23CY305 - APPLIED STATISTICS USING PYTHON

1. Write a Python program to compute the Mode of a List of Scores

```
import statistics
scores=eval(input("Enter the scores as a list : "))
modes=statistics.multimode(scores)

if len(scores)==len(modes):
    print ("No mode")
elif len(modes)==1:
    print("single mode :",modes[0])
else:
    print("Multimodes :",modes)
```

2. Use the tips dataset to create a boxplot comparing the distribution of total bills across different days of the week. What can you infer about the variability and median total bill on different days?

```
import seaborn as sns
import matplotlib.pyplot as plt
tips=sns.load_dataset("tips")
sns.boxplot(x="day",y="total_bill",data=tips)
plt.show()
```

3. **Create a Histogram and KDE Plot**
   Using the tips dataset from Seaborn, create a histogram and

kernel density estimate (KDE) plot to visualize the distribution of the total bill amount. Interpret what the shape of the distribution tells you about the data.

```
import seaborn as sns

import matplotlib.pyplot as plt

tips=sns.load_dataset("tips")

sns.histplot(tips.total_bill,kde=True)

plt.show()
```

4. Write a Python program to find the GCD of 123456 and 7890 using Euclidean Algorithm.

```
def gcd(a,b):
    while b!=0:
        a,b=b,a%b
    return a

x=int(input("Enter num1 : "))

y=int(input("Enter num1 : "))

print("Gcd of",x,y,"is",gcd(x,y))
```

5. Use your chi-square program to analyze real-world data such as survey responses.

```
import numpy as np
from scipy.stats import chi2_contingency

rows = int(input("Enter number of rows: "))
cols = int(input("Enter number of columns: "))

observed = np.array([[int(input(f"Value at ({i+1},{j+1}): ")) for j in range(cols)] for i in range(rows)])

chi2, p, dof, expected = chi2_contingency(observed)

print(f"\nChi-Square Value: {chi2:.4f}")
print(f"P-value: {p:.4f}")
```

print(f"Degrees of Freedom: {dof}")
print("Expected Frequencies:\n", expected)

print("\nReject H0: Significant association." if p < 0.05 else "\nFail to reject H0: No significant association.")

```
import numpy as np
from scipy.stats import chi2_contingency

rows = int(input("Enter number of rows (e.g., categories like Gender): "))
cols = int(input("Enter number of columns (e.g., choices like Product A/B): "))
observed = []

print("\nEnter the observed survey data:")
for i in range(rows):
    row = [int(input(f"Value at ({i+1},{j+1}): ")) for j in range(cols)]
    observed.append(row)

observed_array = np.array(observed)

chi2, p, dof, expected = chi2_contingency(observed_array)

print("\nChi-Square Value:", round(chi2, 4))
print("P-value:", round(p, 4))
print("Degrees of Freedom:", dof)
print("Expected Frequencies:\n", expected)

if p < 0.05:
    print("Reject H0: There is a significant association between the variables.")
else:
    print("Fail to reject H0: No significant association between the variables.")
```

6. **Scatterplot with Regression Line**
   Generate a scatterplot with a regression line (lmplot) to explore the relationship between total bill and tip amount in the tips dataset. Describe the trend and strength of the relationship based on the plot.

   import seaborn as sns, matplotlib.pyplot as plt

   tips = sns.load_dataset("tips")

   sns.lmplot(x="total_bill", y="tip", data=tips)

   plt.show()

7. Write Python program to Image Brightness Adjustment using NumPy.

```python
import numpy as np
from PIL import Image

def change_brightness(image_path, output_path, brightness_value):
    """
    Adjusts the brightness of an image.

    Args:
        image_path (str): The path to the input image.
        output_path (str): The path to save the adjusted image.
        brightness_value (int): Amount to adjust brightness.
                                Positive values brighten, negative values darken.
    """
    try:
        # Open the image and convert it to a NumPy array
        img = Image.open(image_path)
        arr = np.array(img, dtype=float)

        # Add the brightness value and clip the result to the valid [0, 255] range
        arr = np.clip(arr + brightness_value, 0, 255)

        # Convert the array back to an 8-bit integer type and create an image
        new_img = Image.fromarray(arr.astype(np.uint8))
        new_img.save(output_path)

        print(f"Successfully saved brighter image to {output_path}")

    except FileNotFoundError:
        print(f"Error: The file '{image_path}' was not found.")
    except Exception as e:
        print(f"An error occurred: {e}")

# --- Example Usage ---
# Increase brightness by 60 points
change_brightness('input.jpg', 'brighter_image.jpg', 60)

# Decrease brightness by 60 points
change_brightness('input.jpg', 'darker_image.jpg', -60)
```

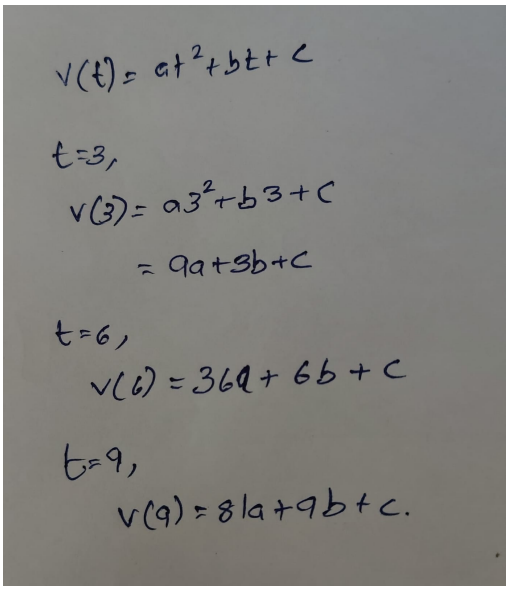
8. Find the count of prime numbers between two given numbers.

```
def is_prime(n):
    if n<=1:
        return False
    for i in range(2,int(n**0.5)+1):
        if n%i==0:
            return False
    return True
a=int(input("Enter starting no: "))
b=int(input("Enter ending no: "))

c=0

for i in range(a,b+1):
    if is_prime(i):
        c+=1
print(f"The number of primes from {a} to {b} is {c}")
```

9. Write a python program to display the names of students (input names separately) with their scores and grades.

```
import statistics

scores=eval(input("Enter the marks : "))

name=input("Enter the names of students seperated by space(the no of marks and names should be equal : ").split()

grades=[]

n=len(scores)

for i in scores:

    if i>=90:

        grades.append("A")

    elif i>=80:

        grades.append("B")

    elif i>=70:

        grades.append("C")

    elif i>=60:

        grades.append("D")

    else:

        grades.append("F")

print("\n Student Details\n")

for i in range(n):

    print(f"Student Name : {name[i]} \n Score : {scores[i]} \n Grade : {grades[i]} \n")
```

10. The upward speed $v(t)$ of a rocket at time t is approximated by $v(t) = at^2 + bt + c$, $0 \leq t \leq 100$ where a, b, and c are constants. It has been found that the speed at times t = 3, t = 6 , and t = 9 seconds are respectively, 64, 133, and 208 miles per second respectively. Find the speed at time t = 15 seconds.




```
import numpy as np
A = np.array([[9, 3, 1], [36, 6, 1], [81, 9, 1]])
B = np.array([64, 133, 208])
X = np.linalg.solve(A, B)
a,b,c=X
a=round(a,2)
b=round(b,2)
c=round(c,2)
print("a = ",round(a,2))
print("b = ",round(b,2))
print("c = ",round(c,2))
t=15
y=a*(t**2)+(b*t)+c
print("Speed of the rocket at time t=15 is ",y,"miles per second")
```

11. Construct a Python program to Image Blurring using a Mean Filter.

```python
import cv2

def blur_image(input_path, output_path, kernel_size=(9, 9)):
    """
    Blurs an image using a mean filter and saves it.

    Args:
        input_path (str): Path to the input image.
        output_path (str): Path to save the blurred image.
        kernel_size (tuple): A tuple of 2 odd integers representing the
                             blurring window size (width, height).
    """
    try:
        # Read the image from the specified path
        image = cv2.imread(input_path)

        # Apply the mean filter using cv2.blur()
        blurred_image = cv2.blur(image, kernel_size)

        # Save the blurred image to the specified path
        cv2.imwrite(output_path, blurred_image)

        print(f"Successfully blurred image and saved it to {output_path}")

    except Exception as e:
        print(f"An error occurred: {e}")

# --- Example Usage ---
# Use the function to blur 'input.jpg' and save it as 'blurred_image.jpg'
blur_image('input.jpg', 'blurred_image.jpg')
```

12. Write a Python program to group the sales data by Salesperson. For each salesperson, calculate their total sales (sum of Total Sales).

```
import pandas as pd

df = pd.read_csv("sales_data.csv")
print(df.groupby("Salesperson")["Total_Amount"].sum())
```

the CSV file needs at least these columns:

- **Salesperson**
- **Total_Amount**

13. Construct a Python program to write a Code for Thresholding using NumPy at 90° clockwise rotation.

```python
import numpy as np
from PIL import Image

def process_image(input_path, output_path, threshold_value=128):
    """
    Rotates an image 90 degrees clockwise and applies a binary threshold.
    """
    try:
        # 1. Open the image and convert it to grayscale ('L' mode)
        img = Image.open(input_path).convert('L')

        # 2. Convert the image to a NumPy array for processing
        arr = np.array(img)

        # 3. Rotate the array 90 degrees clockwise
        # (k=-1 means one rotation in the clockwise direction)
        rotated_arr = np.rot90(arr, k=-1)

        # 4. Apply the threshold
        # Any pixel brighter than threshold_value becomes white (255)
        # All other pixels become black (0)
        threshold_arr = np.where(rotated_arr > threshold_value, 255, 0)

        # 5. Convert the processed array back to an image and save it
        new_img = Image.fromarray(threshold_arr.astype(np.uint8))
        new_img.save(output_path)

        print(f"Successfully processed image and saved to {output_path}")

    except Exception:
        print("Error")

# --- Example Usage ---
# Process 'input.jpg' and save the result as 'output.png'
process_image('input.jpg', 'output.png')
```

14. **Heatmap of Correlation Matrix**

Calculate the correlation matrix for the numerical variables in the iris dataset. Visualize this correlation matrix using a heatmap. Which pairs of variables show the strongest positive and negative correlations?

```
import seaborn as sns, matplotlib.pyplot as plt

iris = sns.load_dataset("iris")



sns.heatmap(iris.corr(numeric_only=True), annot=True,
cmap="coolwarm", center=0)

plt.show()
```

15. Given a dataset weather_data.csv with columns Date, City, Temperature, Humidity, Rainfall, filter the data for a city (e.g., "Delhi"), and plot the monthly average temperature trend over a year using a line chart.

```
import pandas as pd, matplotlib.pyplot as plt



df = pd.read_csv("weather_data.csv", parse_dates=["Date"])

df[df.City=="Delhi"].set_index("Date").Temperature.resample("M").
mean().plot(marker="o", color="red")

plt.show()
```

16. I In a T20 match, Chennai Super Kings needed just 6 runs to win with 1 ball left to go in the last over. The last ball was bowled and the batsman at the crease hit it high up. The ball traversed along a path in a vertical plane and the equation of the path is $y = ax^2 + bx + c$ with respect to a xy - coordinate system in the vertical plane and the ball traversed through the points (10,8), (20,16), (30,18) , can you conclude that Chennai Super Kings won the match?

```
import numpy as np
A = np.array([[100, 10, 1], [400, 20, 1], [900, 30, 1]])
B = np.array([8, 16, 18])
X = np.linalg.solve(A, B)
a, b, c = X
x = -b / (2 * a)
y = a * x**2 + b * x + c
print("Maximum height:", round(y, 2))
if y >= 15:
    print("Conclusion: Chennai Super Kings WON the match.")
else:
    print("Conclusion: Chennai Super Kings did NOT win the match.")
```

17. Write a Python program to load the dataset using Pandas and display the first 5 rows.

```
import pandas as pd

df = pd.read_csv("students.csv")
```

```
# Display first 5 rows
print(df.head(5))
```

18. Write a Python program to count and display how many students received each grade.

```
import statistics

scores=eval(input("Enter the marks : "))

a,b,c,d,e=0,0,0,0,0

for i in scores:

    if i>=90:

        a+=1

    elif i>=80:

        b+=1

    elif i>=70:

        c+=1

    elif i>=60:

        d+=1

    else:

        e+=1

print("\n--- Grade Counts ---")

print(f"A: {a}")

print(f"B: {b}")

print(f"C: {c}")

print(f"D: {d}")
```

```
print(f"F: {e}")
```

19. Write a Python program that takes two numbers as user input and then prints their LCM. Extend the program to find the LCM of three numbers.

```
import math
a=int(input("Enter a number 1 : "))
b=int(input("Enter a number 2 : "))
result=math.lcm(a,b)
print("lcm is ",result)
```

```
import math
a=int(input("Enter a number 1 : "))
b=int(input("Enter a number 2 : "))
c=int(input("Enter a number 3 : "))
result=math.lcm(a,b,c)
print("lcm is ",result)
```

20. Implement a one-sample t-test to check if the sample mean is significantly different from a given value.

```
from scipy.stats import ttest_1samp


# Input sample data

n = int(input("Enter number of elements in sample: "))

data = [float(input(f"Value {i+1}: ")) for i in range(n)]


# Input hypothesized mean

mu = float(input("\nEnter the hypothesized mean: "))
```

```
# One-sample t-test

t, p = ttest_1samp(data, mu)


print(f"\nSample mean = {sum(data)/len(data):.2f}")

print(f"T-statistic = {t:.4f}")

print(f"P-value     = {p:.4f}")


if p < 0.05:

    print("Reject H0: The sample mean is significantly different from the hypothesized mean.")

else:

    print("Fail to reject H0: No significant difference from the hypothesized mean.")
```

21. Modify the t-test program to handle unequal sample sizes.

```
from scipy.stats import ttest_ind

n1 = int(input("Enter number of elements in Group 1: "))
g1 = [float(input(f"Group1 value {i+1}: ")) for i in range(n1)]

n2 = int(input("Enter number of elements in Group 2: "))
g2 = [float(input(f"Group2 value {i+1}: ")) for i in range(n2)]

t, p = ttest_ind(g1, g2, equal_var=False)

print(f"\nT-statistic: {t:.4f}\nP-value: {p:.4f}")
print("Reject H0" if p < 0.05 else "Fail to reject H0")
```

22. Write a Python program to group the dataset by the Product column and calculate the total units sold and total sales for each product.

```
import pandas as pd
```

```
df = pd.read_csv("sales_data.csv")
print(df.groupby("Product").agg(Total_Units=("Quantity","sum"),
Total_Sales=("Total_Amount","sum")))
```

```
Product,Quantity,Total_Amount
Laptop,5,300000
Mobile,10,250000
Tablet,4,80000
Laptop,3,180000
Mobile,6,150000
Tablet,2,40000
Headphones,15,75000
Laptop,2,120000
Headphones,10,50000
Mobile,8,200000
```

23.Extend the chi-square program to handle goodness-of-fit test.

```
from scipy.stats import chisquare

n = int(input("Enter number of categories: "))

print("\nEnter observed frequencies:")
observed = [int(input(f"Observed value for category {i+1}: ")) for i in range(n)]

print("\nEnter expected frequencies:")
expected = [float(input(f"Expected value for category {i+1}: ")) for i in range(n)]

chi2_stat, p_value = chisquare(f_obs=observed, f_exp=expected)

print("\nChi-Square Value:", round(chi2_stat, 4))
```

```
print("P-value:", round(p_value, 4))


if p_value < 0.05:
    print("Reject H0: Observed distribution is significantly different from expected.")
else:
    print("Fail to reject H0: Observed distribution fits the expected distribution.")
```

24. Solve the following system of equations, $2x_1 + 3x_2 + 3x_3 = 5$, $x_1 – 2x_2 + x_3 = -4$, $3x_1 – x_2 – 2x_3 = 3$

```
import numpy as np
A = np.array([[2, 3, 3], [1, -2, 1], [3, -1, -2]])
B = np.array([5, -4, 3])
X = np.linalg.solve(A, B)
print("Solution:", X)
```

25. Write a Python program to check whether 35 and 64 are coprime.

```
def gcd(a,b):
    while b!=0:
        a,b=b,a%b
    return a
x=int(input("Enter num1 : "))
y=int(input("Enter num1 : "))
if gcd(x,y)==1:
    print(f"{x} and {y} are coprime")
```

```
else:
    print(f"{x} and {y} are not coprimes")
```

26. A retail chain wants to classify customers into lifestyle groups using three features: **Age**, **Annual Income**, and **Spending Score**.

The following dataset contains information of 10 customers:

CustomerID	Age	AnnualIncome	SpendingScore
1	19	15000	39
2	21	18000	81
3	20	17000	6
4	23	22000	77
5	31	35000	40
6	35	40000	50
7	40	45000	60
8	52	60000	30
9	58	65000	20
10	63	70000	70

Question:

1. Load the above dataset into a pandas DataFrame (either manually or using CSV).
2. Standardize the features since they are in different ranges.
3. Use the **Elbow Method** to find the best number of clusters.
4. Perform **K-Means clustering**.
5. Visualize the clusters in a **3D scatter plot** (Age vs Income vs Spending Score).
6. Explain the characteristics of the clusters (e.g., young-high income-high spending, old-low income-low spending, etc.).

```
import pandas as pd, matplotlib.pyplot as plt
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D

df = pd.read_csv("customers.csv")

X =
StandardScaler().fit_transform(df[["Age","AnnualIncome","Spendin
gScore"]])

plt.plot(range(1,7), [KMeans(n_clusters=k, n_init=10,
random_state=0).fit(X).inertia_ for k in range(1,7)], "o-")
plt.show()

kmeans = KMeans(n_clusters=3, n_init=10, random_state=0)
df["Cluster"] = kmeans.fit_predict(X)

ax = plt.figure().add_subplot(111, projection="3d")
ax.scatter(df.Age, df.AnnualIncome, df.SpendingScore,
c=df.Cluster, cmap="viridis", s=80)
plt.show()

print(df.groupby("Cluster")[["Age","AnnualIncome","SpendingScore
"]].mean())
```

27. Write a program for thresholding using NumPy.

```python
import numpy as np
from PIL import Image

def apply_threshold(input_path, output_path, threshold_value=128):
    """
    Applies a binary threshold to an image and saves it.
    """
    try:
        # 1. Open the image and convert it to grayscale ('L' mode)
        img = Image.open(input_path).convert('L')

        # 2. Convert the image to a NumPy array for processing
        arr = np.array(img)

        # 3. Apply the threshold.
        # If a pixel is brighter than threshold_value, make it white (255).
        # Otherwise, make it black (0).
        threshold_arr = np.where(arr > threshold_value, 255, 0)

        # 4. Convert the new array back to an image and save it
        new_img = Image.fromarray(threshold_arr.astype(np.uint8))
        new_img.save(output_path)

        print(f"Successfully applied threshold and saved to {output_path}")

    except FileNotFoundError:
        print(f"Error: The file '{input_path}' was not found.")
    except Exception as e:
        print(f"An error occurred: {e}")

# --- Example Usage ---
# Process 'input.jpg' and save the result as 'threshold_image.png'
apply_threshold('input.jpg', 'threshold_image.png')
```

28. A shopping mall collects customer information containing **Age** and **Annual Income**. The management wants to identify different customer groups for targeted marketing campaigns.

The following dataset contains information of 10 customers:

CustomerID	Age	AnnualIncome
1	19	15000
2	21	18000
3	20	17000
4	23	22000
5	31	35000
6	35	40000

CustomerID	Age	AnnualIncome
7	40	45000
8	52	60000
9	58	65000
10	63	70000

Question:

I. Load the above dataset into a pandas DataFrame (you can enter manually or save as CSV and load).
II. Apply the **Elbow Method** to determine the optimal number of clusters.
III. Perform **K-Means clustering** to group customers.
IV. Visualize the clusters in a scatter plot (Age vs Annual Income).
V. Interpret the results (e.g., young-low income, middle-aged-high income, etc.).

```
import pandas as pd, matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler



df = pd.read_csv("customers.csv")



X = StandardScaler().fit_transform(df[["Age","AnnualIncome"]])

plt.plot(range(1,7), [KMeans(n_clusters=k, n_init=10, random_state=0).fit(X).inertia_ for k
in range(1,7)], "o-")

plt.show()



df["Cluster"] = KMeans(n_clusters=3, n_init=10, random_state=0).fit_predict(X)



plt.scatter(df.Age, df.AnnualIncome, c=df.Cluster, cmap="viridis", s=80)

plt.show()
```

print(df.groupby("Cluster")[["Age","AnnualIncome"]].mean())

29.Write a program for Image Inversion using NumPy

```
import numpy as np
from PIL import Image

def invert_image(input_path, output_path):
    """
    Inverts the colors of an image and saves it.
    """
    try:
        # 1. Open the image and convert it to a NumPy array
        img = Image.open(input_path)
        arr = np.array(img)

        # 2. Invert the image by subtracting each pixel value from 255
        inverted_arr = 255 - arr

        # 3. Convert the inverted array back to an image and save it
        inverted_img = Image.fromarray(inverted_arr.astype(np.uint8))
        inverted_img.save(output_path)

        print(f"Successfully inverted image and saved it to {output_path}")

    except FileNotFoundError:
        print(f"Error: The file '{input_path}' was not found.")
    except Exception as e:
        print(f"An error occurred: {e}")

# --- Example Usage ---
# Invert 'input.jpg' and save it as 'inverted_image.png'
invert_image('input.jpg', 'inverted_image.png')
```

30.Solve the following system of linear equations, $4x + 3y + 6z = 25$, $x + 5y + 7z = 13$, $2x + 9y + z = 1$.

```
import numpy as np
A = np.array([[4, 3, 6], [1, 5, 7], [2, 9, 1]])
B = np.array([25, 13, 1])
X = np.linalg.solve(A, B)
print("Solution:", X)
```

31. Load a CSV file containing student data and apply logistic regression.

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
```

```
# Load data
df = pd.read_csv("student.csv")


# Train model
X = df[["StudyHours","PreviousMarks"]]
y = df["Passed"]
model = LogisticRegression().fit(X, y)


# Predict
print("Prediction:",model.predict([[6,58]]))
```

32. Using Pandas, filter the dataset to show only the sales records where the Region is **"South"** and the Product is **"Laptop"**.

```
import pandas as pd
df = pd.read_csv('sales_data.csv')
filtered_df = df[(df['Region'] == 'South') & (df['Product'] == 'Laptop')]
print("Filtered Records (Region = 'South' AND Product = 'Laptop'):")
print(filtered_df)
```