CS215 Assignment 1

Abhineet Majety 23B0923
Mohana Evuri 23B1017
Saksham Jain 23B1074

1 Let's Gamble

There are two friends playing a dice-roll game. Friend A has $(n+1)$ fair dice and Friend B has n fair dice (a fair die has equal probability of every face). On every roll, a win is achieved if we get a prime number on the top. What is the probability that A will have more wins than B if both roll all of their dice?

Each roll's number on top for a win is $\{2,3,5\}$. Since the dice are fair, the probability of a win is $\frac{3}{6} = \frac{1}{2}$. Each roll is independent.
First, consider the first n rolls. Let X_n be the number of wins with X after n rolls. Then, by symmetry, we have

$$P(A_n > B_n) = P(B_n > A_n) \tag{1}$$

Now, we have

$$\begin{aligned} P(A_n = B_n) &= \sum_{i=1}^{n} P(A_n = B_n = i) \\ &= \sum_{i=1}^{n} P(A_n = i)P(B_n = i) \\ &= \sum_{i=1}^{n} {}^{n}C_i \left(\frac{1}{2}\right)^n \cdot {}^{n}C_i \left(\frac{1}{2}\right)^n \\ &= \sum_{i=1}^{n} {}^{n}C_i^2 \left(\frac{1}{2}\right)^{2n} \\ &= \left(\frac{1}{2}\right)^{2n} \cdot {}^{2n}C_n \end{aligned} \tag{2}$$

Since the three events are mutually exclusive, we have

$$P(A_n > B_n) + P(A_n < B_n) + P(A_n = B_n) = 1 \tag{3}$$

From (1), (2) and (3), we have

$$P(A_n > B_n) = \frac{1}{2} - (\frac{1}{2})^{2n+1} \cdot {}^{2n}C_n \tag{4}$$

Now, for $(n+1)$ rolls, using (2) and (4),

$$\begin{aligned} P(A_{n+1} > B_n) &= P(A_n > B_n) + P(A_n = B_n \cap A_{n+1} = A_n + 1) \\ &= \frac{1}{2} - \left(\frac{1}{2}\right)^{2n+1} \cdot {}^{2n}C_n + \left(\frac{1}{2}\right) \cdot \left(\frac{1}{2}\right)^{2n} \cdot {}^{2n}C_n \\ &= \frac{1}{2} \end{aligned} \tag{5}$$

Hence, the probability of A having more wins than B at the end is $\frac{1}{2}$.

2 Two Trading Teams

You are playing a trading game against two teams A and B (will happen in reality soon). The game is played in the form of a three-set series with A and B alternately. Also, Team B is better at trading than Team A. To encourage your trading career, the exchange (an organization responsible for managing the trades) gives you two options A-B-A (which means you play a game with Team A, then Team B and at last Team A again) or B-A-B. You will win if you win two sets in a row. Which of the two options should you choose? Justify your choice with proper calculations.

Let p_A and p_B be the probabilities of a win against A and B, respectively. Since B plays better than A, we have $p_A > p_B$. (Assuming that each set is independent of the others)

Case-1: The option A-B-A is chosen. I can win if

1. I win the first two sets:
$$P(win_1) = p_A \cdot p_B$$

2. I lose the 1st set, win the 2nd and 3rd set:
$$P(win_2) = (1 - p_A) \cdot p_B \cdot p_A$$

Hence, the probability is

$$\begin{aligned} P(win_{ABA}) &= P(win_1) + P(win_2) \\ &= p_A p_B + (1 - p_A) p_B p_A \\ &= p_A p_B (2 - p_A) \end{aligned} \tag{1}$$

Case-2: The option B-A-B is chosen. I can win if

1. I win the first two sets:
$$P(win_1) = p_B \cdot p_A$$

2. I lose the 1st set, win the 2nd and 3rd set:
$$P(win_2) = (1 - p_B) \cdot p_A \cdot p_B$$

Hence, the probability is

$$\begin{aligned} P(win_{BAB}) &= P(win_1) + P(win_2) \\ &= p_B p_A + (1 - p_B) p_A p_B \\ &= p_A p_B (2 - p_B) \end{aligned} \tag{2}$$

Since $p_B < p_A$, it follows that $P(win_{BAB}) > P(win_{ABA})$. Hence, I would choose the option B-A-B.

3 Random Variables

3.1

Let Q_1, Q_2 be non-negative random variables. Let $P(Q_1 < q_1) \geq 1 - p_1$ and $P(Q_2 < q_2) \geq 1 - p_2$, where q_1, q_2 are non-negative. Then show that $P(Q_1 Q_2 < q_1 q_2) \geq 1 - (p_1 + p_2)$

Since Q_1 and Q_2 are non-negative, we have

$$P(Q_1 Q_2 < q_1 q_2) \geq P(Q_1 < q_1 \cap Q_2 < q_2) \tag{1}$$

Define the pairwise mutually exclusive events E_1, E_2, E_3 as

$$\begin{aligned} E_1 &:= \{Q_1 < q_1 \cap Q_2 < q_2\} \\ E_2 &:= \{Q_1 < q_1 \cap Q_2 \geq q_2\} \\ E_3 &:= \{Q_1 \geq q_1 \cap Q_2 < q_2\} \end{aligned} \tag{2}$$

Then, we have

$$\begin{aligned} P(E_1) + P(E_2) &= 1 - p_1 \\ P(E_1) + P(E_3) &= 1 - p_2 \\ P(E_1) + P(E_2) + P(E_3) &\leq 1 \end{aligned} \tag{3}$$

Using the equations in (3), we have

$$\begin{aligned} 2P(E_1) + P(E_2) + P(E_3) &= 2 - p_1 - p_2 \\ \implies 2 - p_1 - p_2 &\leq 1 + P(E_1) \\ \implies P(E_1) &\geq 1 - (p_1 + p_2) \end{aligned} \tag{4}$$

Finally, using (1) and (4), we have

$$P(Q_1 Q_2 < q_1 q_2) \geq 1 - (p_1 + p_2) \tag{5}$$

3.2

> Given n distinct values $\{x_i\}_{i=1}^n$ with mean μ and standard deviation σ, prove that for all i, we have $|x_i - \mu| \leq \sigma\sqrt{n-1}$. How does this inequality compare with Chebyshev's inequality as n increases? (give an informal answer)

Consider the definition of σ,

$$\begin{aligned} \sigma &= \sqrt{\frac{\sum_{i=1}^n (x_i - \mu)^2}{n-1}} \\ \implies \sigma\sqrt{n-1} &= \sqrt{\sum_{i=1}^n (x_i - \mu)^2} \end{aligned} \tag{1}$$

Now, for all $1 \leq i \leq n$, we have

$$|x_1 - \mu| \leq \sqrt{\sum_{i=1}^n (x_i - \mu)^2} \tag{2}$$

Hence, from (1), we have

$$|x_i - \mu| \leq \sigma\sqrt{n-1} \tag{3}$$

for all $1 \leq i \leq n$.
As n increases, this inequality gives significantly less information about the distribution of x_i, but it provides an upper bound on the value x_i can take. On the other hand, Chebyshev's inequality gives a bound on the number of x_i, which has a high deviation from the mean.

4 Staff Assistant

> You need a new staff assistant, and you have n people to interview. You want to hire the best candidate for the position. When you interview a candidate, you can give them a score, with the highest score being the best and no ties being possible.
>
> You interview the candidates one by one. Because of your company's hiring practices, after you interview the k^{th} candidate, you either offer the candidate the job before the next interview or you forever lose the chance to hire that candidate. We suppose the candidates are interviewed in a random order, chosen uniformly at random from all $n!$ possible orderings.
>
> We consider the following strategy. First, interview m candidates but reject them all: these candidates give you an idea of how strong the field is. After the m^{th} candidate. hire the first candidate you interview who is better than all of the previous candidates you have interviewed.

(a)

Let E be the event that we hire the best assistant, and let E_i; be the event that i^{th} candidate is the best and we hire him. Determine $Pr(E_i)$, and show that

$$Pr(E) = \frac{m}{n} \sum_{j=m+1}^{n} \frac{1}{j-1}$$

Let i be the position of the best assistant.
When $i \leq m$: $\Pr(E_i) = 0$ because the first m candidates are rejected.
When $i \geq m$: Consider a collection C of orderings in which the candidates appearing before i and those appearing after i are the same. The ith candidate (the best assistant) will be selected if the candidate with the highest score among the first $i-1$ candidates is among the first m candidates. If that candidate were not in the first m candidates, he would be selected before the ith candidate. If he were in the first m candidates, the first candidate better than that candidate is the ith candidate. Hence, the best assistant would be selected in this case.
The probability of the best assistant being selected in this collection is

$$\Pr(E_i|C) = \frac{m}{i-1}. \tag{1}$$

Since the ordering is uniformly random, the same probability of E_i holds for any such collection of orderings. Here, $\cup C$ is the set of orderings where the ith candidate is the best assistant. All the collections C as defined above are pairwise mutually exclusive. Hence, by total probability theorem, $\forall i > m$,

$$\begin{aligned} \Pr(E_i) &= \sum \frac{m}{i-1} \cdot \Pr(C) \\ &= \frac{m}{i-1} \cdot \sum \Pr(C) \\ &= \frac{m}{i-1}. \end{aligned} \tag{2}$$

Let $\Pr(i)$ be the probability of the best assistant being at position i. Then,

$$\begin{aligned} \Pr(E) &= \sum_{i=1}^{n} \Pr(E_i) \cdot \Pr(i) \\ &= \sum_{i=1}^{m} \Pr(E_i) \Pr(i) + \sum_{i=m+1}^{n} \Pr(E_i) \Pr(i) \\ &= \sum_{i=m+1}^{n} \frac{m}{i-1} \cdot \frac{1}{n} \\ &= \frac{m}{n} \sum_{j=m+1}^{n} \frac{1}{j-1}. \end{aligned} \tag{3}$$

(b)

Bound $\sum_{j=m+1}^{n} \frac{1}{j-1}$ to obtain:

$$\frac{m}{n}(\ln(n) - \ln(m)) \leq Pr(E) \leq \frac{m}{n}(\ln(n-1) - \ln(m-1))$$

We know that $\ln(1+x) < x$ for $x > 0$ and $\ln(1+x) > x$ for $x < 0$. Hence, for $j \geq 2$,

$$\begin{aligned} &\ln\left(1+\frac{1}{j-1}\right) \leq \frac{1}{j-1} \text{ and } -\frac{1}{j-1} \geq \ln\left(1-\frac{1}{j-1}\right) \\ \implies &\ln\left(\frac{j}{j-1}\right) \leq \frac{1}{j-1} \leq \ln\left(\frac{1}{1-\frac{1}{j-1}}\right) \\ \implies &\ln\left(\frac{j}{j-1}\right) \leq \frac{1}{j-1} \leq \ln\left(\frac{j-1}{j-2}\right). \end{aligned} \tag{4}$$

Hence, from (3) and (4),

$$\begin{aligned} &\frac{m}{n}\sum_{j=m+1}^{n} \ln\left(\frac{j}{j-1}\right) \leq \Pr(E) \leq \frac{m}{n}\sum_{j=m+1}^{n} \ln\left(\frac{j-1}{j-2}\right) \\ \implies &\frac{m}{n}(\ln(n) - \ln(m)) \leq \Pr(E) \leq \frac{m}{n}(\ln(n-1) - \ln(m-1)). \end{aligned} \tag{5}$$

(c)

Show that $\frac{m}{n}(\ln(n) - \ln(m))$ is maximized when $m = \frac{n}{e}$, and explain why this means $Pr(E) \geq \frac{1}{e}$ for this choice of m.

Let $k = \dfrac{m}{n}$. Then

$$\begin{aligned} \frac{m}{n}(\ln(n) - \ln(m)) &= k\ln(\frac{1}{k}) \\ &= -k\ln(k). \end{aligned} \tag{6}$$

Now,

$$\frac{d(-k\ln(k))}{dk} = -1 - \ln(k) \tag{7}$$

which becomes 0 at $k = \dfrac{1}{e}$. Also

$$\begin{aligned} \frac{d^2(-k\ln(k))}{dk^2} &= -\frac{1}{k} \\ \frac{d^2(-k\ln(k))}{dk^2}\Big|_{k=\frac{1}{e}} &= -e < 0. \end{aligned} \tag{8}$$

Hence, the function is maximized when $k = \dfrac{1}{e}$, i.e., when $m = \dfrac{n}{e}$. For this choice of m, using the lower bound from equation (5),

$$\Pr(E) \geq \frac{m}{n}(\ln(n) - \ln(m)) = \frac{1}{e}. \tag{9}$$

5 Free Trade

Imagine an infinitely long line of traders waiting outside a brokerage firm to place their trades. Each trader is assigned an ID number from 1 to 200 (both inclusive, obviously these IDs are not unique). The firm's director announces a special offer: the first trader in the queue whose ID number matches the ID of any trader who has already placed a trade will receive a free trade (i.e., a trade without any margins). You have the option to choose your position in this queue. However, you don't know the ID numbers of the traders ahead of you or behind you. Your goal is to maximize your chances of being the first trader whose ID matches someone who has already placed a trade. Given this situation, what position in the queue should you choose to maximize your chances of receiving the free trade?

Let $P(n)$ be the probability of the first *free trade* to occur at the nth position.
For the first trade to occur at the nth position, the ID number of the first $(n-1)$ traders should be distinct, and the ID of the nth trader should be one of the first $n-1$ numbers. The total number of possible values of the first n ID numbers is 200^n. Hence,

$$\begin{aligned} P(n) &= \frac{^{200}P_{n-1} \cdot (n-1)}{200^n} \\ &= \frac{200! \cdot (n-1)}{(201-n)! \cdot 200^n} \end{aligned} \tag{1}$$

We want to study the behavior of $P(n)$ and find its maxima. We can see that for $n > 201$, $P(n) = 0$. This is because the first repetition of the number will occur in at most the 201th digit (by the Pigeonhole principle). Also, $P(1) = 0$ because repetition cannot occur in the first position. For $2 \leq n \leq 201$, $P(n)$ is non-zero.
Now, using (1)

$$\begin{aligned} \frac{P(n-1)}{P(n)} &= \frac{(n-2) \cdot (201-n)! \cdot 200^n}{(202-n)! \cdot (n-1) \cdot 200^{n-1}} \\ &= \frac{200 \cdot (n-2)}{(202-n) \cdot (n-1)} \end{aligned} \tag{2}$$

Hence,

$$\frac{P(n-1)}{P(n)} - 1 = \frac{n^2 - 3n - 198}{(202-n) \cdot (n-1)} \tag{3}$$

In (3), the denominator is always positive, and the numerator is positive for $n \geq 16$. Hence, the chances of receiving free trade are maximum at the 15th position.

6 Update Functions

Suppose that you have computed the mean, median and standard deviation of a set of n numbers stored in array A where n is very large. Now, you decide to add another number to A. Write a python function to update the previously computed mean, another python function to update the previously computed median, and yet another python function to update the previously computed standard deviation. Note that you are not allowed to simply recompute the mean, median or standard deviation by looping through all the data. You may need to derive formulae for this. Include the formulae and their derivation in your report. Note that your python functions should be of the following form:

```
function newMean = UpdateMean(OldMean, NewDataValue, n, A),
function newMedian = UpdateMedian(OldMedian, NewDataValue, n, A),
function newStd = UpdateStd(OldMean, OldStd, NewMean, NewDataValue, n, A).
```

Also explain, how would you update the histogram of A, if you received a new value to be added to A? (Only explain, no need to write code.) Please specify clearly if you are making any assumptions.

Updating Mean

Let μ_{old} be the mean before adding the new number x_{n+1}. Then

$$\mu_{old} = \frac{\sum_{i=1}^{n} x_i}{n}. \tag{1}$$

Let μ_{new} be the updated mean. Then using (1),

$$\begin{aligned} \mu_{new} &= \frac{\sum_{i=1}^{n+1} x_i}{n+1} \\ &= \frac{(\sum_{i=1}^{n} x_i) + x_{n+1}}{n+1} \\ &= \frac{\mu_{old} \cdot n + x_{n+1}}{n+1}. \end{aligned} \tag{2}$$

The final code:

```
def UpdateMean(OldMean: float, NewDataValue: float, n: int, A: list[float]) -> float:
    return (OldMean * n + NewDataValue) / (n + 1)
```

Updating Median

I have assumed that the array A is sorted. If the number of elements in the updated array is even, the function `UpdateMedian` outputs the **average** of the two middle numbers. To update the median, we need to know the position of the new number relative to the old median. Let M_{old} and M_{new} be the old and updated medians, respectively. Let x_{new} be the number added.
If n is odd, i.e., the updated array has an even number of elements, then,

$$M_{new} = \begin{cases} \dfrac{x_{\frac{n+1}{2}} + x_{\frac{n+3}{2}}}{2} & x_{new} \geq x_{\frac{n+3}{2}} \\ \dfrac{x_{\frac{n+1}{2}} + x_{new}}{2} & x_{\frac{n+3}{2}} > x_{new} > x_{\frac{n-1}{2}} \\ \dfrac{x_{\frac{n+1}{2}} + x_{\frac{n-1}{2}}}{2} & x_{\frac{n+3}{2}} \geq x_{new}. \end{cases} \tag{1}$$

If n is even,

$$M_{new} = \begin{cases} x_{\frac{n}{2}+1} & x_{new} \geq x_{\frac{n}{2}+1} \\ x_{new} & x_{\frac{n}{2}+1} > x_{new} > x_{\frac{n}{2}} \\ x_{\frac{n}{2}} & x_{\frac{n}{2}} \geq x_{new}. \end{cases} \tag{2}$$

The final code:

```
def UpdateMedian(OldMedian: float, NewDataValue: float, n: int, A: list[float]) -> float:
    if n % 2 == 0:
        if NewDataValue >= A[n // 2]:
            return A[n // 2]
        elif NewDataValue <= A[n // 2 - 1]:
            return A[n // 2 - 1]
        else:
            return NewDataValue
    else:
        if NewDataValue >= A[n // 2 + 1]:
            return (A[n // 2] + A[n // 2 + 1]) / 2
        elif NewDataValue <= A[n // 2 - 1]:
            return (A[n // 2 - 1] + A[n // 2]) / 2
        else:
            return (A[n // 2] + NewDataValue) / 2
```

Updating Standard Deviation

Let σ_{old} be the standard deviation before adding the new number x_{n+1} to the array A. Let μ_{old} be the old mean. Then, by definition and using (1),

$$\begin{aligned}
\sigma_{old} &= \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu_{old})^2}{n-1}} \\
&= \sqrt{\frac{\sum_{i=1}^{n}(x_i^2 - 2\mu_{old}\cdot x_i + \mu_{old}^2)}{n-1}} \\
&= \sqrt{\frac{\sum_{i=1}^{n} x_i^2 - 2\mu_{old}\sum_{i=1}^{n} x_i + n\mu_{old}^2}{n-1}} \\
&= \sqrt{\frac{\sum_{i=1}^{n} x_i^2 - n\mu_{old}^2}{n-1}} \\
\implies \sum_{i=1}^{n} x_i^2 &= (n-1)\sigma_{old}^2 + n\mu_{old}^2.
\end{aligned} \tag{3}$$

Let σ_{new} be the updated standard deviation. Using the above equation,

$$\begin{aligned}
\sigma_{new} &= \sqrt{\frac{\sum_{i=1}^{n+1} x_i^2 - (n+1)\mu_{new}^2}{(n+1)-1}} \\
&= \sqrt{\frac{(n-1)\sigma_{old}^2 + n\mu_{old}^2 + x_{n+1}^2 - (n+1)\mu_{new}^2}{n}}.
\end{aligned} \tag{4}$$

The final code:

```python
import math

def UpdateStd(OldMean: float, OldStd: float, NewMean: float, NewDataValue: float, n: int, A:
list[float]) -> float:
    return math.sqrt((n * pow(OldMean, 2) + (n - 1) * pow(OldStd, 2) + pow(NewDataValue, 2) - (n +
        1) * pow(NewMean, 2)) / n)
```

Updating Histogram

If the new number lies in one of the bins of the old histogram, then the corresponding height will be increased by one. If it does not lie in any bin, a new bin will be created with only the new number.

Code

The python code is in the file `UpdateFunctions/update_functions.py`.

7 Plots

Read about the following plots:

- Violin Plot
- Pareto Chart
- Coxcomb Chart
- Waterfall Plot

Describe the uses of these plots. Take some sample data and generate one example plot for each of them.

7.1 Violin Plot

Violin plot is a powerful data visualization tool that helps see **numeric data distribution** and is also used to **compare two data sets**. It combines the features of a **box plot and a density plot**, giving a detailed view of how the data is distributed throughout the range.
The main feature of a violin plot is its shape, which resembles a violin. The **width** of the plot at different points tells about the **number of data points**. Violin plots include a **marker for the median** (a red horizontal line here). It often contains a box plot that shows the interquartile range (IQR). The top and bottom of the box represent the third quartile (Q3) and first quartile (Q1). The whiskers extend from the edges of the IQR box to show the range of the data. They typically extend up to 1.5 times the IQR from the quartiles.
Violin plot is used to understand detailed distribution characteristics, compare groups, or perform exploratory data analysis, especially with large datasets or when investigating patterns and trends.
An example plot:

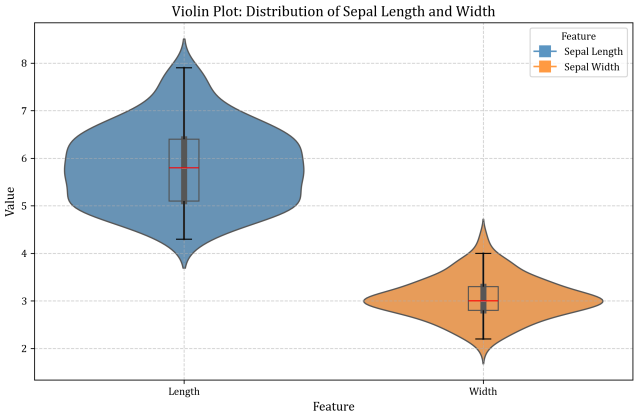

Figure 1: Violin Plot

Violin plots representing two data sets(sepal length and sepal width from the iris dataset) are shown.
A violin plot compares two data sets by visualizing their center, spread, and distribution. It shows that sepal length has a broader distribution with significant variability, possibly multiple peaks, and a wider IQR, indicating more diversity. In contrast, sepal width has a narrower range with less variability, suggesting a more homogeneous data set.

7.2 Pareto Chart

A **Pareto chart** contains bars and a line graph. The **bars** represent the **frequency** or magnitude of individual categories in **descending order**, whereas the **line graph** shows the **cumulative percentage** of these categories.
The key idea is based on the **Pareto principle**, also known as the **80/20 rule**, which suggests that **80% of effects come from 20% of causes**.

Uses

1. **Prioritizing Issues**: The Pareto chart highlights the most important among a (typically large) set of factors.
2. **Quality Control**: Pareto charts help identify and prioritize the defects to observe the most significant overall improvement. It often represents the most common sources of defects, the highest occurring type, and the most frequent reasons for customer complaints.
3. **Decision-Making**: Aids make informed decisions by visually distinguishing between the ”vital few” and the ”trivial many,” ensuring efforts are directed towards the most impactful areas.

An example plot:

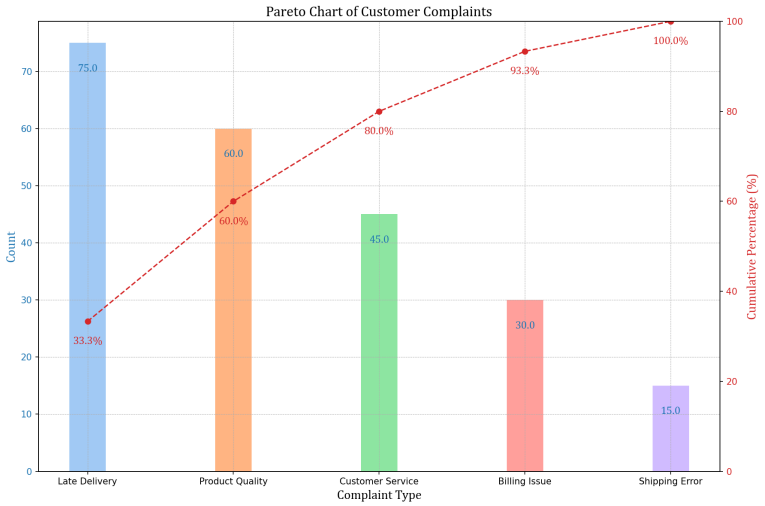

Figure 2: Pareto Chart

7.3 Coxcomb Chart

Coxcomb charts are **modified pie charts** where each slice/section represents some category or interval. In a pie chart, **measures** are **encoded using angles**, i.e., the larger the value, the larger the angle. Each **radial bar** extends from the center and represents the **magnitude of a feature** for a specific category, arranged in a circular format. The length of these bars indicates the proportions of the feature, with **longer bars** indicating **higher values**.

An example plot:

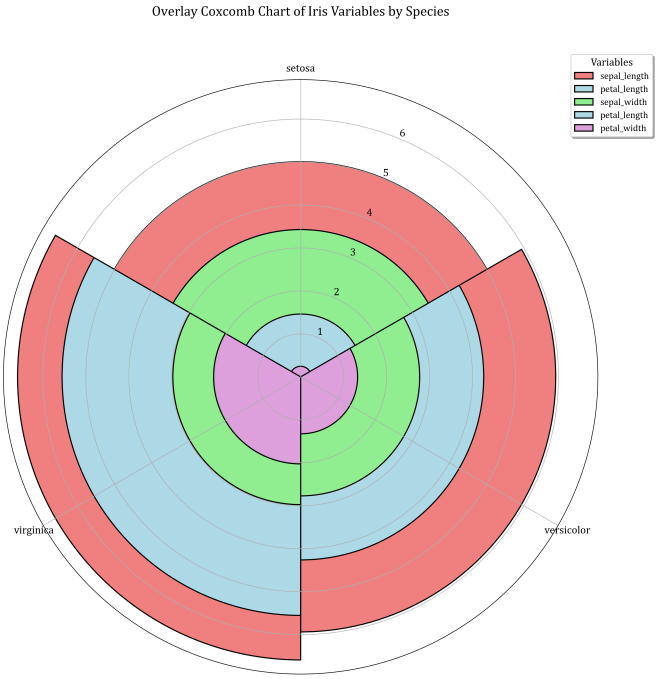

Figure 3: Coxcomb Chart

In the given example, each chart sector is assigned to a species (Setosa, Virginica, or Versicolor), making comparing features across these categories easy. Features (sepal length, sepal width, petal length, or petal width) are represented using unique colors, which helps understand their distribution and importance. Overlapping bars in

multiple colors offer a layered perspective on how each element contributes to the overall data for each category, revealing more elaborate insights into their interactions. The chart is labeled to identify each category, and a legend is provided to explain the color coding for categories. This design allows for precise and engaging visualization of complex data, making analyzing and comparing numerous variables within each category easier.

7.4 Waterfall Plot

A **Waterfall plot** is a 3-D plot in which multiple curves of data, such as spectra and time-series, are displayed simultaneously. The curves are staggered across both the screen and vertically, with the nearer ones closer to us. As a result, we see a series of similar curves appearing side-by-side.

An example plot:

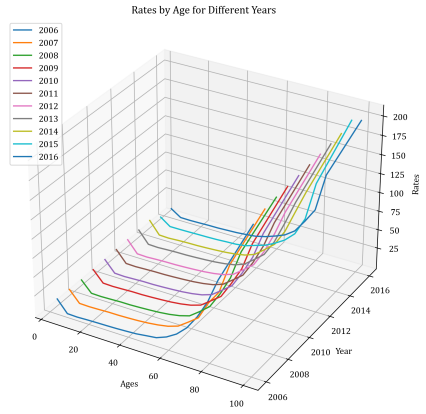

Figure 4: Waterfall Plot

The example shown plots the death rate by age group for the years 2006 to 2016. The data from the years 2006, 2007, 2008, 2015, and 2016 are taken from the Indian government's website. Data for the remaining years has been projected using the known data. The values are hard-coded into the Python file. The plot shows the variation in the death rate trends for different age groups for different years.
Uses:

1. Showing the results of **spectral density estimation** at successive intervals of time.
2. Spectra at different engine speeds when **testing engines**.

Code

The `python` code and the output images are in the folder `Plots`.
Code: `Plots/*.py`
Images: `Plots/plots`

8 Monalisa

Download the image of Monalisa from here. Read the image using matplotlib (example). Write a piece of python code to shift the image along the X direction by t_x pixels where t_x is an integer ranging from -10 to +10 (so, in total you need to do this for 20 values). While doing so, assign a value of 0 to unoccupied pixels. For each shift, compute the correlation coefficient between the original image and its shifted version. Make a plot of correlation coefficients across the shift values. Also, generate a normalized histogram for the original image. You might need to refer to section 3.3 from this book. You are not allowed to use any inbuilt function for generating the histogram. If you are using any other libraries, then please mention about them in the pdf.

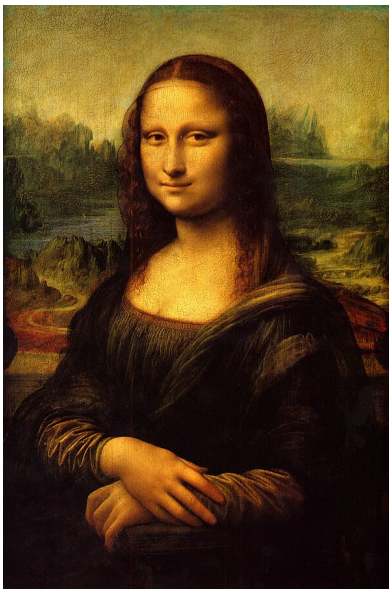

Figure 5: The MonaLisa image used (594x899 pixels)

In the first part of the question, we have to compute and plot the correlation coefficient between the original image and its shifted versions. Firstly, our code reads the Monalisa image using matplotlib; then, we use the slicing method in Python to shift the original image along the X direction by t_x pixels ranging from [-10, 10]. We compute correlation coefficients using the pearsonr function from the scipy library of Python and plot the graph for all 21 t_x values. The graph is symmetric about the value 0 (No shifting of pixels in the original image).

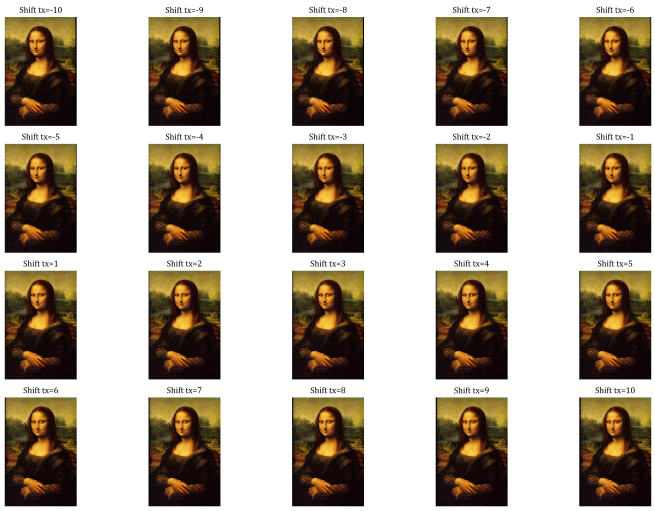

Figure 6: Monalisa with various shifts(tx)

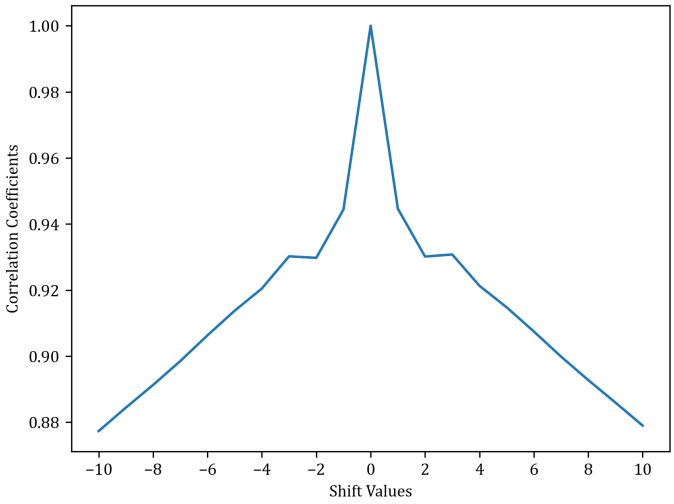

Figure 7: Correlation Coefficient Graph

In the second part of the question, we have to plot a normalized histogram for the original image without using any built-in function in Python. Firstly, our code reads the original image and converts it to a gray channel from the PIL library. We also convert the original into red, blue, and green channels. The normalized histogram for a particular channel of the original image is defined as $p(r_k) = \frac{n_k}{MN}$ where n_k is the no. of pixels in particular of the original image with intensity r_k; M, and N is the number of image rows and columns, respectively. The sum of p(r_k) for all values of k is always 1.

Here is the graph representing the channel pixel intensities:
NOTE: The x-axis of the graphs in this figure represent *Pixel Value* and the y-axis represents the $p(r_k)$.

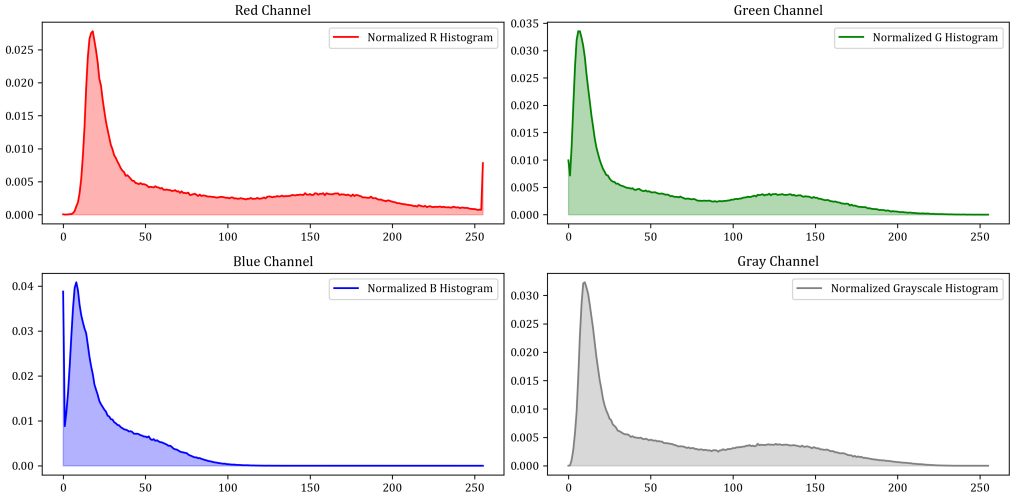

Figure 8: Channel Pixel Intensities Histogram

Code

The python code and the outputs are in the folder `Monalisa`.
Code: `Monalisa/monalisa.py`