mean, median, mode, variance, standard deviation, etc.

   On the other hand, inferential statistics involves making inferences or generalizations about a population based on a sample of data. It is used to estimate parameters of the population using sample statistics, such as mean and standard deviation, and to test hypotheses about the population parameters. Inferential statistics is concerned with making predictions and drawing conclusions about the population based on the sample data.

2. What is the relevance of statistics in everyday life?
   Statistics is an important field because it helps us understand the general trends and patterns in a given data set. Statistics can be used for analyzing data and drawing conclusions from it. It can also be used for making predictions about future events and behaviors.

3. What are some of the common tasks you can accomplish using the statistics libraries you have studied in Python?
   It provides a wide range of tools and algorithms for machine learning and artificial intelligence tasks, including supervised, unsupervised, reinforcement, and deep learning.

   Applications:
   - Pattern recognition.
   - Time-series prediction.
   - Reinforcement learning.
   - Natural language processing

Student Work Area

Algorithm/Flowchart/Code/Sample Output

Experiment 1

Introduction to statistics using Python.

```
[63]
# Importitng the libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```
Python

```
[64]
# Importing the dataset
dataset = pd.read_csv('diabetes_dataset.csv')
```
Python

```
[71]
# Checking the top 5 rows of the dataset
print("Top 5 rows of the dataset: ")
print(dataset.head().to_markdown())
```
Python

```
Top 5 rows of the dataset:
|    |   Pregnancies |   Glucose |   BloodPressure |   SkinThickness |   Insulin |   BMI |   DiabetesPedigreeFunction |   Age | Outcome      |
|---:|--------------:|----------:|----------------:|----------------:|----------:|------:|---------------------------:|------:|:-------------|
|  0 |             6 |       148 |              72 |              35 |         0 |  33.6 |                      0.627 |    50 | Diabetic     |
|  1 |             1 |        85 |              66 |              29 |         0 |  26.6 |                      0.351 |    31 | Non-Diabetic |
|  2 |             8 |       183 |              64 |               0 |         0 |  23.3 |                      0.672 |    32 | Diabetic     |
|  3 |             1 |        89 |              66 |              23 |        94 |  28.1 |                      0.167 |    21 | Non-Diabetic |
|  4 |             0 |       137 |              40 |              35 |       168 |  43.1 |                      2.288 |    33 | Diabetic     |
```

```
[66]
# Replacing 0 and 1 in the Outcome column with Non-Diabetic and Diabetic respectively
dataset['Outcome'] = dataset['Outcome'].replace(0, 'Non-Diabetic')
dataset['Outcome'] = dataset['Outcome'].replace(1, 'Diabetic')
# Checking the top 5 rows of the dataset
print("Top 5 rows of the dataset: ")
print(dataset.head().to_markdown())
```
Python

```
Top 5 rows of the dataset:
|    |   Pregnancies |   Glucose |   BloodPressure |   SkinThickness |   Insulin |   BMI |   DiabetesPedigreeFunction |   Age | Outcome      |
|---:|--------------:|----------:|----------------:|----------------:|----------:|------:|---------------------------:|------:|:-------------|
|  0 |             6 |       148 |              72 |              35 |         0 |  33.6 |                      0.627 |    50 | Diabetic     |
|  1 |             1 |        85 |              66 |              29 |         0 |  26.6 |                      0.351 |    31 | Non-Diabetic |
|  2 |             8 |       183 |              64 |               0 |         0 |  23.3 |                      0.672 |    32 | Diabetic     |
|  3 |             1 |        89 |              66 |              23 |        94 |  28.1 |                      0.167 |    21 | Non-Diabetic |
|  4 |             0 |       137 |              40 |              35 |       168 |  43.1 |                      2.288 |    33 | Diabetic     |
```

```
[67]
# Checking the information of the dataset
print("Information of the dataset:")
print(dataset.info())
```
Python

```
Information of the dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    object
dtypes: float64(2), int64(6), object(1)
memory usage: 54.1+ KB
None
```

```python
# Checking the statistical description of the dataset
print("Statistical description of the dataset:")
print(dataset.describe().to_markdown())
```

[68] Python

```
Statistical description of the dataset:
|       |   Pregnancies |   Glucose |   BloodPressure |   SkinThickness |   Insulin |       BMI |   DiabetesPedigreeFunction |      Age |
|:------|--------------:|----------:|----------------:|----------------:|----------:|----------:|---------------------------:|---------:|
| count |     768       |  768      |        768      |        768      |  768      | 768       |                  768       | 768      |
| mean  |       3.84505 |  120.895  |         69.1055 |         20.5365 |   79.7995 |  31.9926  |                    0.471876 |  33.2409 |
| std   |       3.36958 |   31.9726 |         19.3558 |         15.9522 |  115.244  |   7.88416 |                    0.331329 |  11.7602 |
| min   |       0       |    0      |          0      |          0      |    0      |   0       |                    0.078    |  21      |
| 25%   |       1       |   99      |         62      |          0      |    0      |  27.3     |                    0.24375  |  24      |
| 50%   |       3       |  117      |         72      |         23      |   30.5    |  32       |                    0.3725   |  29      |
| 75%   |       6       |  140.25   |         80      |         32      |  127.25   |  36.6     |                    0.62625  |  41      |
| max   |      17       |  199      |        122      |         99      |  846      |  67.1     |                    2.42     |  81      |
```

```python
# Checking the number of null values in the dataset
null_values = dataset.isnull().sum()
print(null_values)
```

[69] Python

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```python
# Checking the number of unique values in the dataset
unique_values = dataset.nunique()
print(unique_values)
```

[70] Python

```
Pregnancies                 17
Glucose                    136
BloodPressure               47
SkinThickness               51
Insulin                    186
BMI                        248
DiabetesPedigreeFunction   517
Age                         52
Outcome                      2
dtype: int64
```

```python
# Checking the number of Diabetic and Non-Diabetic patients
print("Number of Diabetic and Non-Diabetic patients:")
print(dataset['Outcome'].value_counts())
```

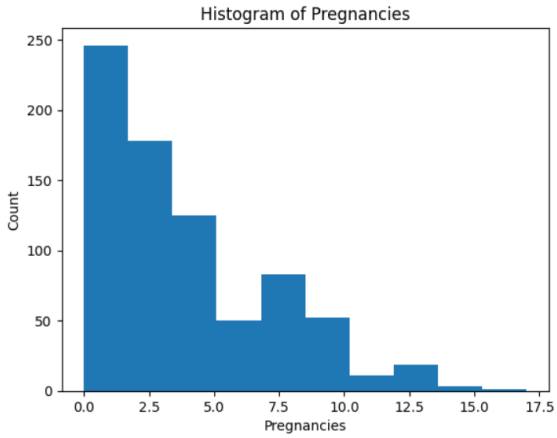
[72] Python

```
Number of Diabetic and Non-Diabetic patients:
Non-Diabetic    500
Diabetic        268
Name: Outcome, dtype: int64
```

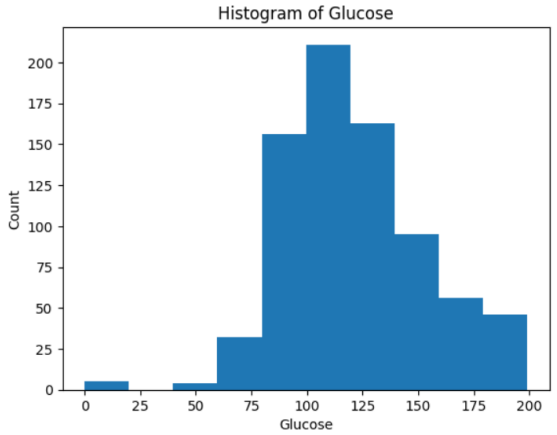
```python
# Plotting Histograms for all the features in the dataset to check the distribution of the data
for i in dataset.columns:
    plt.hist(dataset[i])
    plt.xlabel(i)
    plt.ylabel('Count')
    plt.title("Histogram of " + i)
    plt.show()

```

[78] Python

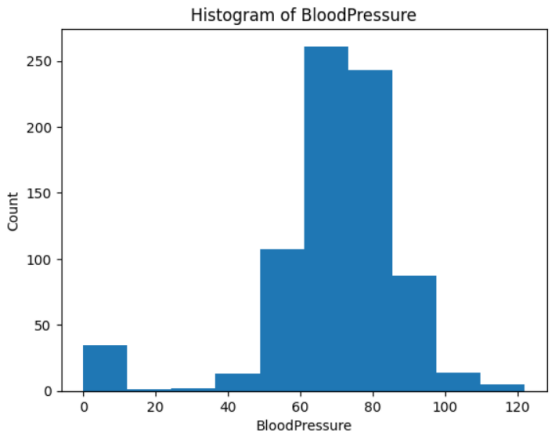
...

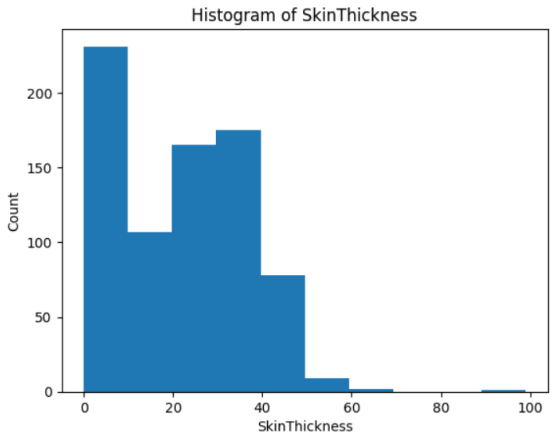


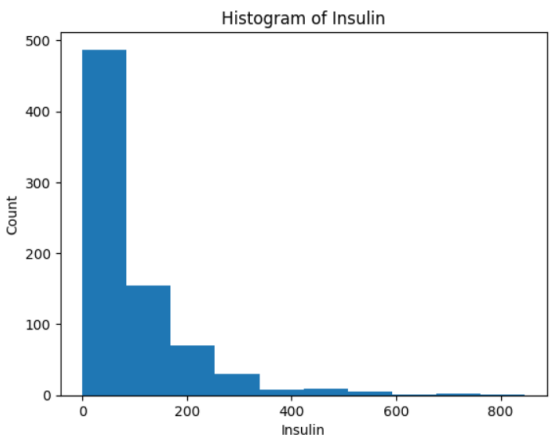
</>

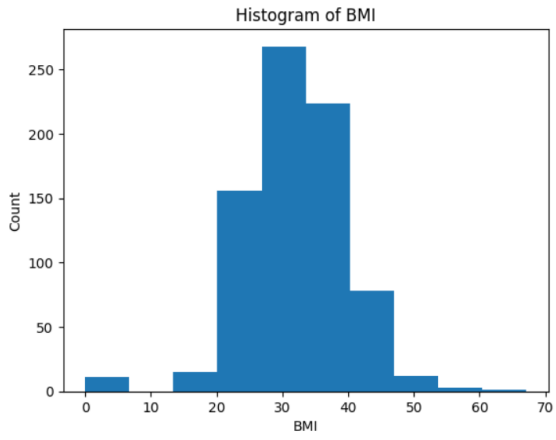


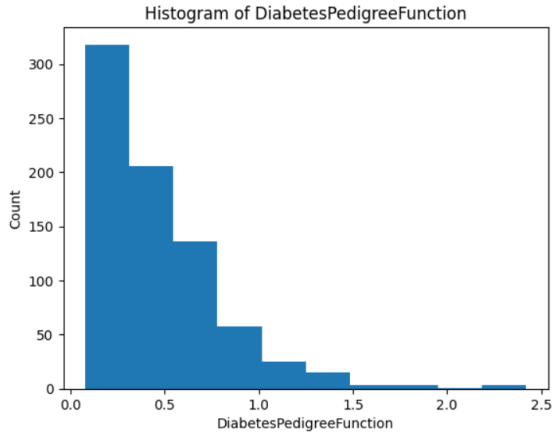
</>

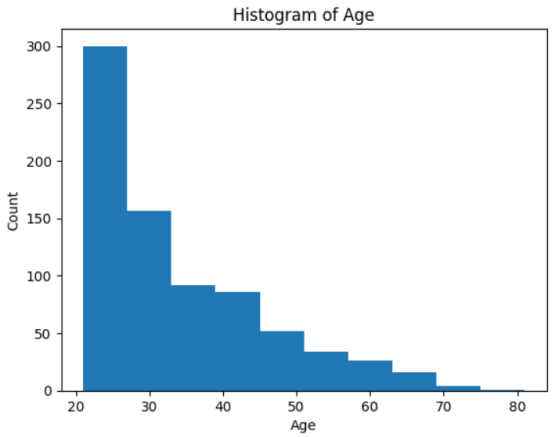


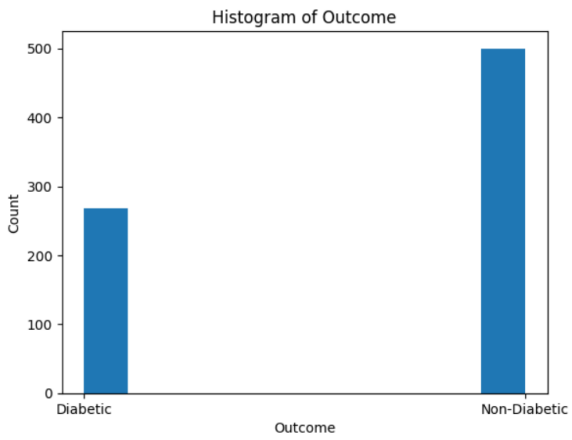

Histogram of SkinThickness
Count
SkinThickness
0
50
100
150
200
0
20
40
60
80
100


Histogram of Insulin
Count
Insulin
0
100
200
300
400
500
0
200
400
600
800


Histogram of BMI
Count
BMI
0
50
100
150
200
250
0
10
20
30
40
50
60
70


Histogram of DiabetesPedigreeFunction
Count
DiabetesPedigreeFunction
0
50
100
150
200
250
300
0.0
0.5
1.0
1.5
2.0
2.5

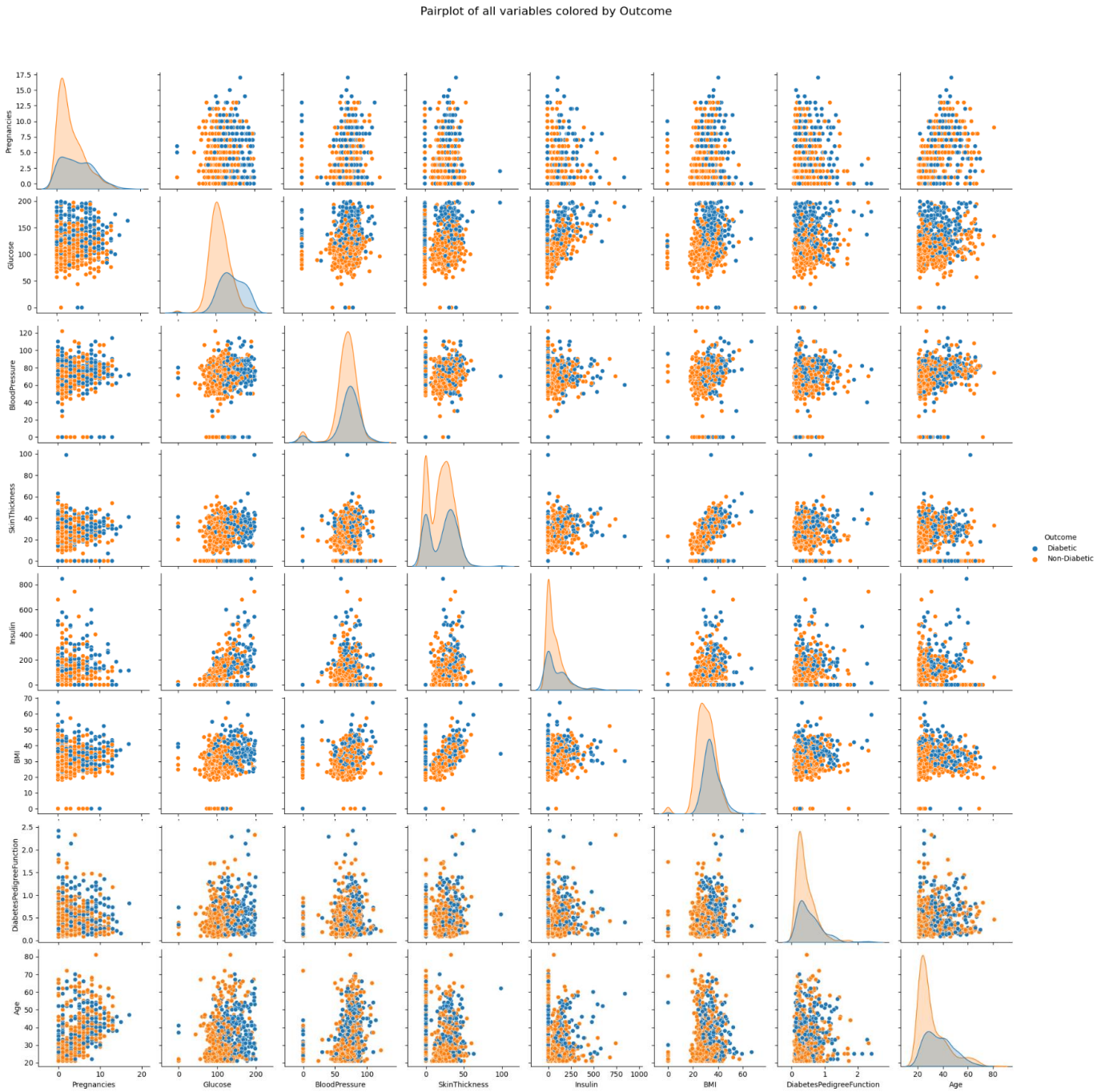

Histogram of Age
Count
Age
0
50
100
150
200
250
300
20
30
40
50
60
70
80


Histogram of Outcome
Count
Outcome
Diabetic
Non-Diabetic
0
100
200
300
400
500

```python
# Pairplot of the dataset
sns.pairplot(dataset, hue='Outcome')
plt.suptitle("Pairplot of all variables colored by Outcome", y=1.05, fontsize=16)
plt.show()
```

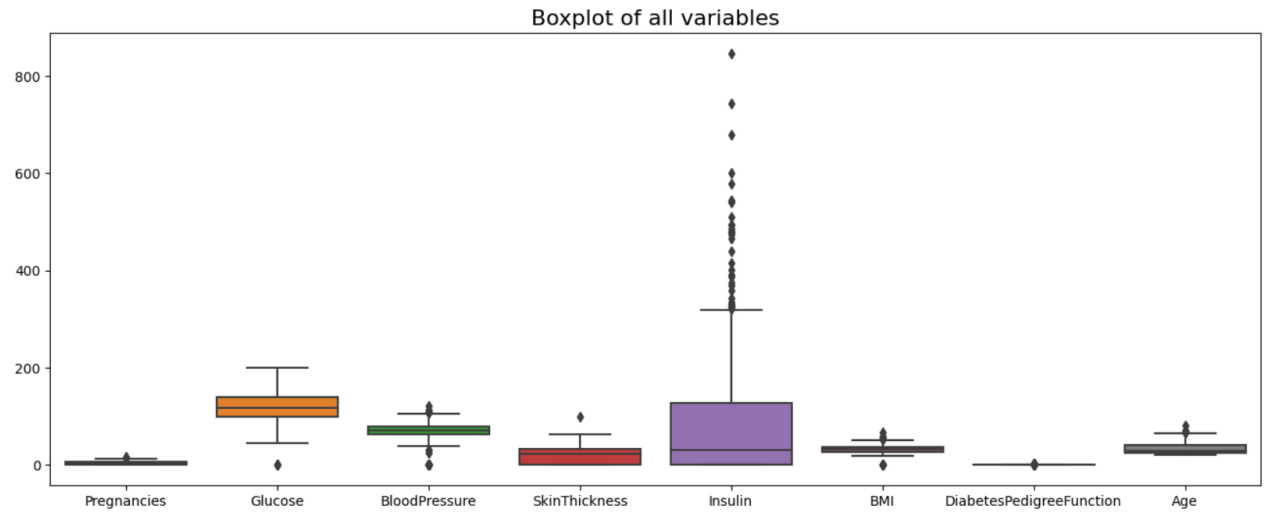
[92] Python

...



```
# Boxplot of the dataset
fig, ax = plt.subplots(figsize=(16,6))
sns.boxplot(data=dataset)
plt.title("Boxplot of all variables", fontsize=16)
plt.show()
```

[19] ✓ 0.3s Python Python



EXPERIMENT NO. 2

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL227-ACS-Lab_Manual (github.com)
Date: 03.02.2023
Faculty Signature:
Marks/Grade:

Objective(s):
Find mean, median, mode, standard deviation for a given dataset

Outcome: Measure of central tendency, measure of variation and measure of position

Problem Statement: Demonstration of descriptive statistical measures

Background Study:
Descriptive statistics are brief descriptive coefficients that summarize a given data set, which can be either a representation of the entire or a sample of a population. Descriptive statistics are broken down into measures of central tendency and measures of variability (spread). Measures of central tendency include the mean, median and mode, while measures of variability include standard deviation, variance, minimum and maximum variables.

Question Bank:

1. What is difference between mean mode medians?
   The mean is the average of a set of data, the mode is the most common value in the set, and the median is the middle value of the set.

2. What is advantage of median over mean?
   The advantages of using the median over the mean are that the median is less sensitive to outliers and extreme values, and it can be used for non-normally distributed data.

3. Is mean affected by presence of outlier in data.
   Yes, the mean is affected by outliers in the data as it can significantly change its value.

4. How to find the deviation of data in given dataset?
   To find the deviation of data in a given dataset, you can follow these steps:
   1) Calculate the mean of the dataset.
   2) For each data point, subtract the mean from the data point.
   3) Square each of the differences obtained in step 2.
   4) Calculate the average of the squared differences obtained in step 3.
   5) Take the square root of the value obtained in step 4.

   This final value is the standard deviation of the dataset, which represents how much the data points deviate from the mean.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 2

Demonstration of descriptive statistical measures.

```
[2]
1  # importing required libraries
2  import math
3  import pandas as pd
4  import numpy as np
5  import matplotlib.pyplot as plt
6  import seaborn as snss
7
8  import scipy
9  from scipy import stats
10 from scipy.stats import zscore, kurtosis, variation, scoreatpercentile
11 from scipy.stats.mstats import gmean
✓ 2.6s                                                    Python
```

```
[3]
1  # Read the data
2  data = pd.read_csv('iris_dataset.csv')
✓ 0.0s                                                    Python
```

```
[4]
1  # Drop the Id column
2  data = data.drop('Id', axis=1)
✓ 0.1s                                                    Python
```

```
[5]
1  # Print the first 5 rows of the data
2  print(data.head().to_markdown())
✓ 0.0s                                                    Python
```

```
...
|    |   SepalLengthCm |   SepalWidthCm |   PetalLengthCm |   PetalWidthCm | Species     |
|---:|----------------:|---------------:|----------------:|---------------:|:------------|
|  0 |             5.1 |            3.5 |             1.4 |            0.2 | Iris-setosa |
|  1 |             4.9 |            3   |             1.4 |            0.2 | Iris-setosa |
|  2 |             4.7 |            3.2 |             1.3 |            0.2 | Iris-setosa |
|  3 |             4.6 |            3.1 |             1.5 |            0.2 | Iris-setosa |
|  4 |             5   |            3.6 |             1.4 |            0.2 | Iris-setosa |
```

```
[6]
1  # Information about the data
2  print(data.info())
✓ 0.1s                                                    Python
```

```
...
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
```

```
# Describe the data
print(data.groupby('Species').describe())
```

[7] ✓ 0.1s Python

```
                SepalLengthCm
                        count   mean       std  min    25%  50%  75%  max  \
Species
Iris-setosa              50.0  5.006  0.352490  4.3  4.800  5.0  5.2  5.8
Iris-versicolor          50.0  5.936  0.516171  4.9  5.600  5.9  6.3  7.0
Iris-virginica           50.0  6.588  0.635880  4.9  6.225  6.5  6.9  7.9

                SepalWidthCm         ... PetalLengthCm      PetalWidthCm
                       count   mean  ...           75%  max        count
Species                              ...
Iris-setosa             50.0  3.418  ...         1.575  1.9         50.0  \
Iris-versicolor         50.0  2.770  ...         4.600  5.1         50.0
Iris-virginica          50.0  2.974  ...         5.875  6.9         50.0


                  mean       std  min  25%  50%  75%  max
Species
Iris-setosa      0.244  0.107210  0.1  0.2  0.2  0.3  0.6
Iris-versicolor  1.326  0.197753  1.0  1.2  1.3  1.5  1.8
Iris-virginica   2.026  0.274650  1.4  1.8  2.0  2.3  2.5

[3 rows x 32 columns]
```

Mean, Median, Mode

```
# Mean of the data
print(data.groupby('Species').mean().to_markdown())
```

[8] ✓ 0.0s Python

Species	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Iris-setosa	5.006	3.418	1.464	0.244
Iris-versicolor	5.936	2.77	4.26	1.326
Iris-virginica	6.588	2.974	5.552	2.026

```
# Median of the data
print(data.groupby('Species').median().to_markdown())
```

[9] ✓ 0.0s Python

Species	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Iris-setosa	5	3.4	1.5	0.2
Iris-versicolor	5.9	2.8	4.35	1.3
Iris-virginica	6.5	3	5.55	2

```
# Mode of the data
data.groupby('Species').value_counts()
```

[10] ✓ 0.0s Python

```
Species         SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
Iris-setosa     4.9            3.1           1.5            0.1            3
                4.3            3.0           1.1            0.1            1
                4.4            2.9           1.4            0.2            1
                5.0            3.4           1.5            0.2            1
                               3.6           1.6            0.4            1
                                                                           ..
Iris-virginica  6.5            3.0           5.8            2.2            1
                               3.2           5.1            2.0            1
                6.7            2.5           5.8            1.8            1
                               3.0           5.2            2.3            1
                7.9            3.8           6.4            2.0            1
Name: count, Length: 147, dtype: int64
```

Geometric Mean and Harmonic Mean

```
# gemetric mean of the data
print(data.groupby('Species').apply(gmean).to_markdown())

# Harmonic mean of the data
print(data.groupby('Species').apply(scipy.stats.hmean).to_markdown())
```

[11] ✓ 0.0s Python

```
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\tabulate\__init__.py:107: FutureWarning: elementwise comparison failed; returning scalar instead, but in the fut
  (len(row) >= 1 and row[0] == SEPARATING_LINE)
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\tabulate\__init__.py:108: FutureWarning: elementwise comparison failed; returning scalar instead, but in the fut
  or (len(row) >= 2 and row[1] == SEPARATING_LINE)
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\tabulate\__init__.py:107: FutureWarning: elementwise comparison failed; returning scalar instead, but in the fut
  (len(row) >= 1 and row[0] == SEPARATING_LINE)
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\tabulate\__init__.py:108: FutureWarning: elementwise comparison failed; returning scalar instead, but in the fut
  or (len(row) >= 2 and row[1] == SEPARATING_LINE)
| Species         | 0                                          |
|:----------------|:-------------------------------------------|
| Iris-setosa     | [4.99384106 3.3969062  1.45373856 0.22346247] |
| Iris-versicolor | [5.91397936 2.75187353 4.23308089 1.31118738] |
| Iris-virginica  | [6.55779452 2.95701356 5.52578887 2.00721413] |
| Species         | 0                                          |
|:----------------|:-------------------------------------------|
| Iris-setosa     | [4.9816814  3.37529787 1.44318022 0.20449898] |
| Iris-versicolor | [5.89193597 2.73301053 4.20449935 1.29602855] |
| Iris-virginica  | [6.52735823 2.94008724 5.50023704 1.98786729] |
```

Standard Devation

```
# Standard Deviation of the data
print(data.groupby('Species').std().to_markdown())
```

[12] ✓ 0.0s Python

```
| Species         |   SepalLengthCm |   SepalWidthCm |   PetalLengthCm |   PetalWidthCm |
|:----------------|----------------:|---------------:|----------------:|---------------:|
| Iris-setosa     |         0.35249 |       0.381024 |        0.173511 |        0.10721 |
| Iris-versicolor |         0.516171 |       0.313798 |        0.469911 |        0.197753 |
| Iris-virginica  |         0.63588 |       0.322497 |        0.551895 |        0.27465 |
```

```
# Standard Deviation of the data using numpy
print(data.groupby('Species').agg(np.std).to_markdown())
```

[13] ✓ 0.0s Python

```
| Species         |   SepalLengthCm |   SepalWidthCm |   PetalLengthCm |   PetalWidthCm |
|:----------------|----------------:|---------------:|----------------:|---------------:|
| Iris-setosa     |         0.35249 |       0.381024 |        0.173511 |        0.10721 |
| Iris-versicolor |         0.516171 |       0.313798 |        0.469911 |        0.197753 |
| Iris-virginica  |         0.63588 |       0.322497 |        0.551895 |        0.27465 |
```

```
# Standard Deviation Using Formula
variance = data.groupby('Species').var()
standard_deviation = np.sqrt(variance)
print(standard_deviation.to_markdown())
```

[14] ✓ 0.0s Python

```
| Species         |   SepalLengthCm |   SepalWidthCm |   PetalLengthCm |   PetalWidthCm |
|:----------------|----------------:|---------------:|----------------:|---------------:|
| Iris-setosa     |         0.35249 |       0.381024 |        0.173511 |        0.10721 |
| Iris-versicolor |         0.516171 |       0.313798 |        0.469911 |        0.197753 |
| Iris-virginica  |         0.63588 |       0.322497 |        0.551895 |        0.27465 |
```

Variance

```
# Variance of the data
print(data.groupby('Species').var().to_markdown())
```

[15] ✓ 0.0s Python

Species	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Iris-setosa	0.124249	0.14518	0.0301061	0.0114939
Iris-versicolor	0.266433	0.0984694	0.220816	0.0391061
Iris-virginica	0.404343	0.104004	0.304588	0.0754327

```
# Variance of the data using numpy
print(data.groupby('Species').agg(np.var).to_markdown())
```

[16] ✓ 0.0s Python

Species	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Iris-setosa	0.124249	0.14518	0.0301061	0.0114939
Iris-versicolor	0.266433	0.0984694	0.220816	0.0391061
Iris-virginica	0.404343	0.104004	0.304588	0.0754327

```
# Variance of the data using formula
standard_deviation = data.groupby('Species').std()
variance = np.square(standard_deviation)
print(variance.to_markdown())
```

[17] ✓ 0.0s Python

Species	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Iris-setosa	0.124249	0.14518	0.0301061	0.0114939
Iris-versicolor	0.266433	0.0984694	0.220816	0.0391061
Iris-virginica	0.404343	0.104004	0.304588	0.0754327

THE NORTHCAP UNIVERSITY

EXPERIMENT NO. 3

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL227-ACS-Lab_Manual (github.com)
Date: 04.02.2023
Faculty Signature:
Marks/Grade:

Objective(s):
Find skewness and kurtosis for a given dataset

Outcome: Understand skewness and kurtosis and their implementation

Problem Statement: Demonstrate measures of shape using Python

Background Study:
Descriptive statistics are brief descriptive coefficients that summarize a given data set, which can be either a representation of the entire or a sample of a population. Descriptive statistics are broken down into measures of central tendency and measures of variability (spread). Measures of central tendency include the mean, median and mode, while measures of variability include standard deviation, variance, minimum and maximum variables, and kurtosis and skewness.

Question Bank:

1. Can we identify outlier from stem-and-leaf plot?
   Yes, we can identify outliers from a stem-and-leaf plot, although it may not be as straightforward as identifying them from other types of plots like box plots or histograms.
   In a stem-and-leaf plot, outliers are typically defined as values that are much smaller or much larger than the other values in the data set. This can be assessed by looking for any "leaves" that are significantly separated from the rest of the data points.

2. What is kernel density estimation?
   Kernel Density Estimation (KDE) is a non-parametric technique used for estimating the probability density function (PDF) of a random variable. It is particularly useful when the underlying distribution of the data is unknown, or when a smooth estimate of the density is needed.

3. How do we decide upon the number of bins in histogram.
   The number of bins in a histogram can have a significant impact on the way the data is presented and interpreted, so it is important to choose an appropriate number of bins.

4. Which plot is used for analysis of bivariate data?
   Scatter plot is the most common plot used for analyzing the relationship between two continuous variables. Other plots include line plots, bubble plots, contour plots, and heatmaps.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 3

Demonstrate measures of shape using Python.

```python
# importing required libraries
import math
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as snss

import scipy
from scipy import stats
from scipy.stats import zscore, kurtosis, variation, scoreatpercentile
from scipy.stats.mstats import gmean

```

[1] ✓ 1.6s Python

```python
# Read the data
data = pd.read_csv('iris_dataset.csv')

```

[2] ✓ 0.0s Python

```python
# Drop the Id column
data = data.drop('Id', axis=1)

```

[3] ✓ 0.0s Python

```python
# Print the first 5 rows of the data
print(data.head().to_markdown())

```

[4] ✓ 0.0s Python

```
|    |   SepalLengthCm |   SepalWidthCm |   PetalLengthCm |   PetalWidthCm | Species     |
|---:|----------------:|---------------:|----------------:|---------------:|:------------|
|  0 |             5.1 |            3.5 |             1.4 |            0.2 | Iris-setosa |
|  1 |             4.9 |            3   |             1.4 |            0.2 | Iris-setosa |
|  2 |             4.7 |            3.2 |             1.3 |            0.2 | Iris-setosa |
|  3 |             4.6 |            3.1 |             1.5 |            0.2 | Iris-setosa |
|  4 |             5   |            3.6 |             1.4 |            0.2 | Iris-setosa |
```

```python
# Information about the data
print(data.info())

```

[5] ✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
```

```
# Describe the data
print(data.groupby('Species').describe())

```

[6] ✓ 0.1s Python

```
                SepalLengthCm
                        count   mean       std  min    25%  50%  75%  max
Species
Iris-setosa              50.0  5.006  0.352490  4.3  4.800  5.0  5.2  5.8  \
Iris-versicolor          50.0  5.936  0.516171  4.9  5.600  5.9  6.3  7.0
Iris-virginica           50.0  6.588  0.635880  4.9  6.225  6.5  6.9  7.9

                SepalWidthCm         ... PetalLengthCm      PetalWidthCm
                       count   mean  ...           75%  max        count
Species                                ...
Iris-setosa             50.0  3.418  ...         1.575  1.9         50.0  \
Iris-versicolor         50.0  2.770  ...         4.600  5.1         50.0
Iris-virginica          50.0  2.974  ...         5.875  6.9         50.0


                  mean       std  min  25%  50%  75%  max
Species
Iris-setosa      0.244  0.107210  0.1  0.2  0.2  0.3  0.6
Iris-versicolor  1.326  0.197753  1.0  1.2  1.3  1.5  1.8
Iris-virginica   2.026  0.274650  1.4  1.8  2.0  2.3  2.5

[3 rows x 32 columns]
```

Z-Score

```
# Z-Score of the data
print(data.groupby('Species').apply(zscore).to_markdown())

```

[7] ✓ 0.0s Python

```
Output exceeds the size limit. Open the full output data in a text editor
|                        |   SepalLengthCm |   SepalWidthCm |   PetalLengthCm |   PetalWidthCm |
|:-----------------------|----------------:|---------------:|----------------:|---------------:|
| ('Iris-setosa', 0)     |        0.269382 |       0.217394 |       -0.372597 |       -0.414578 |
| ('Iris-setosa', 1)     |       -0.303771 |      -1.10818  |       -0.372597 |       -0.414578 |
| ('Iris-setosa', 2)     |       -0.876924 |      -0.577951 |       -0.95478  |       -0.414578 |
| ('Iris-setosa', 3)     |       -1.1635   |      -0.843065 |        0.209586 |       -0.414578 |
| ('Iris-setosa', 4)     |       -0.0171946 |       0.482509 |       -0.372597 |       -0.414578 |
| ('Iris-setosa', 5)     |        1.12911  |       1.27785  |        1.37395  |        1.46987  |
| ('Iris-setosa', 6)     |       -1.1635   |      -0.0477207 |       -0.372597 |        0.527645 |
| ('Iris-setosa', 7)     |       -0.0171946 |      -0.0477207 |        0.209586 |       -0.414578 |
| ('Iris-setosa', 8)     |       -1.73665  |      -1.3733   |       -0.372597 |       -0.414578 |
| ('Iris-setosa', 9)     |       -0.303771 |      -0.843065 |        0.209586 |       -1.3568   |
| ('Iris-setosa', 10)    |        1.12911  |       0.747624 |        0.209586 |       -0.414578 |
| ('Iris-setosa', 11)    |       -0.590348 |      -0.0477207 |        0.791769 |       -0.414578 |
| ('Iris-setosa', 12)    |       -0.590348 |      -1.10818  |       -0.372597 |       -1.3568   |
| ('Iris-setosa', 13)    |       -2.02323  |      -1.10818  |       -2.11915  |       -1.3568   |
| ('Iris-setosa', 14)    |        2.27542  |       1.54297  |       -1.53696  |       -0.414578 |
| ('Iris-setosa', 15)    |        1.98884  |       2.60343  |        0.209586 |        1.46987  |
| ('Iris-setosa', 16)    |        1.12911  |       1.27785  |       -0.95478  |        1.46987  |
| ('Iris-setosa', 17)    |        0.269382 |       0.217394 |       -0.372597 |        0.527645 |
| ('Iris-setosa', 18)    |        1.98884  |       1.01274  |        1.37395  |        0.527645 |
| ('Iris-setosa', 19)    |        0.269382 |       1.01274  |        0.209586 |        0.527645 |
| ('Iris-setosa', 20)    |        1.12911  |      -0.0477207 |        1.37395  |       -0.414578 |
| ('Iris-setosa', 21)    |        0.269382 |       0.747624 |        0.209586 |        1.46987  |
| ('Iris-setosa', 22)    |       -1.1635   |       0.482509 |       -2.70133  |       -0.414578 |
...
| ('Iris-virginica', 146) |       -0.457514 |      -1.4847   |       -1.01035  |       -0.463423 |
| ('Iris-virginica', 147) |       -0.139796 |       0.0814395 |       -0.644278 |       -0.095627 |
| ('Iris-virginica', 148) |       -0.616373 |       1.33435  |       -0.278211 |        1.00776  |
| ('Iris-virginica', 149) |       -1.09295  |       0.0814395 |       -0.827312 |       -0.831219 |
```

Skewness

```
1  # Skewness
2  print(data.groupby('Species').skew().to_markdown())
3
```

[8] ✓ 0.0s Python

Species	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Iris-setosa	0.120087	0.107053	0.0718461	1.19724
Iris-versicolor	0.105378	-0.362845	-0.606508	-0.0311796
Iris-virginica	0.118015	0.365949	0.549445	-0.129477

Kurtosis

```
1  # Kurtosis
2  data_agg = data.groupby('Species').agg({'SepalLengthCm': kurtosis, 'SepalWidthCm': kurtosis,
3                                          'PetalLengthCm': kurtosis, 'PetalWidthCm': kurtosis}).reset_index()
4  print(data_agg.to_markdown())
5
```

[9] ✓ 0.0s Python

	Species	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	Iris-setosa	-0.345765	0.685134	0.813665	1.29648
1	Iris-versicolor	-0.598827	-0.448272	-0.0744018	-0.487833
2	Iris-virginica	-0.0879422	0.519766	-0.256472	-0.661348

EXPERIMENT NO. 4

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL227-ACS-Lab_Manual (github.com)
Date: 10.02.2023
Faculty Signature:
Marks/Grade:

Objective(s): Data visualization for qualitative and quantitative data
Outcome: Students will be familiarized with the use of python libraries such as matplotlib and seaborn
Problem Statement: Write a python program to use visualize qualitative and quantitative data
Background Study: Pareto Chart, Pie chart, bar charts and boxplot are some examples of visualization for qualitative datastem-and-leaf chart, count plot, histogram, frequency polygon and scatter plot are some examples of visualization for qualitative data
Question Bank: 1. What kind of data is generated using boxplot. A box plot is a graphical representation of the distribution of numerical data through their quartiles, allowing the observation of the spread and skewness of the data, as well as the presence of outliers. Box plots are particularly useful for comparing data between different groups or datasets. 2. Can we plot boxplot in both matplotlib and seaborn library? Yes, both Matplotlib and Seaborn libraries support the creation of box plots. In Matplotlib, you can create a box plot using the boxplot() function from the pyplot module. In Seaborn, you can create a box plot using the boxplot() function from the seaborn module.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 4 (Qualitative)

Data visualization for qualitative data.

```
1  # importing required libraries
2  import pandas as pd
3  import numpy as np
4  import matplotlib.pyplot as plt
5  import seaborn as sns
6
7  %matplotlib inline
```

[1] ✓ 2.7s Python

```
1  # Read the data
2  housing = pd.read_csv('housing_dataset.csv')
```

[2] ✓ 0.1s Python

```
1  # Printing the First 5 Rows of the Data
2  print(housing.head().to_markdown())
3
```

[3] ✓ 0.0s Python

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41	880	129	322	126	8.3252	452600	NEAR BAY
1	-122.22	37.86	21	7099	1106	2401	1138	8.3014	358500	NEAR BAY
2	-122.24	37.85	52	1467	190	496	177	7.2574	352100	NEAR BAY
3	-122.25	37.85	52	1274	235	558	219	5.6431	341300	NEAR BAY
4	-122.25	37.85	52	1627	280	565	259	3.8462	342200	NEAR BAY

```
1  # Describing the Data
2  print(housing.describe().to_markdown())
```

[4] ✓ 0.1s Python

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640	20640	20640	20640	20433	20640	20640	20640	20640
mean	-119.57	35.6319	28.6395	2635.76	537.871	1425.48	499.54	3.87067	206856
std	2.00353	2.13595	12.5856	2181.62	421.385	1132.46	382.33	1.89982	115396
min	-124.35	32.54	1	2	1	3	1	0.4999	14999
25%	-121.8	33.93	18	1447.75	296	787	280	2.5634	119600
50%	-118.49	34.26	29	2127	435	1166	409	3.5348	179700
75%	-118.01	37.71	37	3148	647	1725	605	4.74325	264725
max	-114.31	41.95	52	39320	6445	35682	6082	15.0001	500001

```
1  # Information about the Data
2  print(housing.info())
```

[5] ✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
None
```

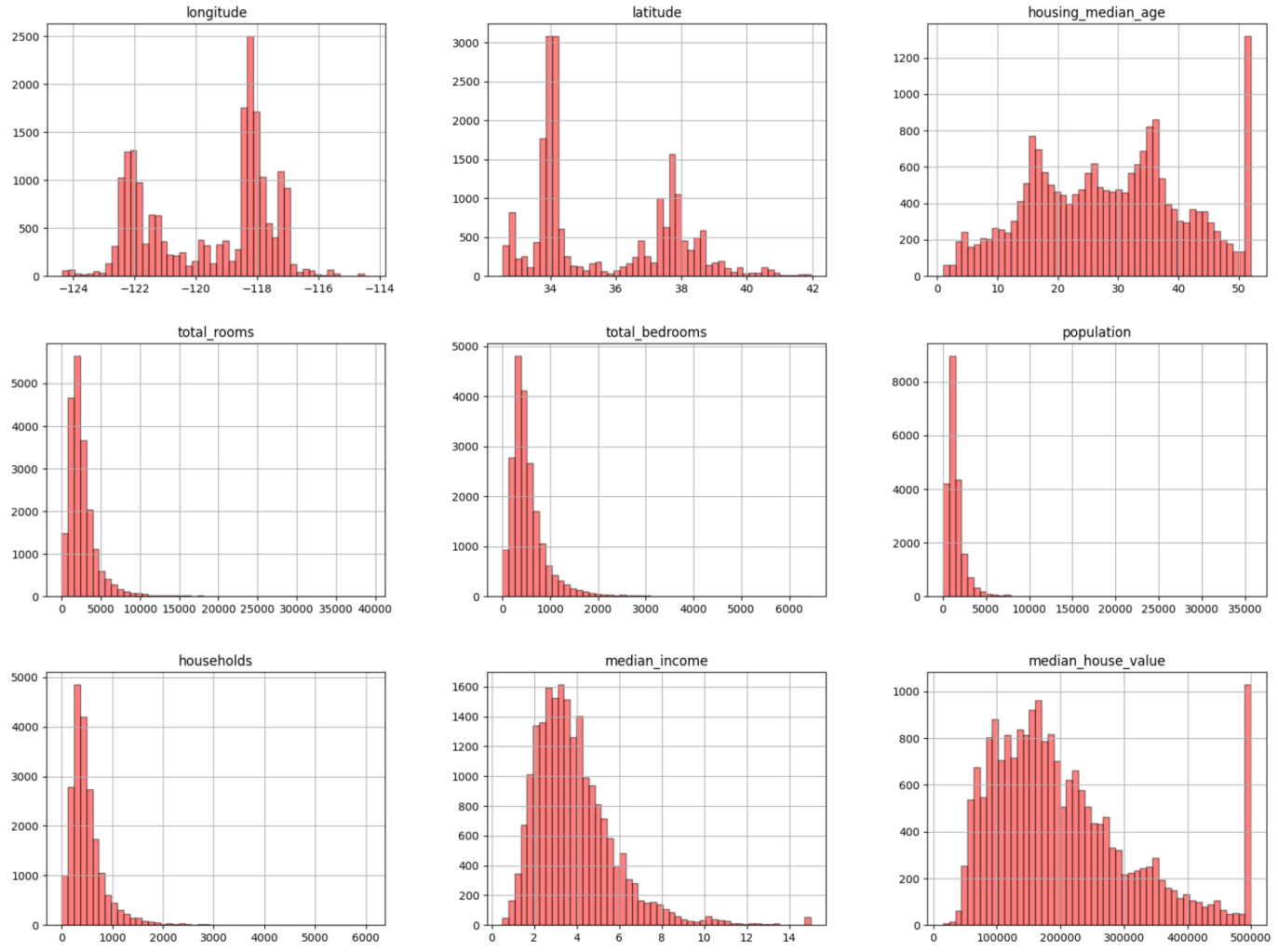
```
1  # Checking for Missing Values
2  print(housing.isnull().sum())
```

[6] ✓ 0.1s Python

```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64
```

```
1  # Plotting Histograms for each Numerical Feature
2  housing.hist(bins=50, figsize=(20, 15), color='red',
3               alpha=0.5, edgecolor='black')
4  plt.show()
```

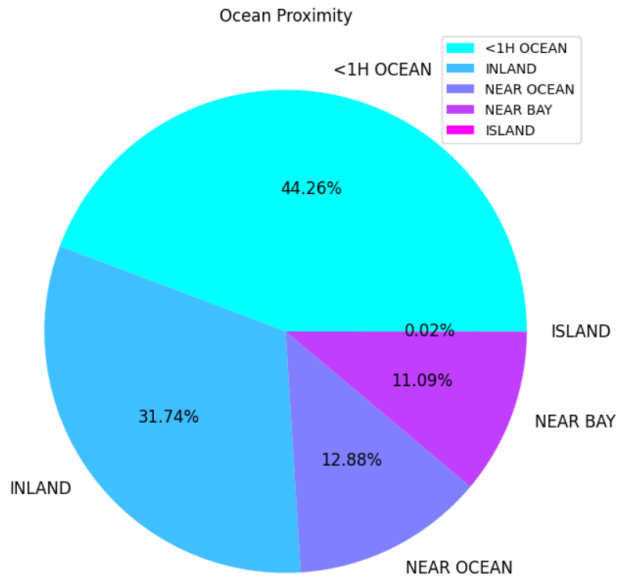
[7] ✓ 2.4s Python



```
# Plotting Pie Chart for the ocean-proximity feature
housing['ocean_proximity'].value_counts().plot(kind='pie', figsize=(20, 8), ylabel=" ", autopct='%1.2f%%', legend=True, fontsize=12, cmap='cool', title='Ocean Proximity')
plt.legend(loc='upper right')
plt.show()
```

[14] ✓ 0.2s Python

...



Experiment 4 (Quantitative)

Data visualization for quantitative data.

```python
# importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pylab

%matplotlib inline
```

[1] ✓ 1.9s Python

```python
# Loading the dataset
data = pd.read_csv('heart_dataset.csv')
```

[2] ✓ 0.0s Python

```python
# Prining the first 5 rows of the dataset
print(data.head().to_markdown())
```

[3] ✓ 0.0s Python

```
|    |   age |   sex |   cp |   trestbps |   chol |   fbs |   restecg |   thalach |   exang |   oldpeak |   slope |   ca |   thal |   target |
|---:|------:|------:|-----:|-----------:|-------:|------:|----------:|----------:|--------:|----------:|--------:|-----:|-------:|---------:|
|  0 |    52 |     1 |    0 |        125 |    212 |     0 |         1 |       168 |       0 |       1   |       2 |    2 |      3 |        0 |
|  1 |    53 |     1 |    0 |        140 |    203 |     1 |         0 |       155 |       1 |       3.1 |       0 |    0 |      3 |        0 |
|  2 |    70 |     1 |    0 |        145 |    174 |     0 |         1 |       125 |       1 |       2.6 |       0 |    0 |      3 |        0 |
|  3 |    61 |     1 |    0 |        148 |    203 |     0 |         1 |       161 |       0 |       0   |       2 |    1 |      3 |        0 |
|  4 |    62 |     0 |    0 |        138 |    294 |     1 |         1 |       106 |       0 |       1.9 |       1 |    3 |      2 |        0 |
```

```python
# Information about the dataset
print(data.info())

```

[4] ✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1025 non-null   int64  
 1   sex       1025 non-null   int64  
 2   cp        1025 non-null   int64  
 3   trestbps  1025 non-null   int64  
 4   chol      1025 non-null   int64  
 5   fbs       1025 non-null   int64  
 6   restecg   1025 non-null   int64  
 7   thalach   1025 non-null   int64  
 8   exang     1025 non-null   int64  
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64  
 11  ca        1025 non-null   int64  
 12  thal      1025 non-null   int64  
 13  target    1025 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
None
```

```
[5]
# Check for missing values
print(data.isnull().sum())
```

✓ 0.0s Python

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

```
[6]
# Dropping the missing values
data.dropna(inplace=True)
```

✓ 0.0s Python

```
[7]
# Information about the dataset
print(data.info())
```

✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
None
```

```
[8]
# Checking the first 5 rows of the dataset
print(data.head().to_markdown())

```

✓ 0.0s Python

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

```
[9]
# Extract the age (relevant column) for the stem and leaf plot
age_data = data['age'].head(10).tolist()

# Convert the values in the data list to integers
age_list = [int(value) for value in age_data]
```

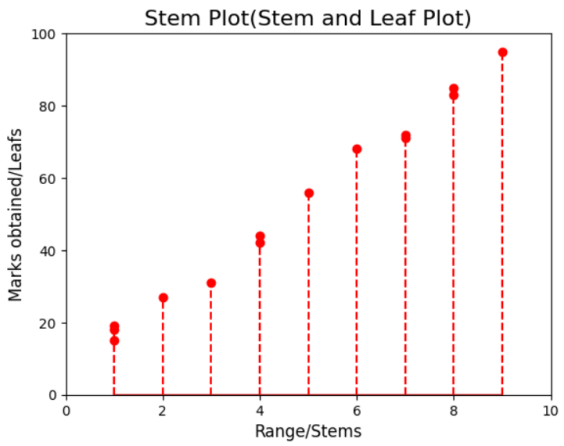
✓ 0.0s Python

Leaf and Stem Plot

```
# Create the stem and leaf plot

# marks obtained by students in an examination
marks = [15, 18, 19, 27, 31, 42, 44, 56, 68, 71, 72, 83, 85, 95]
# corresponding stems
stems = [1, 1, 1, 2, 3, 4, 4, 5, 6, 7, 7, 8, 8, 9]

# set the x axis and y axis limits
pylab.xlim([0, 10])
pylab.ylim([0, 100])

# Provide a title for the stem plot
plt.title('Stem Plot(Stem and Leaf Plot)', fontsize=16)

# Give x and y label for the stem plot
plt.xlabel('Range/Stems', fontsize=12)
plt.ylabel('Marks obtained/Leafs', fontsize=12)

# plot the stem plot using matplotlib
markerline, stemlines, baseline = plt.stem(
    stems, marks, '--')

plt.setp(markerline, color='red')
plt.setp(stemlines, color='red')

# display the stem plot
plt.show()

```

[10] ✓ 0.2s Python

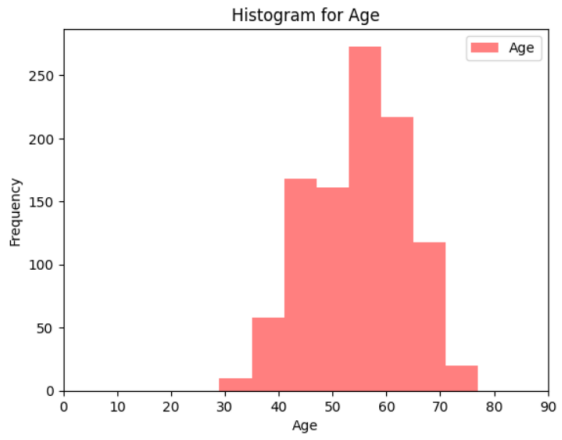
...



Histogram: For Univariate Variable

```
# Plotting histogram for age
plt.hist(data['age'], bins = 8, color='red', alpha=0.5)
plt.title('Histogram for Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.xticks(range(0, 100, 10))
plt.yticks(range(0, 300, 50))
plt.legend(['Age'])
plt.show()
```

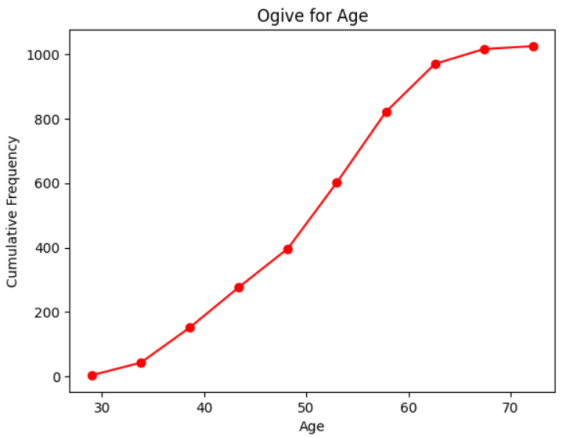
[11] ✓ 0.2s Python



Ogive /Cumulative Frequency Polygon Chart for Univariate Quantitative data

```

# we can use the numpy.histogram function to automatically find the classes and the class frequencies.
# Then we can use matplotlib to actually create the ogive:

# obtain histogram values with 10 bins
values, base = np.histogram(data['age'], bins=10)

# find the cumulative sums
cumulative = np.cumsum(values)

# plot the ogive
plt.plot(base[:-1], cumulative, 'ro-')
plt.xlabel('Age')
plt.ylabel('Cumulative Frequency')
plt.title('Ogive for Age')
plt.show()
```

[12] ✓ 0.2s Python



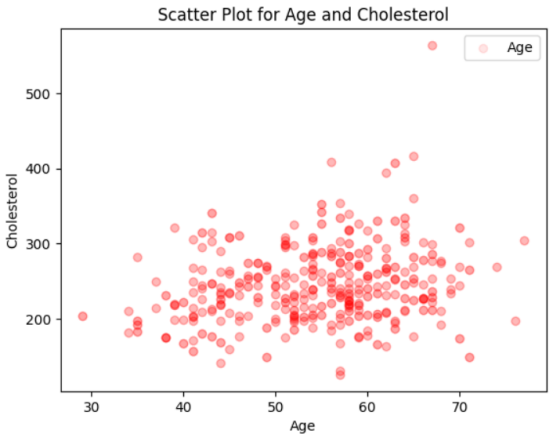
Scatter Plot

```
[13] ✓ 0.3s
```

```python
# Plotting Scatter Plot
plt.scatter(data['age'], data['chol'], color='red', alpha = 0.1)
plt.title('Scatter Plot for Age and Cholesterol')
plt.xlabel('Age')
plt.ylabel('Cholesterol')
plt.legend(['Age', 'Weight'])
plt.show()
```

Python

...



EXPERIMENT NO. 5

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL227-ACS-Lab_Manual (github.com)
Date: 24.02.2023
Faculty Signature:
Marks/Grade:

Objective(s): Write python program to implement Probability
Problem Statement: Basics of probability using Python
Background Study: Various types of events, conditional probability, permutation and combination

Question Bank:

1. What are mutually exclusive events?
   Mutually exclusive events are events that cannot occur at the same time. In other words, if one event happens, the other event cannot happen. For example, if you flip a coin, the result can be either heads or tails, but not both. Therefore, "getting heads" and "getting tails" are mutually exclusive events. Similarly, if you roll a dice, getting a 1 and getting a 6 are also mutually exclusive events. In probability theory, the probability of the union of mutually exclusive events is equal to the sum of the probabilities of each individual event.

2. What is the importance of studying probability? Relate with real-life situations and explain.
   It helps us make informed decisions based on the likelihood of different outcomes. Probability is used in a wide range of real-life situations, from weather forecasting to sports betting to business decision-making. By understanding the probability of different events occurring, we can make more accurate predictions, assess risks more effectively, and make better-informed decisions. Probability is a fundamental concept in mathematics and statistics, and its applications extend to many different fields. Whether we are trying to predict the weather, assess risks in business or medicine, or make smart bets in sports, probability helps us make sense of the world around us and make better decisions based on the available data.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

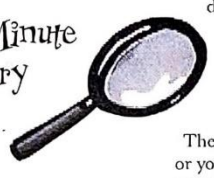
DRILL-9

The Case of the Two Classes

The Head First Health Club prides itself on its ability to find a class for everyone. As a result, it is extremely popular with both young and old.

The Health Club is wondering how best to market its new yoga class, and the Head of Marketing wonders if someone who goes swimming is more likely to go to a yoga class. "Maybe we could offer some sort of discount to the swimmers to get them to try out yoga."

Five Minute Mystery



The CEO disagrees. "I think you're wrong," he says. "I think that people who go swimming and people who go to yoga are independent. I don't think people who go swimming are any more likely to do yoga than anyone else."

They ask a group of 96 people whether they go to the swimming or yoga classes. Out of these 96 people, 32 go to yoga and 72 go swimming. 24 people are exceptionally eager and go to both.

So who's right? Are the yoga and swimming classes dependent or independent?

$P(\text{Yoga}) = \frac{32}{96} = \frac{1}{3}$

$P(\text{Swim}) = \frac{72}{96} = \frac{3}{4}$

$P(Y \cap S) = \frac{24}{96} = \frac{1}{4}$

$P(Y) \times P(S) = \frac{1}{3} \times \frac{3}{4} = \frac{1}{4}$

$= P(Y \cap S)$

Manic Mango selects one of the volunteers at random to ask if she enjoyed playing the game, and she says she did. Given that the volunteer enjoyed playing the game, what's the probability that she played game 2? Use Bayes' Theorem.

Hint: What's the probability of someone choosing game 2 and being satisfied? What's the probability of someone being satisfied overall? Once you've found these, you can use Bayes Theorem to obtain the right answer.

$P(\text{Game2} / \text{Enjoyed})$

$\rightarrow P(G_2/E) = \dfrac{P(G_2) \cdot P(E/G_2)}{P(G_2) \cdot P(E/G_2) + P(G_1) \cdot P(E/G_1)}$

$= \dfrac{0.2 \times 0.7}{(0.2 \times 0.7) + (0.8 \times 0.6)}$

$= \dfrac{0.14}{0.14 + 0.48}$

$= \dfrac{0.14}{0.62} = 0.226$

long exercise

Long Exercise — DRILL-7, 8

The Manic Mango games company is testing two brand-new games. They've asked a group of volunteers to choose the game they most want to play, and then tell them how satisfied they were with game play afterwards.

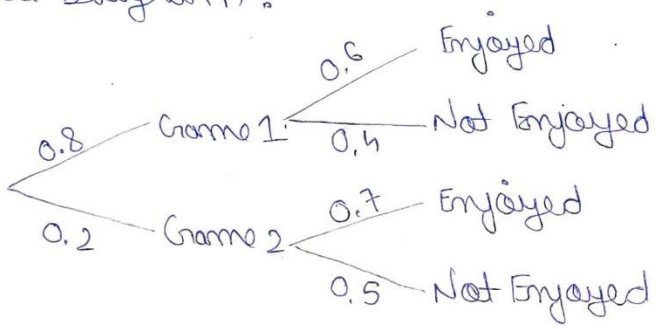
80 percent of the volunteers chose Game 1, and 20 percent chose Game 2. Out of the Game 1 players, 60 percent enjoyed the game and 40 percent didn't. For Game 2, 70 percent of the players enjoyed the game and 30 percent didn't.

Your first task is to fill in the probability tree for this scenario.

P(Game 1) = .80
P(Game 2) = .20

Game 1
P(Enjoy) = .60 P(Enjoy) = .70
P(Didn't Enjoy) = .40 P(Didn't Enjoy) = .30

Tree Diagram:

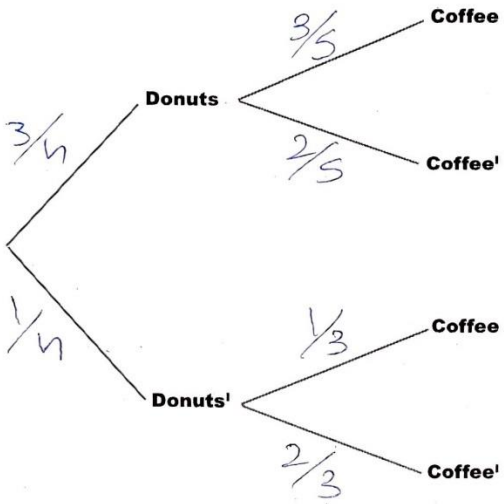


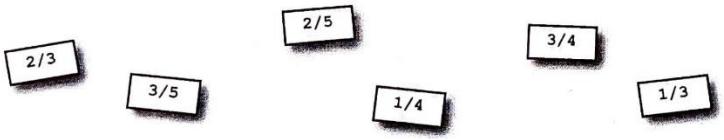
probability magnets

Probability Magnets

Duncan's Donuts are looking into the probabilities of their customers buying donuts *and* coffee. They drew up a probability tree to show the probabilities, but in a sudden gust of wind, they all fell off. Your task is to pin the probabilities back on the tree. Here are some clues to help you.

P(Donuts) = 3/4 P(Coffee | Donuts') = 1/3 P(Donuts ∩ Coffee) = 9/20





DRILL-6

Exercise

We haven't quite finished with Duncan's Donuts! Now that you've completed the probability tree, you need to use it to work out some probabilities.

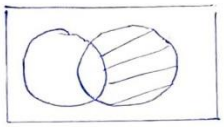
$P(D) = 3/4$ $P(C \cap D) = 1/3$ $P(D \cap C) = 9/20$

1. P(Donuts')

$P(D') = 1 - P(D)$
$= 1 - \frac{3}{4}$
$= \frac{1}{4}$

2. P(Donuts' ∩ Coffee)

$\frac{1}{3} \times \frac{1}{4} = \frac{1}{12}$



3. P(Coffee' | Donuts)

$P(C' | D) = \frac{2}{5}$

4. P(Coffee) ← Hint: How many ways are there of getting coffee? (You can get coffee with or without donuts.)

$P(C) = P(D' \cap C) + P(D \cap C)$
$= \frac{1}{12} + \frac{9}{20} = \frac{8}{15}$

← Hint: maybe some of your other answers can help you.

5. P(Donuts | Coffee)

$P(D|C) = \frac{9/20}{8/15} = \frac{27}{32}$

$n(\text{Basketball}) = 18 + 6 = 28$

$n(\text{Football}) = 12 + 4 = 16$

$n(\text{Baseball}) = 10 + 6 = 16$

$P(\text{Baseball} \cap \text{Football}) = 0$

$\therefore$ Basketball and Football are mutually exclusive.

$P(\text{Baseball} \cup \text{Football} \cup \text{Basketball}) = 1$

So, the events for baseball, football and basketball are exhaustive.

DRILL-4

Exercise

50 sports enthusiasts at the Head First Health Club are asked whether they play baseball, football, or basketball. 10 only play baseball. 12 only play football. 18 only play basketball. 6 play baseball and basketball but not football. 4 play football and basketball but not baseball.

Draw a Venn diagram for this probability space. How many enthusiasts play baseball in total? How many play basketball? How many play football?

Are any sports' rosters mutually exclusive? Which sports are exhaustive (fill up the possibility space)?

B - Basketball

K - Basketball

Total = 50

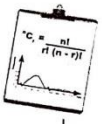
n P(B) = 10

n P(F) = 12

n P(K) = 18

n (B ∩ K) = 6

n (F ∩ K) = 4



Vital Statistics

A or B

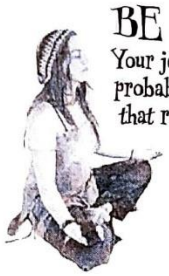
To find the probability of getting event A or B, use

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

$\cup$ means OR

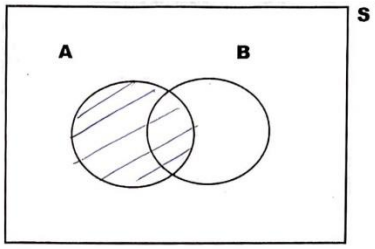
$\cap$ means AND

DRILL-3

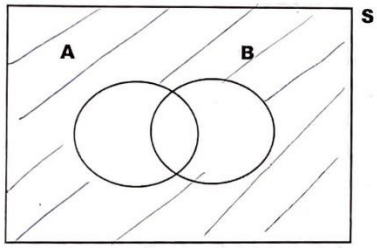


BE the probability

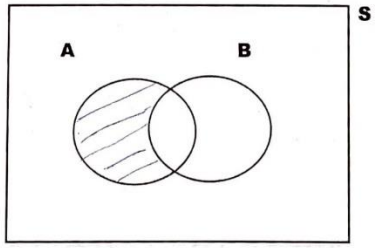
Your job is to play like you're the probability and shade in the area that represents each of the following probabilities on the Venn diagrams.



$P(A \cap B) + P(A \cap B')$



$P(A' \cap B')$



$P(A \cup B) - P(B)$

Sharpen your pencil DRILL-1,2

Let's find the probability of getting a black or even (assume 0 and 00 are not even).

1. What's the probability of getting a black?

$$P(\text{Getting Black}) = \frac{18}{38} = 0.474$$

2. What's the probability of getting an even number?

$$P(\text{Even Number}) = \frac{18}{38} = 0.474$$

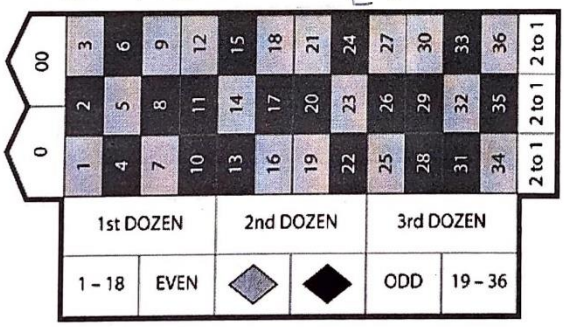
3. What do you get if you add these two probabilities together?

$$\text{Adding Probabilities} = 0.947 \left[\frac{18}{38} + \frac{18}{38}\right]$$

4. Finally, use your roulette board to count all the holes that are either black or even, then divide by the total number of holes. What do you get?

$$P\left(\frac{\text{Black or Even}}{\text{Total Number of Holes}}\right) = \frac{26}{38} = 0.684$$

$$\left[\frac{18}{38} + \frac{18}{35} - \frac{10}{38} = \frac{26}{38}\right]$$



We can say that the classes are indepen-
- dent.

EXPERIMENT NO. 6

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL227-ACS-Lab_Manual (github.com)
Date: 03.03.2023
Faculty Signature:
Marks/Grade:

Objective(s): Understand and apply the laws of probability
Problem Statement: To implement laws of probability
Background Study: Laws of probability-General and special laws of addition and multiplication, concepts of conditional probability and Bayes theorem.

Question Bank:

1. What is the difference between independent and dependent events?

   In probability theory, independent events are events that do not affect each other, meaning the occurrence of one event does not change the probability of the other event happening. In contrast, dependent events are events where the occurrence of one event affects the probability of the other event happening. In other words, the outcome of one event is dependent on the outcome of the other event.

2. What is the relationship between conditional probability ad bayes theorem?

   Bayes' theorem is a way to calculate conditional probabilities. It states that the probability of an event A given that another event B has occurred is equal to the probability of event B given event A multiplied by the prior probability of event A, divided by the prior probability of event B. In mathematical notation, this can be expressed as:
   P(A|B) = P(B|A) * P(A) / P(B)

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

```
"""
Experiment 6
Write Python Program to Implement Laws of Probability
"""

# Addition Law of Probability
def addition_law(a, b):
    return a + b - (a * b)

# Multiplication Law of Probability
def multiplication_law(a, b):
    return a * b

# Conditional Probability
def conditional_probability(a, b):
    return (a * b) / b

# Total Probability Theorem
def total_probability_theorem(probability):

    event = int(input("Enter the Event Number: "))
    probability_event = 0
    for i in range(len(probability)):
        probability_event_given_i = conditional_probability(probability[event-1],
probability[i])
        probability_event += probability[i] * probability_event_given_i

    print("\nThe Probability if the {} is : {}".format(event, probability_event))


# Complement of an Event
def complement(a):
    return 1 - a


def main():
    # Menu for user to choose which law of probability to use
    print("1. Addition Law of Probability")
    print("2. Multiplication Law of Probability")
    print("3. Conditional Probability")
```

```
print("4. Complement of an Event")
print("5. Total Probability Theorem")
print("6. Exit")
choice = int(input("\nEnter Your Choice: "))

# Addition Law of Probability
if choice == 1:
    a = float(input("Enter the probability of event A: "))
    b = float(input("Enter the probability of event B: "))

    if(a > 1 or b > 1):
        print("Probability cannot be greater than 1")
        exit()
    print("\n")
    print("P(A or B) = ", addition_law(a, b))

# Multiplication Law of Probability
elif choice == 2:
    a = float(input("Enter the probability of event A: "))
    b = float(input("Enter the probability of event B: "))
    if(a > 1 or b > 1):
        print("Probability cannot be greater than 1")
        exit()
    print("\n")
    print("P(A and B) = ", multiplication_law(a, b))

# Conditional Probability
elif choice == 3:
    a = float(input("Enter the probability of event A: "))
    b = float(input("Enter the probability of event B: "))
    if(a > 1 or b > 1):
        print("Probability cannot be greater than 1")
        exit()
    print("\n")
    print("P(A | B) = ", conditional_probability(a, b))

# Complement of an Event
elif choice == 4:
    a = float(input("Enter the probability of event A: "))
    print("P(not A) = ", complement(a))

# Total Probability Theorem
elif choice == 5:
    probability = []
    number_of_events = int(input("Enter the number of events: "))
```

```
        for i in range(number_of_events):
            inp = float(input("Enter the probability of event {}: ".format(i + 1)))
            if(inp > 1):
                print("Probability cannot be greater than 1")
                exit()
            probability.append(inp)

    # Exit
    elif choice == 6:
        exit()

    # Invalid Choice
    else:
        print("Invalid Choice")


main()
```

```
PowerShell
PS C:\Users\mainp\OneDrive\College Material\4. 2nd Year (EVEN Semester)\CSL227 - Applied Computational Statistics\Lab Manual\NCU-CSL227-ACS-Lab_Manual\Experiment 6> python experiment_6.py
1. Addition Law of Probability
2. Multiplication Law of Probability
3. Conditional Probability
4. Complement of an Event
5. Total Probability Theorem
6. Exit

Enter Your Choice: 1
Enter the probability of event A: 0.24
Enter the probability of event B: 0.14


P(A or B) =  0.3464
PS C:\Users\mainp\OneDrive\College Material\4. 2nd Year (EVEN Semester)\CSL227 - Applied Computational Statistics\Lab Manual\NCU-CSL227-ACS-Lab_Manual\Experiment 6> python experiment_6.py
1. Addition Law of Probability
2. Multiplication Law of Probability
3. Conditional Probability
4. Complement of an Event
5. Total Probability Theorem
6. Exit

Enter Your Choice: 4
Enter the probability of event A: 0.23
P(not A) =  0.77
PS C:\Users\mainp\OneDrive\College Material\4. 2nd Year (EVEN Semester)\CSL227 - Applied Computational Statistics\Lab Manual\NCU-CSL227-ACS-Lab_Manual\Experiment 6>
```

EXPERIMENT NO. 7

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL227-ACS-Lab_Manual (github.com)
Date: 17.03.2023
Faculty Signature:
Marks/Grade:

Objective(s): Write a python program to compute PMF in discrete distribution

Outcome: Explore the functions related to implementation of discrete distributions

Problem Statement: Implementation of probability mass function for discrete random variable using Python.

Background Study: The **Probability Mass Function (PMF)** is also called a **probability. function** or **frequency function** which characterizes the distribution of a discrete random variable. Let X be a discrete random variable of a function, then the probability mass function of a random variable X is given by

$\mathbf{P_x(x) = P(X=x)}$, For all x belongs to the range of X

It is noted that the probability function should fall on the condition:

- $\mathbf{P_x(x) \geq 0}$ **and**
- $\mathbf{\sum_{x \epsilon Range(x)} P_x(x) = 1}$

The discrete distributions under consideration are geometric distribution, binomial distribution,poisson distribution, hypergeometric distribution

Question Bank:

1. What is meant by random variable? Explain with example.
   In probability theory, a random variable is a variable that takes on different values based on the outcomes of a random event. It can be thought of as a numerical representation of the outcomes of a random process.
   For example, suppose we flip a fair coin two times and count the number of heads that appear. The random variable in this case is the number of heads, which can take on values of 0, 1, or 2.

2. How to calculate the probability for the random variable?
   To calculate the probability for a random variable, we first need to define the probability distribution of the random variable, which specifies the probability of the random variable taking on each possible value.

3. Explain the probability mass function of various discrete distributions?
   The probability mass function (PMF) is a function that specifies the probability distribution of a discrete random variable. It gives the probability that the random variable takes on each possible value.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 7

Implementation of probability mass function for discrete random variable using Python.

```
[1]
# importing required libraries
from scipy.stats import bernoulli
from scipy.stats import binom
from scipy.stats import geom
from scipy.stats import hypergeom
from scipy.stats import poisson
```

✓ 1.4s Python

1. Bernoulli Distribution

```
[2]
# defining paratemers
n = 10 # number of trials
p = 0.5 # probability of success

print('Mean' , bernoulli.mean(p))
print('Variance' , bernoulli.var(p))
```

✓ 0.0s Python

```
Mean 0.5
Variance 0.25
```

2. Binomial Distribution

```
[3]
# defining paratemers
n = 10 # number of trials
x = 7 # number of successes
p = 0.2 # probability of success

print("Mean: ", binom.mean(n, p))
print("Variance: ", binom.var(n, p))
print("Probability Mass Function: ", binom.pmf(x, n, p))
print("Cumulative Distribution Function: ", binom.cdf(x,n,p))
```

✓ 0.1s Python

```
Mean:  2.0
Variance:  1.6
Probability Mass Function:  0.000786432000000002
Cumulative Distribution Function:  0.9999220736
```

3. Geomteric Distribution

```
[4]
# defining paratemers
n = 10 # number of trials
p = 0.2 # probability of success

print("Mean: ", geom.mean(p))
print("Variance: ", geom.var(p))
print("Probability Mass Function: ", geom.pmf(3, p))
print("Cumulative Distribution Function: ", geom.cdf(3, p))
```

✓ 0.0s Python

```
Mean:  5.0
Variance:  20.0
Probability Mass Function:  0.12800000000000003
Cumulative Distribution Function:  0.488
```

4. Hypergeometric Distribution

```
# defining paratemers
n = 10 # number of trials
x = 7 # number of successes
M = 20 # number of balls in urn
N = 10 # number of balls drawn

print("Mean: ", hypergeom.mean(M, n, N))
print("Variance: ", hypergeom.var(M, n, N))
print("Probability Mass Function: ", hypergeom.pmf(x, M, n, N))
print("Cumulative Distribution Function: ", hypergeom.cdf(x, M, n, N))
```

[5] ✓ 0.1s Python

```
Mean:  5.0
Variance:  1.3157894736842106
Probability Mass Function:  0.07794063521617701
Cumulative Distribution Function:  0.9884929312173895
```

5. Poisson Distribution

```
# defining parameters
x = 5 # number of events
Lambda = 2/5 # lambda parameter

print("Mean: ", poisson.mean(Lambda))
print("Variance: ", poisson.var(Lambda))
print("Probability mass function: ", poisson.pmf(x, Lambda))
print("Cumulative distribution function: ", poisson.cdf(x, Lambda))
```

[6] ✓ 0.0s Python

```
Mean:  0.4
Variance:  0.4
Probability mass function:  5.720064392837453e-05
Cumulative distribution function:  0.999959573168 26
```

EXPERIMENT NO. 8

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL227-ACS-Lab_Manual (github.com)
Date: 17.03.2023
Faculty Signature:
Marks/Grade:

Objective(s): To find expectation and variance in a discrete distribution
Outcome: Knows how to compute mean and variance in a given discrete distribution
Problem Statement: Write a python program to find expectation and variance in discrete distribution
Background Study: Mean and variance are measure of centre and variation in data, respectively. We need to compute them for a given distribution, whether discrete or continuous
Question Bank: **1.** How do we compute the mean of a discrete distribution? To compute the mean (or expected value) of a discrete distribution, we can use the formula: E(X) = Σ x * P(X=x), where x is a possible outcome and P(X=x) is its corresponding probability. This means that we multiply each possible outcome by its probability, and then sum up these products to get the expected value. **2.** How do we compute the variance in a discrete distribution? To compute the variance of a discrete distribution, we can use the formula: Var(X) = Σ (x - E(X))^2 * P(X=x), where x is a possible outcome, E(X) is its expected value, and P(X=x) is its corresponding probability. This means that we subtract each possible outcome from its expected value, square this difference, multiply it by its probability, and then sum up these products to get the variance.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 8

Write a python program to find expectation and variance in discrete distribution.

```python
# importing required libraries
from scipy.stats import rv_discrete
```

[1] ✓ 1.2s Python

```python
# define the distribution
x = [1, 2, 3, 4]
p = [0.2, 0.3, 0.1, 0.4]

# create a discrete random variable
random_variable = rv_discrete(values=(x, p))

# calculate the expectation
expectation = random_variable.expect()

# calculate the variance
variance = random_variable.var()

print("Expectation: ", expectation)
print("Variance: ", variance)
```

[2] ✓ 0.0s Python

```
Expectation:  2.7
Variance:  1.4099999999999984
```

EXPERIMENT NO. 9

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL227-ACS-Lab_Manual (github.com)
Date: 24.03.2023
Faculty Signature:
Marks/Grade:

Objective(s): To understand various types of continuous probability distributions
Outcome: Students will be familiarized with normal and uniform distribution.
Problem Statement: Write Python Program for Continuous Probability Distributions
Background Study: Probability density function and cumulative distribution function for the normal and uniform distribution is computed

Question Bank:

1. What is the mean and variance of the uniform distribution?

   The mean and variance of the uniform distribution are:

   - Mean: (a + b) / 2, where a and b are the lower and upper bounds of the distribution.
   - Variance: $(b - a)^2 / 12$

   Note that the uniform distribution is a continuous probability distribution that assigns equal probability to all values within a given interval (a, b).

2. What is the mean and the variance of the normal distribution?

   The mean and variance of the normal distribution are:

   - Mean: μ, where μ is the mean of the population
   - Variance: σ^2, where σ is the standard deviation of the population

   Note that the normal distribution is a continuous probability distribution that is symmetric and bell-shaped, and it is characterized by two parameters, the mean and the standard deviation. The mean determines the center of the distribution, while the standard deviation determines the spread of the distribution.

3. How do we compute probability density function and cumulative distribution function for the normal distribution?

   The probability density function (PDF) of the normal distribution is $f(x) = (1 / \sigma\sqrt{(2\pi)}) * e^{(-(x-\mu)^2/(2\sigma^2))}$, and the cumulative distribution function (CDF) can be computed by standardizing the variable and using the CDF of the standard normal distribution.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 9

Write Python Program for Continuous Probability Distributions.

1. Uniform Distribution

```
# importing required libraries
from scipy.stats import uniform

# setting the values of a and b
a = 32
b = 37

# printing the mean, variance, PDF, CDF
print("Mean: ", uniform.mean(a, b))
print("Variance: ", uniform.var(a, b))
print("PDF: ", uniform.pdf(a, b))
print("CDF: ", uniform.cdf(a, b))


```

[1] ✓ 0.7s Python

```
Mean:  50.5
Variance:  114.08333333333333
PDF:  0.0
CDF:  0.0
```

2. Normal Distribution

```
# importing required libraries
from scipy.stats import norm as normal

# setting the values of mean and standard deviation
x = 1.6
mean = 8
sd = 5

# printing the mean, variance, PDF, CDF
print("Mean: ", normal.mean(x))
print("Variance: ", normal.var(x))
print("PDF: ", normal.pdf(x))
print("CDF: ", normal.cdf(x))
```

[2] ✓ 0.0s Python

```
Mean:  1.6
Variance:  1.0
PDF:  0.11092083467945554
CDF:  0.945200708300442
```

3. Chi-Square Distribution

```
# importing required libraries
from scipy.stats import chi2 as chi

# setting the values of degrees of freedom, variable, critical value
nu = 6  # degrees of freedom
x = 3  # variable
alpha = 0.24  # critical value

# printing the mean, variance, PDF, CDF, survival function, percent point function, critical value
print("Mean: ", chi.mean(nu))
print("Variance: ", chi.var(nu))
print("PDF: ", chi.pdf(x, nu))
print("CDF: ", chi.cdf(x, nu))
print("Survival Function: ", chi.sf(x, nu))
print("Percent Point Function: ", chi.ppf(alpha, nu))
print("Critical Value: ", chi.isf(alpha, nu))
```

[3] ✓ 0.0s Python

```
Mean:  6.0
Variance:  12.0
PDF:  0.12551071508349182
CDF:  0.1911531694619418
Survival Function:  0.8088468305380582
Percent Point Function:  3.3789420922671023
Critical Value:  7.974194328808714
```

4. The t-Distribution

```
# importing required libraries
from scipy.stats import t

# setting the values of degrees of freedom, variable, critical value
nu = 68 # degrees of freedom
x = 1 # variable
alpha = 0.925 # critical value

# printing the mean, variance, PDF, CDF, survival function, percent point function, critical value
print("Expectation: ", t.mean(nu))
print("Variance: ", t.var(nu))
print("Probability Distribution Function: ", t.pdf(x ,nu))
print("Cumulative Distribution Function: ", t.cdf(x, nu))
print("Survival Function (1-cdf): ", t.sf(x, nu))
print("Percent Point Function (inverse of cdf): ", t.ppf(alpha, nu))
print("Inverse Survival Function (inverse of (1-cdf)): ", t.isf(alpha, nu)) # Useful for finding the critical value
```

[4] ✓ 0.0s Python

```
Expectation:  0.0
Variance:  1.0303030303030303
Probability Distribution Function:  0.2402023823138007
Cumulative Distribution Function:  0.8395721066953445
Survival Function (1-cdf):  0.1604278933046555
Percent Point Function (inverse of cdf):  1.455979454119329
Inverse Survival Function (inverse of (1-cdf)):  -1.455979454119329
```

5. The F-Distribution

```python
# importing required libraries
from scipy.stats import f


# setting the values of degrees of freedom, variable, critical value
nu1 = 24 # degree of freedom 1
nu2 = 16 # degree of freedom 2
x = 2 # variable
alpha = 0.925 # critical value

# printing the mean, variance, PDF, CDF, survival function, percent point function, critical value
print("Expectation: ", f.mean(nu1, nu2))
print("Variance: ", f.var(nu1, nu2))
print("Probability Distribution Function: ", f.pdf(x ,nu1, nu2))
print("Cumulative Distribution Function: ", f.cdf(x, nu1, nu2))
print("Survival Function (1-cdf): ", f.sf(x, nu1, nu2))
print("Percent Point Function (inverse of cdf): ", f.ppf(alpha, nu1, nu2))
print("Inverse Survival Function (inverse of (1-cdf)): ", f.isf(alpha, nu1, nu2)) # Useful for finding the critical value
```

[5] ✓ 0.0s Python

```
Expectation:  1.1428571428571428
Variance:  0.34467120181405897
Probability Distribution Function:  0.14612805411889215
Cumulative Distribution Function:  0.922542818763759
Survival Function (1-cdf):  0.07745718123624103
Percent Point Function (inverse of cdf):  2.0171044619502414
Inverse Survival Function (inverse of (1-cdf)):  0.5257867781767004
```

EXPERIMENT NO. 10

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL227-ACS-Lab_Manual (github.com)
Date: 14.04.2023
Faculty Signature:
Marks/Grade:

Objective(s): To understand how to compute critical value of standardized test statistic.
Outcome: Students will compute critical value from the standard normal distribution table
Problem Statement: Write python code to find critical value in normal distribution
Background Study: The normal distribution is the most common type of continuous distribution for which we need to compute the standardized test statistic and the its critical value.
Question Bank: 1. What is standardized test statistic? A standardized test statistic is a value that measures how many standard deviations a sample statistic (such as the sample mean or sample proportion) is away from the expected value under the null hypothesis. It is often used in hypothesis testing to determine the statistical significance of the sample statistic. 2. How do we compute the critical value of the standardized test statistic? The critical value of the standardized test statistic is the value that separates the rejection region from the non-rejection region in a hypothesis test. It is determined based on the level of significance, the degrees of freedom, and the type of test (one-tailed or two-tailed).

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 10

Write python code to find critical value in normal distribution.

```python
# import the required libraries
from scipy.stats import norm

# Set the probability value (alpha)
alpha = 0.05

# Set the mean and standard deviation of the normal distribution
mu = 0
sigma = 1

# Calculate the critical value
critical_value = norm.ppf(1 - alpha/2, loc=mu, scale=sigma)

# Print the critical value
print("The critical value is:", critical_value)

```

[2] Python

```
The critical value is: 1.959963984540054
```

EXPERIMENT NO. 11

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL227-ACS-Lab_Manual (github.com)
Date: 28.04.2023
Faculty Signature:
Marks/Grade:

Objective(s): To understand central limit theorem and its applications
Outcome: Students will be able to apply central limit theorem on the sampling distribution
Problem Statement: Write a python program to Implement central limit theorem
Background Study: When sample size approaches 30, central limit theorem kicks in
Question Bank: 1. What is central limit theorem? The central limit theorem states that the distribution of the sample means approaches a normal distribution as the sample size increases, regardless of the underlying distribution of the population from which the samples are taken. 2. How is it related to sample size? The central limit theorem is directly related to sample size. It states that as the sample size increases, the distribution of the sample means approaches a normal distribution, regardless of the underlying distribution of the population from which the samples are taken. This means that the larger the sample size, the more accurate the estimate of the population mean will be, and the more confident we can be in our results. In other words, as the sample size increases, the variability of the sample means decreases, and the distribution becomes more concentrated around the true population mean. 3. Can we apply central limit theorem for both small and large sample size? The central limit theorem is typically only applicable for large sample sizes. While there is no specific cut-off for what constitutes a "large" sample size, a general rule of thumb is that the sample size should be at least 30 in order for the central limit theorem to apply. For sample sizes smaller than 30, the central limit theorem may not hold, and other techniques may be necessary to estimate population parameters.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 11

Write a python program to Implement central limit theorem.

```
# importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[1] ✓ 1.6s Python

Central Limit Theorem

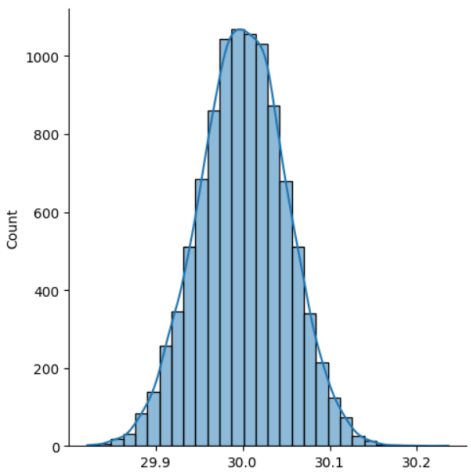
The central limit theorem states that if you have a population with mean μ and standard deviation σ and take sufficiently large random samples from the population with replacement , then the distribution of the sample means will be approximately normally distributed.

```
# implementing the central limit theorem
mu = 30
sigma = 5

samples = np.zeros(10000)

for i in range(10000):
    samples[i] = np.random.normal(mu, sigma, 10000).mean()

print("Samples mean: ", samples.mean())

sns.displot(samples, bins=30, kde=True)
```

[14] ✓ 3.4s Python

```
Samples mean:  30.000085130241704
```

```
<seaborn.axisgrid.FacetGrid at 0x2a248f9a410>
```



EXPERIMENT NO. 12

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL227-ACS-Lab_Manual (github.com)
Date:
Faculty Signature:
Marks/Grade:

Objective(s): To understands the steps of hypothesis testing for single population
Outcome: Ability to **t**est hypothesis on the population
Problem Statement: Write a python program to conduct a hypothesis test for unknown variance.
Background Study: Hypothesis testing allows the statistical evaluation of the claim made
Question Bank: 1. What is hypothesis testing? Hypothesis testing is a statistical method for making decisions or drawing conclusions about a population based on a sample of data. It involves formulating two hypotheses - a null hypothesis (H0) and an alternative hypothesis (Ha) - and testing whether the sample data provides sufficient evidence to reject the null hypothesis in favor of the alternative hypothesis. 2. What are the steps of hypothesis testing? Step 1: State the Null Hypothesis. Note. Step 2: State the Alternative Hypothesis. Step 3: Set α Step 4: Collect Data. Step 5: Calculate a test statistic. Step 6: Construct Acceptance / Rejection regions. Step 7: Based on steps 5 and 6, draw a conclusion about H0. 3. What is level of significance? The level of significance, denoted by the symbol alpha (α), is the maximum allowable probability of making a Type I error in a statistical hypothesis test. A Type I error occurs when the null hypothesis is rejected when it is actually true, which means that the test has produced a false positive result.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 12

Write a python program to conduct a hypothesis test for unknown variance.

```python
# importing required libraries
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
```

[21] ✓ 0.2s Python

One Sample T-Test

```python
# Set the random seed for reproducibility
np.random.seed(8)

# Generate a population age distribution with a mean of 40 and a size of 200,000
population_ages1 = stats.poisson.rvs(loc=18, mu=40, size=150000)
population_ages2 = stats.poisson.rvs(loc=18, mu=10, size=50000)
population_ages = np.concatenate((population_ages1, population_ages2))

# Generate a sample from a specific state (e.g., Minnesota) with a mean of 35 and a sample size of 50
minnesota_ages1 = stats.poisson.rvs(loc=18, mu=35, size=30)
minnesota_ages2 = stats.poisson.rvs(loc=18, mu=10, size=20)
minnesota_ages = np.concatenate((minnesota_ages1, minnesota_ages2))

# Print the mean age of the population and the sample
print("Mean age of the population:", population_ages.mean())
print("Mean age of the sample:", minnesota_ages.mean())

```

[22] ✓ 0.1s Python

```
Mean age of the population: 50.48598
Mean age of the sample: 42.92
```

```python
# Calculate the t-statistic and p-value for the one-sample t-test
t_stat, p_val = stats.ttest_1samp(a=minnesota_ages, popmean=population_ages.mean())

print("The t-statistic is {} and the p-value is {}.".format(t_stat, p_val))
```

[23] ✓ 0.0s Python

```
The t-statistic is -4.155007645348416 and the p-value is 0.00012990129848405887.
```

```
# Check the quantiles for a 95% confidence level with 49 degrees of freedom
t_quantile_low = stats.t.ppf(q=0.025, df=49)
t_quantile_high = stats.t.ppf(q=0.975, df=49)
print("The t-quantiles for a 95% confidence level with 49 degrees of freedom are {} and {}.".format(t_quantile_low, t_quantile_high))
```

[24] ✓ 0.1s Python

```
The t-quantiles for a 95% confidence level with 49 degrees of freedom are -2.0095752344892093 and 2.009575234489209.
```

```
# Calculate the p-value with the t-statistic and 49 degrees of freedom
p_val_calculated = stats.t.cdf(x=t_stat, df=49) * 2
print("The calculated p-value is {}.".format(p_val_calculated))
```

[25] ✓ 0.0s Python

```
The calculated p-value is 0.00012990129848405887.
```

```
# Calculate the standard error of the mean for the sample
sigma = minnesota_ages.std() / math.sqrt(50)
```

[26] ✓ 0.0s Python

```
# Construct a 95% confidence interval for the sample mean
conf_int_95 = stats.t.interval(alpha=0.95, df=49, loc=minnesota_ages.mean(), scale=sigma)
print("The 95% confidence interval for the sample mean is {}.".format(conf_int_95))
```

[27] ✓ 0.0s Python

```
The 95% confidence interval for the sample mean is (39.29748102093582, 46.542518979064184).
C:\Users\mainp\AppData\Local\Temp\ipykernel_14524\4103696102.py:2: DeprecationWarning: Use of keyword argument 'alpha' for method 'interval' is deprecated and wil be removed in SciPy 1.
  conf_int_95 = stats.t.interval(alpha=0.95, df=49, loc=minnesota_ages.mean(), scale=sigma)
```

```
# Construct a 99% confidence interval for the sample mean
conf_int_99 = stats.t.interval(alpha=0.99, df=49, loc=minnesota_ages.mean(), scale=sigma)
print("The 99% confidence interval for the sample mean is {}.".format(conf_int_99))
```

[28] ✓ 0.2s Python

```
The 99% confidence interval for the sample mean is (38.08904034203606, 47.750959657963946).
C:\Users\mainp\AppData\Local\Temp\ipykernel_14524\176364369.py:2: DeprecationWarning: Use of keyword argument 'alpha' for method 'interval' is deprecated and wil be removed in SciPy 1.
  conf_int_99 = stats.t.interval(alpha=0.99, df=49, loc=minnesota_ages.mean(), scale=sigma)
```

Two Sample T-Test

```
# Generate a sample of voter age data for Wisconsin
np.random.seed(10)
wisconsin_ages1 = stats.poisson.rvs(loc=18, mu=30, size=30)
wisconsin_ages2 = stats.poisson.rvs(loc=18, mu=15, size=20)
wisconsin_ages = np.concatenate((wisconsin_ages1, wisconsin_ages2))
```

[29] ✓ 0.1s Python

```
# Print the mean of the Wisconsin sample
print("Wisconsin Sample Mean:", wisconsin_ages.mean())
```

[30] ✓ 0.1s Python

```
Wisconsin Sample Mean: 41.8
```

```
# Perform a two-sample t-test to compare the Wisconsin sample to the Minnesota sample
t_stat, p_val = stats.ttest_ind(a=minnesota_ages, b=wisconsin_ages, equal_var=False)
```

[31] ✓ 0.0s Python

```
# Print the p-value
print("P-Value:", p_val)
```

[32] ✓ 0.0s Python

```
P-Value: 0.6175514813720644
```

```
# Check if the p-value is less than the significance level of 0.05
if p_val < 0.05:
    print("Reject null hypothesis, the means are different.")
else:
    print("Fail to reject null hypothesis, the means are the same.")
```

[33] ✓ 0.0s Python

```
Fail to reject null hypothesis, the means are the same.
```

Paired T-Test

```
# set the random seed for reproducibility
np.random.seed(11)

# generate weight data before and after treatment
before = stats.norm.rvs(scale=30, loc=250, size=100)
after = before + stats.norm.rvs(scale=5, loc=-1.25, size=100)

# create a dataframe to store the weight data
weight_df = pd.DataFrame({"weight_before": before,
                          "weight_after": after,
                          "weight_change": after - before})

# print a summary of the weight data
print(weight_df.describe())
```

[34] ✓ 0.1s Python

```
       weight_before  weight_after  weight_change
count     100.000000    100.000000     100.000000
mean      250.345546    249.115171      -1.230375
std        28.132539     28.422183       4.783696
min       170.400443    165.913930     -11.495286
25%       230.421042    229.148236      -4.046211
50%       250.830805    251.134089      -1.413463
75%       270.637145    268.927258       1.738673
max       314.700233    316.720357       9.759282
```

```
# conduct a paired t-test to check if the means of the samples differ
t_statistic, p_value = stats.ttest_rel(a=before, b=after)
```

[35] ✓ 0.1s Python

```
# print the p-value
print("Paired t-test p-value:", p_value)
```

[36] ✓ 0.1s Python

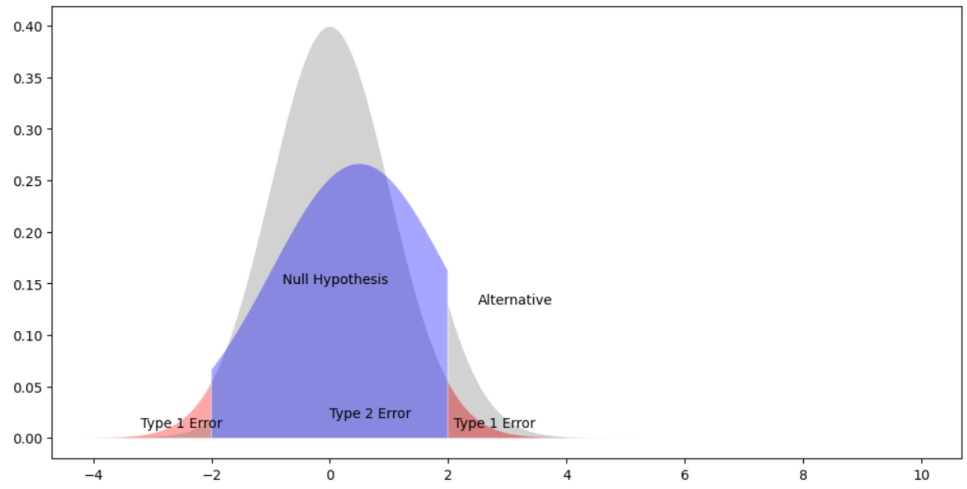
```
Paired t-test p-value: 0.011596444318439859
```

Type I and Type II Errors

```python
# Generate three normal distributions with different means and standard deviations
null_distribution = stats.norm(loc=0, scale=1)
alternative_distribution = stats.norm(loc=0.5, scale=1)
observed_distribution = stats.norm(loc=0.5, scale=1.5)

# Set the plot size
plt.figure(figsize=(12, 10))

# Fill in the areas corresponding to the null distribution
plt.fill_between(x=np.arange(-4, -2, 0.01),
                 y1= null_distribution.pdf(np.arange(-4, -2, 0.01)),
                 facecolor='red', alpha=0.35)
plt.fill_between(x=np.arange(-2, 2, 0.01),
                 y1= null_distribution.pdf(np.arange(-2, 2, 0.01)),
                 facecolor='grey', alpha=0.35)
plt.fill_between(x=np.arange(2, 4, 0.01),
                 y1= null_distribution.pdf(np.arange(2, 4, 0.01)),
                 facecolor='red', alpha=0.5)

# Fill in the areas corresponding to the alternative and observed distributions
plt.fill_between(x=np.arange(-4, -2, 0.01),
                 y1= alternative_distribution.pdf(np.arange(-4, -2, 0.01)),
                 facecolor='grey', alpha=0.35)
plt.fill_between(x=np.arange(-2, 2, 0.01),
                 y1= observed_distribution.pdf(np.arange(-2, 2, 0.01)),
                 facecolor='blue', alpha=0.35)
plt.fill_between(x=np.arange(2, 10, 0.01),
                 y1= alternative_distribution.pdf(np.arange(2, 10, 0.01)),
                 facecolor='grey', alpha=0.35)

# Add labels to the plot
plt.text(x=-0.8, y=0.15, s="Null Hypothesis")
plt.text(x=2.5, y=0.13, s="Alternative")
plt.text(x=2.1, y=0.01, s="Type 1 Error")
plt.text(x=-3.2, y=0.01, s="Type 1 Error")
plt.text(x=0, y=0.02, s="Type 2 Error")
```

[41] ✓ 0.7s Python

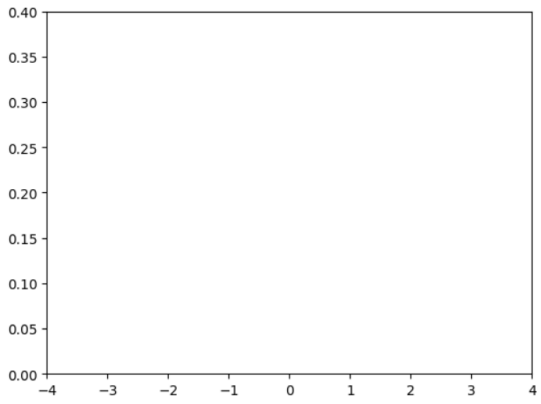
```
Text(0, 0.02, 'Type 2 Error')
```



```python
# Set the x and y limits of the plot
plt.xlim([-4, 4])
plt.ylim([0, 0.4])

# Calculate the Type II error rate for the observed distribution compared to the null distribution
alpha = 0.05  # Significance level
z_alpha = stats.norm.ppf(1 - alpha/2)  # Two-tailed z-score corresponding to alpha
z_beta = (0.5 - 0) / (1.5 / np.sqrt(100)) + z_alpha  # z-score corresponding to beta
beta = 1 - stats.norm.cdf(z_beta, loc=0, scale=1)  # Calculate beta
power = 1 - beta  # Calculate power

# Print the Type II error rate and power
print(f"Type II error rate: {beta:.3f}")
print(f"Power: {power:.3f}")

# Show the plot
plt.show()
```

[38] ✓ 0.3s Python

```
Type II error rate: 0.000
```



EXPERIMENT NO. 13

Student Name and Roll Number: Piyush Gambhir – 21CSU349
Semester /Section: Semester-IV – AIML-B (AL-3)
Link to Code: Piyush-Gambhir/NCU-CSL227-ACS-Lab_Manual (github.com)
Date:
Faculty Signature:
Marks/Grade:

Objective(s): To understand the steps of hypothesis testing for two populations
Outcome: Students will analysis the steps to test hypothesis for two populations
Problem Statement: Write a python program to conduct hypothesis testing for two populations
Background Study: Hypothesis testing allows the statistical evaluation of the claim made
Question Bank: 1. **What is meant by T-test? Explain with example.** A t-test is a statistical test that compares the means of two samples. It is used in hypothesis testing, with a null hypothesis that the difference in group means is zero and an alternate hypothesis that the difference in group means is different from zero. 2. **What is difference between T-test and paired t-test?** Two-sample t-test is used when the data of two samples are statistically independent, while the paired t-test is used when data is in the form of matched pairs. 3. **What is significance of T-Test?** The One Sample t Test examines whether the mean of a population is statistically different from a known or hypothesized value. The One Sample t Test is a parametric test. This test is also known as: Single Sample t Test.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 13

Write a python program to conduct hypothesis testing for two populations

```python
# importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

[5] ✓ 0.0s Python

```python
# Generate two random samples
sample1 = np.random.normal(loc=10, scale=2, size=50)
sample2 = np.random.normal(loc=12, scale=2, size=50)

# Calculate the sample statistics
mean1 = np.mean(sample1)
mean2 = np.mean(sample2)
var1 = np.var(sample1, ddof=1)
var2 = np.var(sample2, ddof=1)
n1 = len(sample1)
n2 = len(sample2)
```

[6] ✓ 0.0s Python

```python
# Calculate the pooled standard deviation
s_p = np.sqrt(((n1 - 1) * var1 + (n2 - 1) * var2) / (n1 + n2 - 2))

# Calculate the test statistic
t = (mean1 - mean2) / (s_p * np.sqrt(1/n1 + 1/n2))

# Calculate the degrees of freedom
df = n1 + n2 - 2

# Calculate the p-value
p_value = stats.t.sf(abs(t), df) * 2

# Set the significance level
alpha = 0.05
```

[7] ✓ 0.1s Python

```python
# Compare the p-value to the significance level
if p_value < alpha:
    print("We reject the null hypothesis that the two populations have equal means.")
else:
    print("We fail to reject the null hypothesis that the two populations have equal means.")
```

[8] ✓ 0.1s Python

```
We reject the null hypothesis that the two populations have equal means.
```