Awarding Body: Arden University
Programme Name: Master of Data Science
Module Name (and Part if applicable): Mathematics for Data Science
Assessment Title: Study on Conditional Probability
Student Number: STU221742
Tutor Name: Mr. Ali Vaisifard
Word Count: 1320 Please refer to the Word Count Policy on your Module Page for guidance

Contents:

Introduction

"Probability is a mathematical tool for measuring the probability of an event occurring in a given sample space" (Feller, 1968,p. 2). In this examination, the effect of weight on the probability result has been checked with Python codes. Based on this research, we examine the influence of the age index on the purchase probability. We consider two possibilities. First, The probability of a weight in which the probability of purchase by the elderly increases. The second is unweighted probability where everyone has the same probability of buying. Analysis of purchasing behavior based on age factors helps understand the effective approach in marketing decisions and sales strategies and the difference between normal and dependent distribution.

Step1: Import Libraries

```
import numpy as np           # A functional library for working on arrays
import pandas as pd
import matplotlib.pyplot as plt
```

Figure1- Python code for calling libraries

Pandas:" provides high-level data structures and functions designed to make working with structured or tabular data fast, easy, and expressive. Since its emergence in 2010, it has helped make Python a powerful and productive data analysis environment" (McKinney,2017, p. 4).

numpy: for performing mathematical operations and numerical array operations

Matplotlib: One of the most useful and practical libraries for drawing graphs and visual data analysis, and in machine learning.

Step2: Setting the seed point

"The seed is the initial input to a pseudorandom number generator that determines the sequence of numbers generated. Different seeds produce different sequences, while the same seed will always produce the sequence." (Knuth, 1997, p. 10). This function allows for the generation of the same random values in several executions, which is essential in machine learning. Argument 42 is chosen as the first seeding value and there is no scientific reason for choosing it.

```
np.random.seed(42)              # A fixed value for the seed function to generate the same selection of random numbers each time it is run
```

Figure 2- Python code for setting the seed

Data distribution:

In the real world, we use non-uniform distribution to calculate weighted and unweighted probabilistic statistics. Because the age distribution of the data as well as the probability of buying for each age is different compared to the entire population. In this project, we calculate probabilities based on a hypothetical model. Therefore, the data has a uniform distribution for both checks and the pre-determined purchase probability is weighted for checking the hypothetical probability.

Step3: Random data generation

```
#Generating 30000 Random Data
```

```
np.random.seed(42)                # A fixed value for the seed function to generate the same selection of random numbers each time it is run

n_samples = 30000                 # Population

ages = [25, 30, 35, 40, 45, 50]     # an array of ages which we want to survey their possibilities

age_data = np.random.choice(ages, size=n_samples) # Selection of 30,000 samples at random as uniformly distributed, each value of which
                                                  #is selected from the age array.

age_counts = {age: np.sum(age_data == age) for age in ages} #Counts the number of times each age appears in random data


plt.figure(figsize=(8, 6))     # adjustting the chart frame
plt.bar(age_counts.keys(), age_counts.values(), color='green', edgecolor='black', width=3)# Drawing the bar chart
plt.title('Age Distribution (Uniform)')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.xticks(ages)  # Inserting a label on the axle x

plt.tight_layout() # Adjustting the bar's distance
plt.show()
```

Figure 3- Python code for creating Population

First, we introduce the total population and samples. Then more than 30,000 random data from the interval (25, 30, 35, 40, 45, 50) are generated as a population. Since our distribution is the same in this analysis, approximately 5000 numbers are generated from each data set. Then count the quantity of each data. We use a bar chart to illustrate the data distribution. Each column represents the distribution of each data (age) about the population.

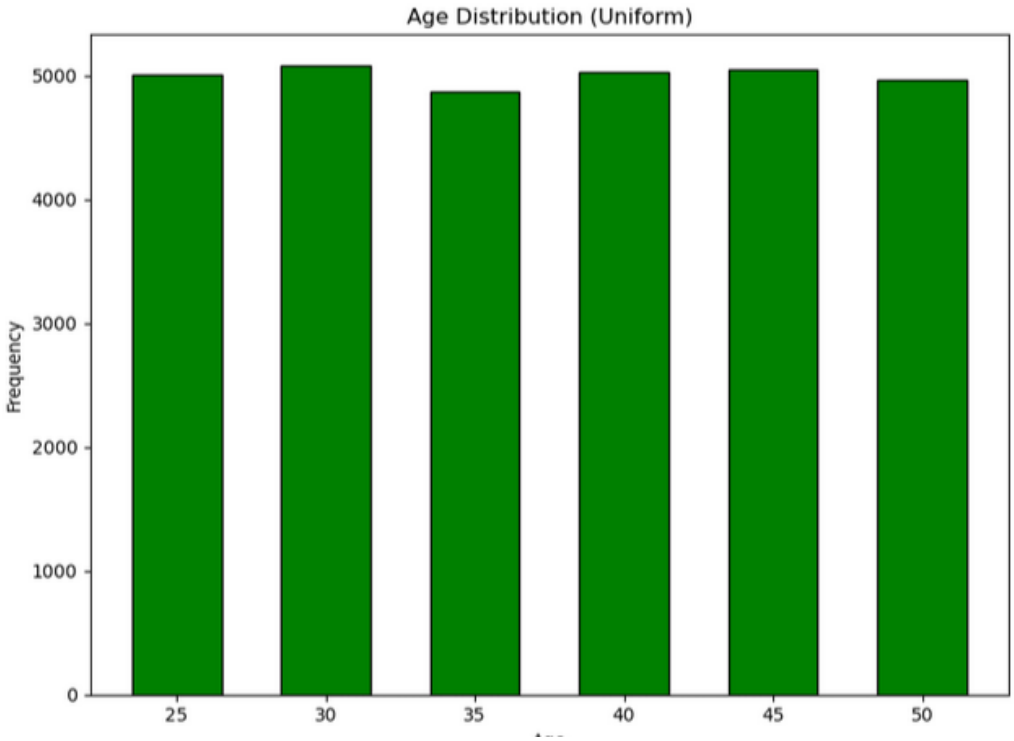

Diagram 1 – Age distribution bar chart

Since the environment is hypothetical and the distribution of the data is the same, the height of the columns has very little difference

Step4: Classification of purchase probability in Weighted probability statistic

Because we have a hypothetical model, we choose 10% for the lowest age range and add 10% to the probability for every 5 years of age increase, and we considered the data distribution for each age equal to 5000 of each age category compared to the total population.

```
# Calculating Weighing Possibilities

purchase_probs = {25: 0.10, 30: 0.20, 35: 0.30, 40: 0.40, 45: 0.50, 50: 0.60} #Allocation of purchase possibility based on individuals's age

purchases = [np.random.rand() < purchase_probs[age] for age in age_data] # Create an array of True or False to Determining whether the person made
                                                                          #a purchase or not
df = pd.DataFrame({'Age': age_data, 'Purchased': purchases}) # convert an array of true/false value to a DataFrame
```

Figure4- Python code of Classified Data frame

First, we allocate the possible percentage of purchases to each age group. Then, by checking the condition whether the random number created is lower or higher than the probability of the purchase of that age group, determine the status of the

person as a buyer. Finally, a new column named purchases is added to the data frame of Boolean type, which has values True for buyer and False for non-buyer.

41]:

	Age	Purchased
0	40	True
1	45	False
2	35	False
3	45	True
4	45	False
...	...	...
29995	40	False
29996	30	True
29997	50	False
29998	45	False
29999	45	True

30000 rows × 2 columns

Figure 5 -A preview of the status of buyers and non-buyers

We have a new column in the Data Frame of Boolean data type which represents the condition of purchases as buyer or not.

Formulation:

A: The probability of purchase(0 or 1)

B: The probability of the considered age.

Conditional probability formula: $\frac{P(A \cap B)}{P(B)} = P(A|B)$

$P(A|B)$: The probability of occurrence of A under the condition of occurrence of B OR $P(A|B) = 0.4\text{-}(Age)*0.02$

$P(A \cap B)$: The probability of the simultaneous occurrence of A and B $P(B) \times P(A|B)$

$P(B)$: the probability of occurrence B $\frac{Number\ of\ 25\ year\ olds}{Total\ data}$

Calculating the weighted probability statistics for a 45-year-old person:

$P(B) = 30000/6 \times 1/30000 = 1/6 \approx 0.1667$

$P(A, B) = 0.1667 \times 0.5 = 0.833$

$P(A|B) = 0.8333/0.1667 = 0.5$

Step 5: Calculation of weighted conditional probability

```
conditional_probs = {}   # Define dictionary to store probabilities
# Define lists to store values for plotting
p_B_list = []
p_A_given_B_list = []
p_occu_list = []

for age in ages:    # Exploring in ages for calculating their probalities
    df_age = df[df["Age"] == age]  # Filter data for each specific age range
        p_B = len(df_age) / n_samples              # Probability of occurrence P(B)
    p_A_given_B = df_age["Purchased"].mean()   # P(A | B)
    p_occu = p_B * p_A_given_B                 # P(A ∩ B) = P(A | B) * P(B)

    conditional_probs[age] = {"P(B)": p_B, "P(A|B)": p_A_given_B, "P(A∩B)": p_occu}  # saving in dictionary

    # Store values in lists for plotting
    p_B_list.append(p_B)
    p_A_given_B_list.append(p_A_given_B)
    p_occu_list.append(p_occu)
     # REPORTS
    print(f"  The age: {age}")
    print(f"  - P(B) for {age} years old: {p_B:.4f}")
    print(f"  - P(A | B) for {age} years old: {p_A_given_B:.4f}")
    print(f"  - P(A ∩ B) for {age} years old: {p_occu:.4f}\n")

# Set up the positions for each group of bars
x = np.arange(len(ages))  # Number of groups
width = 0.25  # Width of the bars
fig, ax = plt.subplots(figsize=(10, 7)) # Adjusting frame size for chart
# Create bars for each probability
ax.bar(x - width, p_B_list, width, label='P(B)', color='skyblue', edgecolor='black')
ax.bar(x, p_A_given_B_list, width, label='P(A|B)', color='lightgreen', edgecolor='black')
ax.bar(x + width, p_occu_list, width, label='P(A∩B)', color='salmon', edgecolor='black')

# Add labels, title, and custom x-axis tick labels
ax.set_xlabel('Age Groups', fontsize=12)
ax.set_ylabel('Probability', fontsize=12)
ax.set_title('Conditional Probabilities for Age Groups', fontsize=14)
ax.set_xticks(x)
ax.set_xticklabels(ages)
ax.legend()

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

Figure6- Python code for calculating the weighted conditional possibility

First, we define a dictionary to store the result of the possibilities. Then, start to explore in ages list, and a value True or False is returned to determine the buyer age. Subsequently, we earn P(B) by dividing the total number of people of that age group by the total

population. Meanwhile, the probability P(A|B) is obtained by taking the mean of the purchased array which is a list of 0 or 1 values for determining the purchases, this command is exactly the same as $\frac{P(A \cap B)}{P(B)}$. In the last step of calculating, the probability of P(A∩ B) is obtained by multiplying P(A|B) * P(B). Finally, all the possibilities are stored in dictionary.

Note: each age group receives the same amount of data, which is 5000 (1/6). Therefore, the possibility of buying is the same for all age groups (0.1667).

As a visualization, we draw the bar graph of each of the obtained possibilities. First, divide the graph space by the number of samples and set the width of each column at 0.25. Second, setting the frame size of the chart. Then start to draw each probability bar.

Extracting the result of weighted analysis calculations:

```
  The age: 25
- P(B) for the 25 years old 0.1670
- P(A | B): 25 years old 0.0990
- p(A n B) 25 years old 0.0165

  The age: 30
- P(B) for the 30 years old 0.1692
- P(A | B): 30 years old 0.1948
- p(A n B) 30 years old 0.0330

  The age: 35
- P(B) for the 35 years old 0.1624
- P(A | B): 35 years old 0.2945
- p(A n B) 35 years old 0.0478

  The age: 40
- P(B) for the 40 years old 0.1676
- P(A | B): 40 years old 0.3936
- p(A n B) 40 years old 0.0660

  The age: 45
- P(B) for the 45 years old 0.1681
- P(A | B): 45 years old 0.4939
- p(A n B) 45 years old 0.0830

  The age: 50
- P(B) for the 50 years old 0.1657
- P(A | B): 50 years old 0.5926
- p(A n B) 50 years old 0.0982
```

Figure 7 - a preview of calculation of weighted conditional probability for Ages

The result of calculating the weighted hypothetical probability of purchase for each age.

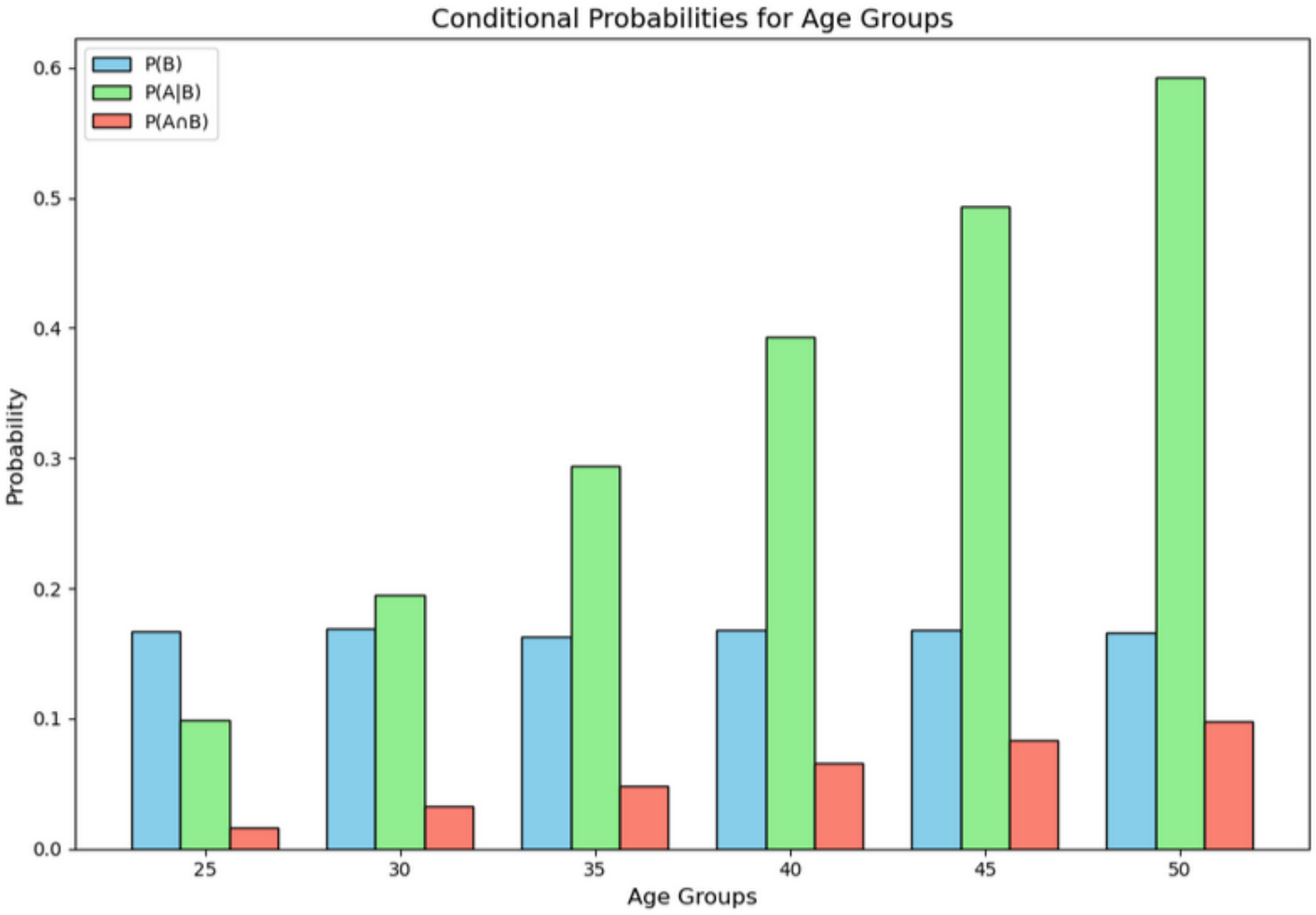

Diagram 2 – bar chart of weighted probabilities

P(A|B): Since the probability of purchase depends on age, it also increases with age, there is a linear and positive relationship between age and the probability of purchase, and the graph has a sharply upward trend.

P (B): Since our model is hypothetical, we took the same population distribution and purchase probability for all age groups. Thus, P(B) is the same for all ages, and has a Steady trend.

$P(A \cap B)$: The probability that the buyer is from the target age group at the same time and depending on the age has a slight upward trend.

Step 6: Calculation of unweighted conditional probability

The weight of age is removed and the probability of buying each person is calculated regardless of their age. We have a hypothetical model. Therefore, the distribution is uniform for all buyers, but the probability of purchase is different, the probability of buying for each age group is predetermined in is study without dependence on any conditions.

P(B): $\frac{\text{The total number of people of a particular age class}}{quantity\ of\ all\ people}$

P(A∩B): $\frac{The\ number\ of\ people\ of\ a\ certain\ age\ group\ who\ have\ made\ a\ purchase.}{quantity\ of\ all\ people}$

P(A|B): $\frac{P(A \cap B)}{P(B)}$

AGE GROUP	QUANTITY	THE PERCENT OF THE TOTAL SAMPLE
25	**5000**	**35%**
30	**5000**	**30%**
35	**5000**	**45%**
40	**5000**	**60%**
45	**5000**	**20%**
50	**5000**	**10%**

Table 1 - Table of un-weighted purchase probabilities

conditional probability for 45-year-old purchasers:

$P(B) = \frac{5000}{30000} = 0.1667$

$P(A \cap B) = \frac{0.20 \times 5000}{30000} = 0.033$

$P(A|B) = \frac{0.033}{0.1667} = 0.2$

```python
total_population = 30000 # population
group_population = 5000 #population of each sample


age_groups = {    # purchase probability for each age group
    "25": 0.35,
    "30": 0.30,
    "35": 0.45,
    "40": 0.60,
    "45": 0.20,
    "50": 0.10
}

ages = list(age_groups.keys())  # Access to ages
p_b = [group_population / total_population] * len(ages)  # P(B)
p_a_given_b = list(age_groups.values())  # P(A | B)
p_a_and_b = [p_b[i] * p_a_given_b[i] for i in range(len(ages))]  # P(A ∩ B) = P(B) * P(A|B)

# Report
for i in range(len(ages)):
    print(f"  The age: {ages[i]} years old")
    print(f"  - P(B) for {ages[i]} years old: {p_b[i]:.4f}")
    print(f"  - P(A | B) for {ages[i]} years old: {p_a_given_b[i]:.4f}")
    print(f"  - P(A ∩ B) for {ages[i]} years old: {p_a_and_b[i]:.4f}\n")


x = np.arange(len(ages))  # setting the bar location!
width = 0.25  # setting the bar width

plt.figure(figsize=(10, 6))
plt.bar(x - width, p_b, width=width, label="P(B)", color="skyblue") # Drawing the P(B) bar
plt.bar(x, p_a_and_b, width=width, label="P(A ∩ B)", color="orange") # Drawing the P(A,B) bar
plt.bar(x + width, p_a_given_b, width=width, label="P(A | B)", color="green") # Drawing the P(A|B) bar

# تنظیمات نمودار
plt.xlabel("Age Group") # x axe lable
plt.ylabel("Probability")# y axe lable
plt.title("Probability Distribution for Age Groups")
plt.xticks(x, ages)
plt.legend()
plt.show()
```

Figure 8- Python code for calculating un-weighted purchase probability

After introducing the 30,000 population, create a dictionary of purchase probabilities regardless of age. We consider the distribution of the data to be 5000 for all ages, But different purchase probabely (Table1)· Consequently, we assign the list from the dictionary to the ages and start to explore and calculate the unweighted probabilities. Finally, print and report probabilities.

Extracting the result of un-weighted analysis calculations:

```
The age: 25 years old
- P(B) for 25 years old: 0.1667
- P(A | B) for 25 years old: 0.3500
- P(A ∩ B) for 25 years old: 0.0583

The age: 30 years old
- P(B) for 30 years old: 0.1667
- P(A | B) for 30 years old: 0.3000
- P(A ∩ B) for 30 years old: 0.0500

The age: 35 years old
- P(B) for 35 years old: 0.1667
- P(A | B) for 35 years old: 0.4500
- P(A ∩ B) for 35 years old: 0.0750

The age: 40 years old
- P(B) for 40 years old: 0.1667
- P(A | B) for 40 years old: 0.6000
- P(A ∩ B) for 40 years old: 0.1000

The age: 45 years old
- P(B) for 45 years old: 0.1667
- P(A | B) for 45 years old: 0.2000
- P(A ∩ B) for 45 years old: 0.0333

The age: 50 years old
- P(B) for 50 years old: 0.1667
- P(A | B) for 50 years old: 0.1000
- P(A ∩ B) for 50 years old: 0.0167
```

Figure 9 - a preview of un-weighted conditional probability calculation for Ages

The result of calculating the weighted hypothetical probability of purchase for each age.

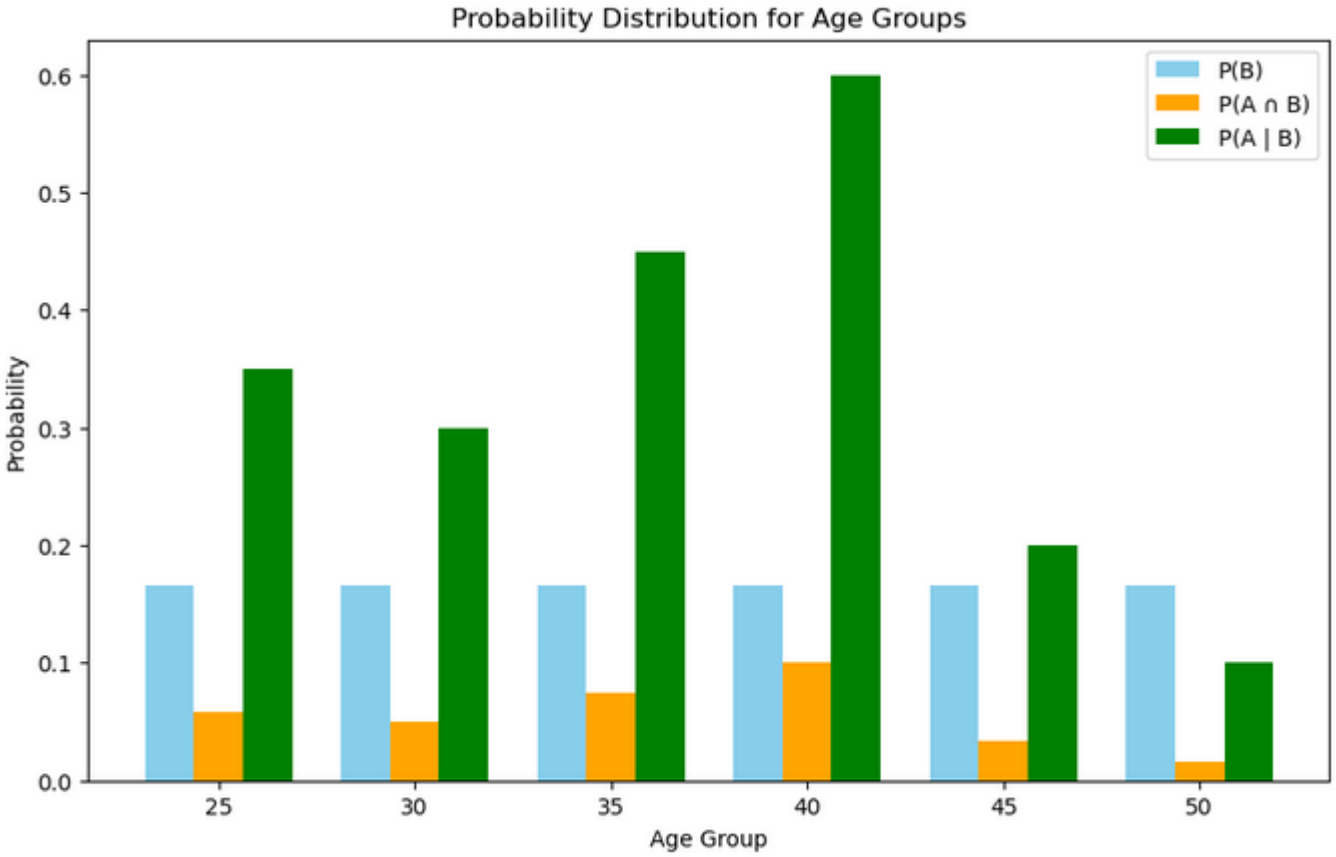

Diagram 3 – bar chart of un-weighted probabilities

P(A|B): Since the probability of purchase depends on age, it also increases with age, there is a linear and upward relationship between age and the probability of purchase, and the graph has an upward trend.

P(B): Since our model is hypothetical, we took the same population distribution for all age groups. So P(B) are the same for all groups.

$P(A \cap B)$: The probability that the buyer is already in the target age group has a significant fluctuation. There is not any constant upward or downward trend observed due to the different and un-weighted probability of purchase for each age.

Conclusion:

In the weighted probability, because of the assumption of an unreal environment, we assume a constant probability and distribution for age groups at (50%) (6/1), respectively. Therefore, a dependency was modeled where older people were more prefer to buy and the result was likely to have an upward trend. In un-weighted probabilities, because we decide to challenge Calculation, we consider a steady distribution for ages, (1/6), but variable probability purchase, (Table 1). The result depicts a lot of fluctuations in the obtained probabilities. Therefore, Increasing the conditional probability in the weighted calculation shows how the dependent purchase probabilities significantly change the outcome. The uniform model assumes that there are no such dependencies and that the probabilities are equal.

References:

McKinney, W. (2017). *Python for data analysis* (2nd ed.). O'Reilly Media.

Faller, W. (1968) An introduction to probability theory and its applications. (3rd ed.). Vol. 1. New York: Wiley, p. 2.

Knuth, D. E. (1997) The Art of Computer Programming. (3rd ed.). Vol. 2: Seminumerical Algorithms. Boston: Addision-Wesley, p. 10.

Python Codes:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
np.random.seed(42)
n_samples = 30000
ages = [25, 30, 35, 40, 45, 50]
age_data = np.random.choice(ages, size=n_samples)
age_counts = {age: np.sum(age_data == age) for age in ages}
plt.figure(figsize=(8, 6))
plt.bar(age_counts.keys(), age_counts.values(), color='green', edgecolor='black', width=3)
plt.title('Age Distribution (Uniform)')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.xticks(ages)
plt.tight_layout()
plt.show()
purchase_probs = {25: 0.10, 30: 0.20, 35: 0.30, 40: 0.40, 45: 0.50, 50: 0.60}
purchases = [np.random.rand() < purchase_probs[age] for age in age_data]
df = pd.DataFrame({'Age': age_data, 'Purchased': purchases})
print(df)
conditional_probs = {}
p_B_list = []
p_A_given_B_list = []
p_occu_list = []
for age in ages:
    df_age = df[df["Age"] == age]
    p_B = len(df_age) / n_samples
```

```
    p_A_given_B = df_age["Purchased"].mean()
    p_occu = p_B * p_A_given_B
    conditional_probs[age] = {"P(B)": p_B, "P(A|B)": p_A_given_B, "P(A∩B)": p_occu}
    p_B_list.append(p_B)
    p_A_given_B_list.append(p_A_given_B)
    p_occu_list.append(p_occu)
    print(f"  The age: {age}")
    print(f"  - P(B) for {age} years old: {p_B:.4f}")
    print(f"  - P(A | B) for {age} years old: {p_A_given_B:.4f}")
    print(f"  - P(A ∩ B) for {age} years old: {p_occu:.4f}\n")
x = np.arange(len(ages))  # Number of groups
width = 0.25  # Width of the bars
fig, ax = plt.subplots(figsize=(10, 7))
ax.bar(x - width, p_B_list, width, label='P(B)', color='skyblue', edgecolor='black')
ax.bar(x, p_A_given_B_list, width, label='P(A|B)', color='lightgreen', edgecolor='black')
ax.bar(x + width, p_occu_list, width, label='P(A∩B)', color='salmon', edgecolor='black')
ax.set_xlabel('Age Groups', fontsize=12)
ax.set_ylabel('Probability', fontsize=12)
ax.set_title('Conditional Probabilities for Age Groups', fontsize=14)
ax.set_xticks(x)
ax.set_xticklabels(ages)
ax.legend()
fig.tight_layout()
plt.show()
total_population = 30000
group_population = 5000
```

```
age_groups = {
    "25": 0.35,
    "30": 0.30,
    "35": 0.45,
    "40": 0.60,
    "45": 0.20,
    "50": 0.10
}

ages = list(age_groups.keys())
p_b = [group_population / total_population] * len(ages)
p_a_given_b = list(age_groups.values())
p_a_and_b = [p_b[i] * p_a_given_b[i] for i in range(len(ages))]
for i in range(len(ages)):
    print(f"  The age: {ages[i]} years old")
    print(f"  - P(B) for {ages[i]} years old: {p_b[i]:.4f}")
    print(f"  - P(A | B) for {ages[i]} years old: {p_a_given_b[i]:.4f}")
    print(f"  - P(A ∩ B) for {ages[i]} years old: {p_a_and_b[i]:.4f}\n")
x = np.arange(len(ages))
width = 0.25

plt.figure(figsize=(10, 6))
plt.bar(x - width, p_b, width=width, label="P(B)", color="skyblue")
plt.bar(x, p_a_and_b, width=width, label="P(A ∩ B)", color="orange")
plt.bar(x + width, p_a_given_b, width=width, label="P(A | B)", color="green")
plt.xlabel("Age Group")
plt.ylabel("Probability")
plt.title("Probability Distribution for Age Groups")
plt.xticks(x, ages)
plt.legend
```