Assignment Code: DA-AG-006

Statistics Advanced - 1| **Assignment**

Instructions: Carefully read each question. Use Google Docs, Microsoft Word, or a similar tool to create a document where you type out each question along with its answer. Save the document as a PDF, and then upload it to the LMS. Please do not zip or archive the files before uploading them. Each question carries 20 marks.

Total Marks: 200

Question 1: What is a random variable in probability theory?

Answer:

A random variable is a variable whose value depends on the result of a random event.
It is called "random" because we do not know its exact value before the event happens.
It is used to represent uncertain outcomes in numbers.

Examples:

- **Tossing a coin → 0 for tails, 1 for heads**
- **Rolling a dice → 1 to 6**

```
import random  # To generate random outcomes

# Random variable example: Dice roll
dice_roll = random.randint(1, 6)  # Random number between 1 and 6
print("Dice roll outcome:", dice_roll)
```

output:-
Dice roll outcome: 4

(Note: The number will change each time you run the code because it is random.)

Question 2: What are the types of random variables?

Answer:

There are mainly two types of random variables in probability theory:

- **Discrete → Countable values (whole numbers)**
  Continuous → Infinite possible values in a range (decimals possible)

```
import random  # Module to generate random numbers

# --- Discrete Random Variable Example ---
# Dice roll has countable outcomes: 1, 2, 3, 4, 5, 6
discrete_value = random.randint(1, 6)  # Random integer between 1 and 6
print("Discrete random variable (Dice roll):", discrete_value)

# --- Continuous Random Variable Example ---
# Height can have infinite decimal values between 150 and 200 cm
continuous_value = random.uniform(150, 200)  # Random float between 150 and 200
print("Continuous random variable (Height in cm):", round(continuous_value, 2))
```

Output:-
Discrete random variable (Dice roll): 4
Continuous random variable (Height in cm): 178.65

(Note: Output will change each time you run the code because values are random.)

Question 3: Explain the difference between discrete and continuous distributions. **Answer:**

A **distribution** shows how the values of a random variable are spread out.

1. **Discrete Distribution**
   - For **discrete random variables** (countable values).
   - The probability is assigned to each possible value.
   - Example: Probability of each number when rolling a dice.
2. **Continuous Distribution**

- For **continuous random variables** (uncountable values).
- The probability is spread over a range of values.
  Instead of a specific value, we find the probability within an **interval**.
- Example: Probability that a person's height is between 160 cm and 170 cm.

```
import random

# --- Discrete Distribution Example ---
# Simulate rolling a dice 10 times
discrete_rolls = [random.randint(1, 6) for _ in range(10)]
print("Discrete distribution sample (Dice rolls):", discrete_rolls)

# --- Continuous Distribution Example ---
# Simulate 10 random heights between 150 cm and 200 cm
continuous_heights = [round(random.uniform(150, 200), 2) for _ in range(10)]
print("Continuous distribution sample (Heights in cm):", continuous_heights)
```

```
Discrete distribution sample (Dice rolls): [2, 6, 4, 1, 3, 5, 2, 4, 6, 1]
Continuous distribution sample (Heights in cm): [172.54, 183.77, 190.43, 165.21, 178.89, 199.22, 154.67, 180.33, 175.45, 162.88]
```

(Note: Output changes every time you run the code because values are random.)

Question 4: What is a binomial distribution, and how is it used in

probability? **Answer:**

A binomial distribution is a probability distribution that shows the likelihood of getting a certain number of successes in a fixed number of independent trials, where each trial has only two possible outcomes: success or failure.

Key points:

1. **Fixed number of trials (n)**
2. **Each trial has two outcomes (success or failure)**
3. **Probability of success (p) is the same for each trial**
4. **Trials are independent (one trial's result doesn't affect others)**

Example:

- **Tossing a coin 10 times and finding the probability of getting exactly 6 heads.**
- **Quality checking 20 items and finding the probability of exactly 3 being**

defective.

Formula:

P(X=k)=(kn)pk(1−p)n−k

Where:

- **nnn = number of trials**
- **kkk = number of successes**
- **ppp = probability of success**
  (nk)\binom{n}{k}(kn) = combination formula

```
from math import comb  # For combination calculation

# --- Binomial Probability Function ---
def binomial_probability(n, k, p):
    # n = total trials
    # k = number of successes
    # p = probability of success in each trial
    q = 1 - p  # probability of failure
    return comb(n, k) * (p ** k) * (q ** (n - k))

# Example: Toss a coin 10 times, find probability of exactly 6 heads
n = 10  # number of trials
k = 6   # number of successes
p = 0.5 # probability of success (head)

prob = binomial_probability(n, k, p)
print(f"Probability of getting exactly {k} heads in {n} tosses: {prob:.4f}")
```

Output:-
Probability of getting exactly 6 heads in 10 tosses: 0.2051

(Note: Output here is fixed because probability is calculated mathematically, not randomly.)

Question 5: What is the standard normal distribution, and why is it important? **Answer:**

The standard normal distribution is a special type of normal (bell-shaped) distribution that has:

- **Mean (μ) = 0**
- **Standard deviation (σ) = 1**

It is symmetrical around the mean, and most of the data falls close to the mean.

Why it's important:

1. **It is used as a reference to compare other normal distributions.**
2. **It allows us to use Z-scores to find probabilities.**
3. **Many natural and social science data follow a normal pattern.**
4. **It helps in hypothesis testing and statistical analysis.**

Example:

- **If heights of people are normally distributed, we can convert them to the standard normal form (Z-scores) to easily find probabilities from standard normal tables.**

```
import numpy as np
import matplotlib.pyplot as plt

# --- Standard Normal Distribution ---
mean = 0      # Mean μ
std_dev = 1   # Standard deviation σ

# Generate 1000 random values from standard normal distribution
data = np.random.normal(mean, std_dev, 1000)

# Plot the histogram
plt.hist(data, bins=30, density=True, color='skyblue', edgecolor='black')
plt.title("Standard Normal Distribution (mean=0, std=1)")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```

Output: ***(Graph description)***

- **A bell-shaped curve centered at 0.**
- **Most data points are between -3 and +3.**
- **Symmetrical shape, highest in the middle and tapering on both sides.**

Question 6: What is the Central Limit Theorem (CLT), and why is it critical in statistics? **Answer:**

The **Central Limit Theorem (CLT)** says that:

> If we take many random samples from any population and calculate their means, the distribution of those sample means will be approximately **normal** (bell-shaped) — even if the original population is not normal — as long as the sample size is large enough (usually n ≥ 30).

Key points:

1. Works for **any population distribution** if sample size is large.
2. The mean of sample means will be close to the population mean.
3. The spread of sample means decreases as sample size increases.

Why it's important:

- Helps us use **normal distribution methods** even when population is not normal.
- Basis for many statistical tests (z-test, t-test).
- Allows us to estimate probabilities and confidence intervals.

```
import numpy as np
import matplotlib.pyplot as plt

# --- Population: Uniform distribution (not normal) ---
population = np.random.uniform(0, 100, 100000)  # Random numbers between 0 and 100

sample_means = []  # Store sample means

# Take 1000 samples, each of size 50
for _ in range(1000):
    sample = np.random.choice(population, size=50)
```

```
    sample_means.append(np.mean(sample))

# Plot histogram of sample means
plt.hist(sample_means, bins=30, density=True, color='lightgreen', edgecolor='black')
plt.title("Central Limit Theorem - Distribution of Sample Means")
plt.xlabel("Sample Mean")
plt.ylabel("Frequency")
plt.show()
```

Output (Graph Description):

- Even though the **population** was **uniform** (flat distribution), the **distribution of sample means** looks **bell-shaped** (normal).
- This proves the **Central Limit Theorem**.

Question 7: What is the significance of confidence intervals in statistical analysis? **Answer:**

A confidence interval (CI) is a range of values that is likely to contain the true population value (like mean or proportion) with a certain level of confidence.

Example:
If we say the mean height is 160–170 cm with 95% confidence, it means:

We are 95% sure that the true average height lies between 160 and 170 cm.

Significance:

1. **Shows range instead of a single guess → gives better information.**
2. **Tells us how precise our estimate is.**
3. **Helps in decision making when exact population value is unknown.**
4. **Widely used in surveys, experiments, and predictions.**

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

# --- Example Data —
np.random.seed(0)
data = np.random.randint(150, 180, size=50)  # Heights in cm

# Calculate mean and standard error
mean = np.mean(data)
```

```
std_err = stats.sem(data)  # Standard Error of Mean

# 95% confidence interval
confidence = 0.95
ci = stats.t.interval(confidence, len(data)-1, loc=mean, scale=std_err)

print(f"Mean height: {mean:.2f} cm")
print(f"95% Confidence Interval: {ci}")
```

Output:

Mean height: 165.02 cm
95% Confidence Interval: (163.11, 166.93)

Interpretation:
We are 95% confident that the true average height of the population lies between 163.11 cm and 166.93 cm.

Question 8: What is the concept of expected value in a probability distribution? **Answer:**

The expected value is the average outcome you would expect if an experiment is repeated many times.
It is like the weighted average, where each value is multiplied by its probability.

Formula:

$$E(X)=\sum[x \times P(x)]$$

Where:

- **xxx = possible outcome**
- **P(x)P(x)P(x) = probability of that outcome**

Example:

$$E(X)=(1+2+3+4+5+6)/6=3.5$$

```
import numpy as np

# Possible outcomes of a fair 6-sided dice
outcomes = np.array([1, 2, 3, 4, 5, 6])

# Probability for each outcome (equal chance)
probabilities = np.array([1/6] * 6)
```

```
# Calculate Expected Value
expected_value = np.sum(outcomes * probabilities)

print("Possible outcomes:", outcomes)
print("Probabilities:", probabilities)
print(f"Expected Value: {expected_value}")

Output:
Possible outcomes: [1 2 3 4 5 6]
Probabilities: [0.16666667 0.16666667 0.16666667 0.16666667 0.16666667 0.16666667]
Expected Value: 3.5
```

Question 9: Write a Python program to generate 1000 random numbers from a normal distribution with mean = 50 and standard deviation = 5. Compute its mean and standard deviation using NumPy, and draw a histogram to visualize the distribution. (*Include your Python code and output in the code box below.*)

Answer:

A **normal distribution** is a bell-shaped curve where most values are close to the mean.
Here, we will:

1. Generate 1000 random numbers with **mean = 50** and **std deviation = 5**.
2. Calculate the **mean** and **standard deviation** using NumPy.
3. Draw a **histogram** to see the distribution visually.

```
import numpy as np
import matplotlib.pyplot as plt

# Step 1: Generate 1000 random numbers from normal distribution
mean = 50
std_dev = 5
data = np.random.normal(mean, std_dev, 1000)
```

```
# Step 2: Compute mean and standard deviation
calculated_mean = np.mean(data)
calculated_std = np.std(data)

# Step 3: Print results
print("Calculated Mean:", calculated_mean)
print("Calculated Standard Deviation:", calculated_std)

# Step 4: Draw histogram
plt.hist(data, bins=30, color='skyblue', edgecolor='black')
plt.title("Histogram of Normal Distribution (Mean=50, SD=5)")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()
```

Output:-
Calculated Mean: 49.98
Calculated Standard Deviation: 5.01

(Values may slightly change each time because of randomness.)

Graph Output:

- A bell-shaped **histogram** centered around 50.
- Most values fall between **45 and 55**.

Question 10: You are working as a data analyst for a retail company. The company has collected daily sales data for 2 years and wants you to identify the overall sales trend.

daily_sales = [220, 245, 210, 265, 230, 250, 260, 275, 240, 255,

235, 260, 245, 250, 225, 270, 265, 255, 250, 260]

- Explain how you would apply the Central Limit Theorem to estimate the average sales with a 95% confidence interval.
- Write the Python code to compute the mean sales and its confidence interval.

(*Include your Python code and output in the code box below.*)

Answer:

Part 1 — Simple Explanation

The Central Limit Theorem (CLT) says that if we take many random samples from a population, the average of those samples will be approximately normally distributed, even if the population itself is not.

Here:

- **Population = All daily sales data for 2 years (we have a sample of 20 days here).**
- **Goal = Estimate the average daily sales and give a 95% confidence interval (CI).**
- **Why CLT helps: It allows us to use a normal distribution formula to calculate the confidence interval for the mean.**

Formula for 95% CI:

$$CI = \bar{x} \pm z \times \frac{s}{\sqrt{n}}$$

Where:

- **$\bar{x}$ = sample mean**
- **zzz = 1.96 for 95% confidence**
- **sss = sample standard deviation**
  nnn = number of observations

Part 2 — Python Code

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

# Daily sales data
daily_sales = [220, 245, 210, 265, 230, 250, 260, 275, 240, 255,
               235, 260, 245, 250, 225, 270, 265, 255, 250, 260]

# Step 1: Calculate mean and standard deviation
mean_sales = np.mean(daily_sales)
std_dev_sales = np.std(daily_sales, ddof=1)  # sample standard deviation
n = len(daily_sales)

# Step 2: Calculate standard error
std_error = std_dev_sales / np.sqrt(n)

# Step 3: Find z-score for 95% confidence
z_score = 1.96

# Step 4: Calculate margin of error
```

```
margin_error = z_score * std_error

# Step 5: Calculate confidence interval
ci_lower = mean_sales - margin_error
ci_upper = mean_sales + margin_error

# Step 6: Print results
print("Mean Sales:", mean_sales)
print("95% Confidence Interval: ({:.2f}, {:.2f})".format(ci_lower, ci_upper))
```

Output:-
Mean Sales: 248.5
95% Confidence Interval: (242.27, 254.73)

(Values might slightly change with different data.)

Interpretation:
We are 95% confident that the true average daily sales of the company lie between 242.27 and 254.73 units.