Short Quiz: Independence, Conditional Probability & Bayes' Theorem

Arin Chauhan

S24CSEU1777

Total Questions: 6

Total Marks: 15

Time: 20 minutes

Instructions: Answer all questions. Use Python where required.

Students must write the explanation of each question (hand written), and it should be uploaded to the LMS.

Section A: Multiple Choice Questions (1 mark each)

Q1. Two events A and B are independent. If P(A) = 0.4 and P(B) = 0.5, then P(A ∩ B) is:

a) 0.2

b) 0.4

c) 0.9

d) 0.1

Answer: a) $P(A \cap B)=P(A)\times P(B)=0.4\times 0.5=0.2$

Q2. If P(A ∩ B) = 0.1 and P(B) = 0.5, then P(A|B) = ?

a) 0.1

b) 0.5

c) 0.2

d) 0.3

Answer: c) $P(A|B) = P(A \cap B)/ P(B)= 0.1/0.5=0.2$

Section B: Short Answer Questions (3 marks each)

Q3. Define event independence in probability. Provide one real-world example related to computing or technology.

Answer: In probability , two events are said to be independent if the occurrence of one event does not affect the probability of the occurrence of the other event. Formally, events A and B are independent if and only if:

$P(A \cap B) = P(A) \times P(B)$

This means the probability that both events happen together is the product of their individual probabilities.

Example:

Event A: Your phone receives an email notification.

Event B: Your laptop's battery gets fully charged.

These two events are unrelated, whether your phone gets an email has nothing to do with your laptop's charging status.

Q4. In a certain university, 70% of students are from city A and 30% from city B. 80% of city A students and 60% of city B students own laptops. If a randomly selected student owns a laptop, what is the probability they are from city A?

(Use Bayes' Theorem. Show steps.)

Answer:

$P(L) = (0.8 * 0.7) + (0.6 * 0.3)$

$P(A|L) = (0.8 * 0.7) / 0.74$

$P(L) = 0.74$

$P(A|L) = 0.56 / 0.74 = 0.7568$

Section C: Python Coding (5 marks)

Q5. Write a Python function that calculates conditional probability:

$P(A|B)=P(A \cap B)/P(B)$

Your function should take p_a_and_b and p_b as inputs and return the result.

Answer:

```
def conditional_probability(p_a_and_b, p_b):
    if p_b == 0:
        raise ValueError("P(B) cannot be zero")
   return p_a_and_b / p_b
#Example:
p_a_and_b = 0.15
p_b = 0.6
```

result = conditional_probability(p_a_and_b, p_b)

print(f"P(A|B) = {result}")

#Assignment 2:

Numpy Library:-

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Questions:

1) Generate 1500 random integers between (90, 100). Print the frequency in dictionary as {number:frequency} of all numbers i.e., 90 to 100 while selecting the seed as

(a) when seed value is 2

(b) when seed value is 4

(c) when seed value is 6

Answer:

```
import numpy as np

def freq_random_integers(seed):
    np.random.seed(seed)
    arr = np.random.randint(90, 101, size=1500)
    freq = {i: np.count_nonzero(arr == i) for i in range(90, 101)}
    print(f"Seed {seed}:", freq)


freq_random_integers(2)

freq_random_integers(4)

freq_random_integers(6)
```

2) Create a function that computes the probability of some number K in a data array,

(a) if array is {20,20,21,21,21,22,23,26,27,27} then compute the probability of 20, 21, 24, 26 and 27.

(b) If array is {a, a, b, b, c, x, y, z, z, z}, then compute the probability of a, d, y, and z.

Answer:

```
def probability(arr, numbers):
```

```
    total = len(arr)
    return {num: arr.count(num) / total for num in numbers}

arr1 = [20, 20, 21, 21, 21, 22, 23, 26, 27, 27]
print(probability(arr1, [20, 21, 24, 26, 27]))

arr2 = ["a", "a", "b", "b", "c", "x", "y", "z", "z", "z"]
print(probability(arr2, ["a", "d", "y", "z"]))
```

3) In a list mol = ["I", "am", "writing", "a", "program"], Randomly sample 1500 words and print the frequency of each word in dictionary form as:{'I': frequency, 'am':frequency, 'writing':frequency,

'a':frequency, 'program':frequency} when

(a) seed is 30

(b) seed is 40

Answer:

```
import random
mol = ["I", "am", "writing", "a", "program"]
def freq_sample(seed):
random.seed(seed)
        samples = [random.choice(mol) for _ in range(1500)]
        freq = {word: samples.count(word) for word in mol}
        print(f"Seed {seed}:", freq)
freq_sample(30)
freq_sample(40)
```

Expert level:

4) Generate a set of 20 random numbers between 10 to 200 and store them in a CSV file. For that, you have to create a ‘rand1.csv’ file. Generate another set of 20 random numbers between 10 to 200, and store them in another file called ‘rand2.csv’.

Now, plot the frequency distribution for both the files ‘rand1.csv’ and ‘rand2.csv’ in a single graph with different colours.

Answer:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

np.random.seed(1)

rand1 = np.random.randint(10, 201, size=20)

pd.DataFrame(rand1).to_csv("rand1.csv", index=False, header=False)

np.random.seed(2)

rand2 = np.random.randint(10, 201, size=20)

pd.DataFrame(rand2).to_csv("rand2.csv", index=False, header=False)

data1 = pd.read_csv("rand1.csv", header=None)[0]

data2 = pd.read_csv("rand2.csv", header=None)[0]

freq1 = data1.value_counts().sort_index()

freq2 = data2.value_counts().sort_index()

plt.bar(freq1.index, freq1.values, width=2, color='blue', alpha=0.7, label='rand1.csv')

plt.bar(freq2.index, freq2.values, width=2, color='red', alpha=0.5, label='rand2.csv')

plt.xlabel("Number")

plt.ylabel("Frequency")

plt.title("Frequency Distribution of Random Numbers")

plt.legend()

plt.show()
```

Assignment 3:

NumPy and Data Analysis Advanced Lab Tasks:

These problems are designed to require critical thinking and deeper application of NumPy, probability, and data handling concepts.

1) Generate a 2D NumPy array of shape (100, 50) with random integers between 1 and 500. For each column, compute the mean, median, and standard deviation, and store the results in a dictionary

with column index as the key. Finally, identify the column that has the highest standard deviation and the one with the lowest mean.

Answer:

```
import numpy as np

arr = np.random.randint(1, 501, size=(100, 50))

stats = {}

for i in range(arr.shape[1]):
    col = arr[:, i]
    stats[i] = {
        'mean': np.mean(col),
        'median': np.median(col),
        'std': np.std(col)
    }
highest_std_col = max(stats, key=lambda x: stats[x]['std'])

lowest_mean_col = min(stats, key=lambda x: stats[x]['mean'])

print("Highest Std Dev Column:", highest_std_col)

print("Lowest Mean Column:", lowest_mean_col)
```

2) Simulate rolling two dice 5000 times. Store the sum of the two dice in an array. Calculate the experimental probability distribution of the sums and compare it with the theoretical probability distribution for two dice. Plot both distributions on the same graph for comparison.

Answer:

```
import numpy as np

import matplotlib.pyplot as plt


rolls = np.random.randint(1, 7, size=(5000, 2))

sums = np.sum(rolls, axis=1)

exp_prob = {i: np.count_nonzero(sums == i) / 5000 for i in range(2, 13)}

theo_prob = {i: (6 - abs(i - 7)) / 36 for i in range(2, 13)}
```

```
plt.bar(exp_prob.keys(), exp_prob.values(), alpha=0.6, label='Experimental')

plt.plot(list(theo_prob.keys()), list(theo_prob.values()), 'ro-', label='Theoretical')

plt.xlabel("Sum of Dice")

plt.ylabel("Probability")

plt.legend()

plt.show()
```

3) Given a dataset of student scores in three subjects (Math, Science, English), generate a 2D array of shape (30, 3) with scores between 0 and 100. Normalize the scores for each student (row-wise) so that their total is 1. Then, identify the top 5 students who have the highest normalized Math scores.

Answer:

```
import numpy as np

scores = np.random.randint(0, 101, size=(30, 3))

norm_scores = scores / scores.sum(axis=1, keepdims=True)

top5 = np.argsort(norm_scores[:, 0])[-5:][::-1]

print("Top 5 Students (by normalized Math scores):", top5)
```

4) Create a 10x10 NumPy array filled with random integers between 50 and 500. Replace all elements divisible by both 5 and 7 with -1. Count how many replacements were made and find the average of the remaining numbers.

Answer:

```
import numpy as np

arr = np.random.randint(50, 501, size=(10, 10))

mask = (arr % 5 == 0) & (arr % 7 == 0)

count = np.count_nonzero(mask)

arr[mask] = -1

avg_remaining = np.mean(arr[arr != -1])

print("Replacements made:", count)

print("Average of remaining:", avg_remaining)
```

5) Generate two CSV files, each containing 100 random integers between 1 and 1000. Merge the two files into a single dataset, remove duplicates, and then sort the numbers in descending order. Finally, plot a histogram of the merged dataset, marking the mean and median values on the plot.

Answer:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data1 = np.random.randint(1, 1001, 100)
data2 = np.random.randint(1, 1001, 100)
pd.DataFrame(data1).to_csv("file1.csv", index=False, header=False)
pd.DataFrame(data2).to_csv("file2.csv", index=False, header=False)
merged = pd.concat([pd.read_csv("file1.csv", header=None), pd.read_csv("file2.csv", header=None)])
merged = merged[0].drop_duplicates().sort_values(ascending=False)
mean_val = merged.mean()
median_val = merged.median()
plt.hist(merged, bins=20, alpha=0.7, color='skyblue', edgecolor='black')
plt.axvline(mean_val, color='red', linestyle='--', label=f'Mean: {mean_val:.2f}')
plt.axvline(median_val, color='green', linestyle='--', label=f'Median: {median_val:.2f}')
plt.legend()
plt.show()
```