Week 1 – Data Analysis Assignments

This document contains my Week 1 submissions for the Data Analysis course. It includes both the probability assignment and the introductory NumPy–Matplotlib exercises. I approached these as practice-heavy tasks, focusing on understanding the ideas while writing clean, working solutions.

Assignment 1: Probability and Random Variables

This assignment was mainly about getting comfortable with probability reasoning and standard random-variable setups. Some questions were straightforward once written properly, while others became much easier after using complements or symmetry instead of brute-force listing.

Q1. Rain on Two Days

Instead of listing cases, I found it simpler to start from what was given directly. Since the probability that it does not rain on either day is 0.3, the probability that it rains on at least one day is 0.7. Using this, the remaining probabilities follow by basic inclusion–exclusion.
Final values: Rain today or tomorrow = 0.7, rain on both days = 0.4, rain today but not tomorrow = 0.2, rain on exactly one day = 0.3.

Q2. Two Dice

I visualized the sample space as a 6×6 grid of outcomes. The pairs that sum to 8 lie along a diagonal. There are five such outcomes out of thirty-six total, giving a probability of 5/36.

Q3. Children Problem

Once one randomly chosen child is known to be a girl, the remaining children are still equally likely to be boys or girls. This leads directly to a probability of (1/2)^(n–1) that all children are girls.

Q4–Q12 Summary

The remaining questions involved standard distributions and properties of jointly normal variables. Using known results for mixed distributions, covariance, normal approximations, and conditional distributions made these problems manageable without unnecessary computation.

Assignment 2: NumPy and Matplotlib

This assignment was my first proper hands-on experience using NumPy arrays and plotting data with Matplotlib. The goal here was not complexity, but comfort—running code, checking outputs, and seeing how small changes affect results.

NumPy Arrays

```
import numpy as np

numbers = np.array([[2, 4, 6],
                    [3, 7, 11]])
print(numbers)
```

```
print(numbers.shape)
```

I intentionally used a 2×3 matrix so that rows and columns are clearly different. Printing the shape helped reinforce how NumPy stores dimensions.

Random Values

```
random_vals = np.random.rand(2, 3)
print(random_vals)
```

Re-running this cell shows how NumPy generates different values each time, which made the behavior of random arrays clearer.

Line Plot

```
import matplotlib.pyplot as plt

time_points = [0, 1, 2, 3, 4]
temperature = [22, 21, 23, 22, 25]

plt.plot(time_points, temperature, color='darkorange')
plt.xlabel("Time")
plt.ylabel("Temperature")
plt.title("Temperature vs Time")
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```

Adding labels, a title, and a grid made the plot easier to read. This step helped me understand how Matplotlib handles basic visualization tasks.

Scatter Plot

```
heights = [152, 159, 165, 173, 181]
weights = [54, 57, 66, 69, 82]

plt.scatter(heights, weights, color='skyblue')
plt.xlabel("Height (cm)")
plt.ylabel("Weight (kg)")
plt.title("Height vs Weight")
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```

The scatter plot shows a general upward trend without forcing the points onto a line. This made the difference between line plots and scatter plots more intuitive.

Closing Note

Overall, these assignments helped me build a solid foundation for both probability reasoning and basic data analysis using Python. The focus throughout was understanding and correctness, rather than over-optimizing or over-formatting.