University of Tehran

Statistical Inference code Report

Hadiseh Mesbah
Student ID: 810102253

November 8, 2023

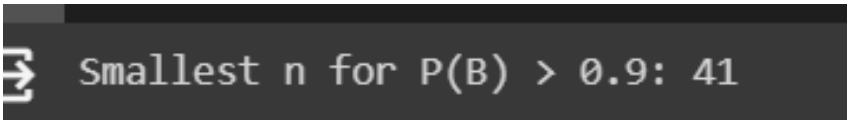

Figure 1: output for part 5 of problem 5

Figure 2: output for part 7 of problem 5

Problem 5

Part 5:

To see the result for this part, refer to Figure 1.

Part 5:

To see the result for this part, refer to Figure 2.

Problem 10

Part 1

To generate random data from various distributions and plot them in different graphs, you can use Python with libraries such as NumPy and Seaborn. In Figure 3, you can see the results for this part of the experiment.

In the code, we first generate random data for each distribution using NumPy's random functions. Then, we use Seaborn's distplot to create and display the distribution plots. The (hist=False) parameter is used to remove the histogram and only show the kernel density estimate. Finally, we say all the plots in a 2x3 grid using Matplotlib.

Part 2:

If you generate the mean values of many random samples from different distributions and then plot the distribution of those mean values, you would expect the sample mean distribution to follow an approximately normal distribution, regardless of the original distribution you are sampling from. This phenomenon is known as the Central Limit Theorem (CLT). In Figure 4, you can see the results for this part of the experiment.

In the code, we calculate the means of the samples from each distribution over multiple iterations and then plot the distribution of these sample means. You will notice that the sample mean distributions for all original distributions tend to resemble a normal distribution, demonstrating the Central Limit Theorem in action.

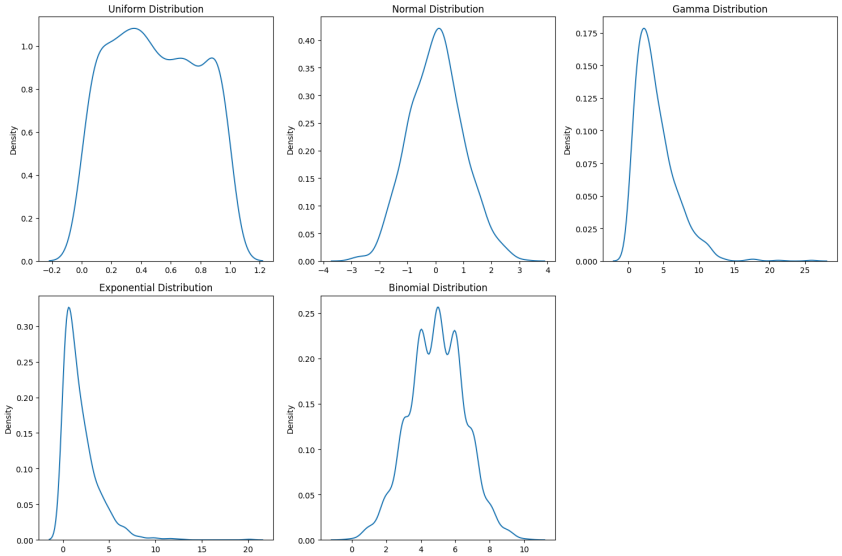

Figure 3: 1) Uniform,2) Normal, 3) Gamma, 4) Exponential, and 5) Binomal

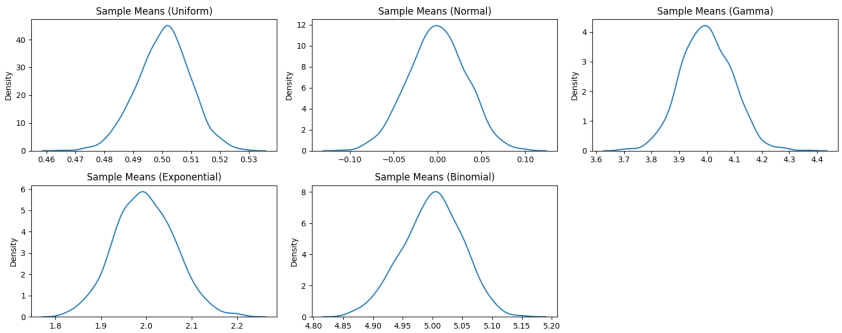

Figure 4: mean values over all iterations

Part 3:

When I wrote this, my code was on Colab, so I imported my data from Google Drive to read it.

Part 4:

I write lots of comments in code to explain what I did, but this is a summary:

First, when we look at the code, we can see some data are invalid and defined with ?. We need to replace them with NaN and count them.

Secondly, fill in missing data of normalized losses, horsepower, peak-rpm, bore, and stroke with the respective column mean. Delete data with a price tag, but the value is NaN (we can use the median, but I think this label is essential, and the correct value is necessary.) Then, fill in missing data in the "Number of doors" category with the mode of the column, i.e., "Four". If we find the out-of-range data, we must delete them, like a negative number for doors. But I did not find any.

Part 5:

1. `symboling`: The insurance risk rating of the vehicle.
2. `normalized-losses`: Normalized losses in the event of an accident.
3. `make`: Manufacturer or make of the automobile.
4. `fuel-type`: Type of fuel used, such as "gas" or "diesel."
5. `aspiration`: Whether the vehicle has a naturally aspirated or turbocharged engine.
6. `num-of-doors`: Number of doors on the vehicle.
7. `body-style`: Body style of the automobile, like "sedan" or "hatchback."
8. `drive-wheels`: Type of drive wheels - e.g. "4wd" for four-wheel drive or "fwd" for front-wheel drive.
9. `engine-location`: The location of the engine is usually at the front of the vehicle.
10. `wheel-base`: Measurement of the wheelbase of the vehicle.
11. `length`: Length of the vehicle.
12. `width`: Width of the vehicle.
13. `height`: Height of the vehicle.
14. `curb-weight`: Weight of the vehicle without passengers or cargo.
15. `engine-type`: Type of engine, such as "ohc" (overhead camshaft) or others.
16. `num-of-cylinders`: Number of cylinders in the engine, e.g., "four" or "six."
17. `engine-size`: Engine size in cubic centimeters (cc).

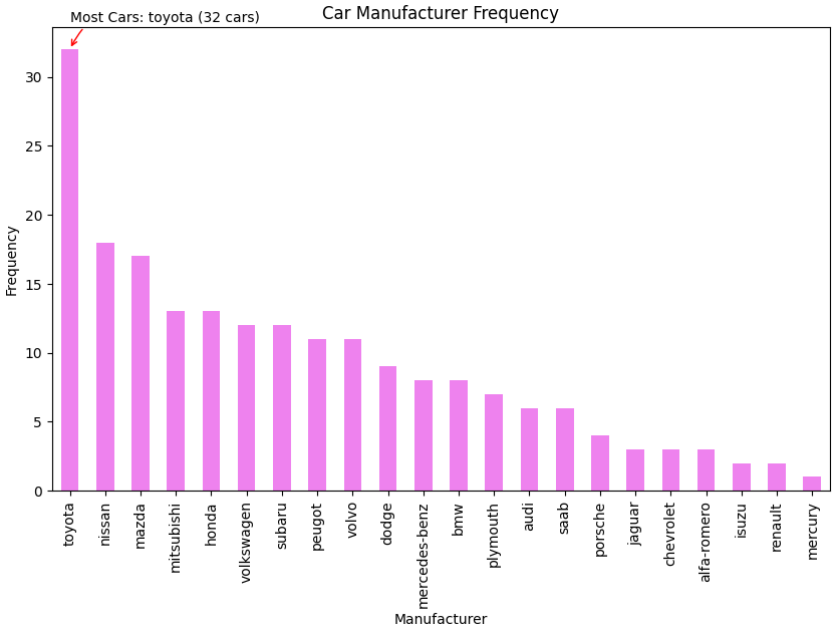

Figure 5: the frequency of each car manufacture

18. `fuel-system`: Fuel delivery system, e.g., "mpfi" (multi-port fuel injection).
19. `bore`: Diameter of the engine's cylinders.
20. `stroke`: Length of the piston stroke in the engine.
21. `compression-ratio`: Compression ratio of the engine.
22. `horsepower`: Horsepower rating of the engine.
23. `peak-rpm`: Engine's peak RPM (revolutions per minute).
24. `city-mpg`: Vehicle's miles per gallon (mpg) in city driving conditions.
25. `highway-mpg`: Vehicle's miles per gallon (mpg) on the highway.
26. `price`: Price of the vehicle.

I found some data on wheelbase and normalized losses that were out of range compared to other data, but when I researched it, I found that it was normal in vehicles.

Part 6:

In Figure 5, you can see the results for this part of the experiment.

Part 7:

I think the Standard Deviation is a good measure for finding and evaluating the dispersion of a dataset. A larger standard deviation indicates greater variability in the dataset, while a low standard deviation suggests a more concentrated distribution.

- **Skewness**:

  - Positive skewness (right-skewed) suggests that the data is skewed to the right, meaning there are more data points on the left side and a long tail on the right.
  - Negative skewness (left-skewed) suggests the opposite.
  - A skewness value near 0 indicates a roughly symmetric distribution.

- **Kurtosis**:
  - Positive kurtosis indicates heavy tails and more outliers.
  - Negative kurtosis suggests light tails and fewer outliers.
  - A kurtosis value of 3 (excess kurtosis) is often used as a reference for a normal distribution. Distributions with kurtosis greater than 3 have heavier tails, while those with kurtosis less than 3 have lighter tails.

I calculated these values for all numerical rows:
Column: wheel-base

- Standard Deviation: 6.021775685025571
- Skewness: 1.042513612401581
- Kurtosis: 0.9632757242326453

Column: length

- Standard Deviation: 12.33728852655518
- Skewness: 0.15481031885453517
- Kurtosis: -0.11001300115343327

Column: width

- Standard Deviation: 2.145203852687183
- Skewness: 0.8973753485201392
- Kurtosis: 0.6566140918303747

Column: height

- Standard Deviation: 2.4435219699049036
- Skewness: 0.06265991683394276
- Kurtosis: -0.46218755571934844

Column: curb-weight

- Standard Deviation: 520.6802035016387
- Skewness: 0.676402180083416
- Kurtosis: -0.07094187922836692

Column: engine-size

- Standard Deviation: 41.64269343817984
- Skewness: 1.9333748457840114
- Kurtosis: 5.148029693803652

Column: bore

- Standard Deviation: 0.2708437032300329
- Skewness: 0.020062526293150085
- Kurtosis: -0.7951494836889594

Column: stroke

- Standard Deviation: 0.31359700827909676
- Skewness: -0.6847268327182717
- Kurtosis: 2.0926696877327755

Column: compression-ratio

- Standard Deviation: 3.972040321863298
- Skewness: 2.5917196238579114
- Kurtosis: 5.077161302538663

Column: horsepower

- Standard Deviation: 39.51921906205876
- Skewness: 1.3877040597952843
- Kurtosis: 2.5845263707251

Column: peak-rpm

- Standard Deviation: 476.97909458533826
- Skewness: 0.07307436743023064
- Kurtosis: 0.055544363668628804

Column: city-mpg

- Standard Deviation: 6.542141653001622
- Skewness: 0.6588377533622138
- Kurtosis: 0.5355048032059506

Column: highway-mpg

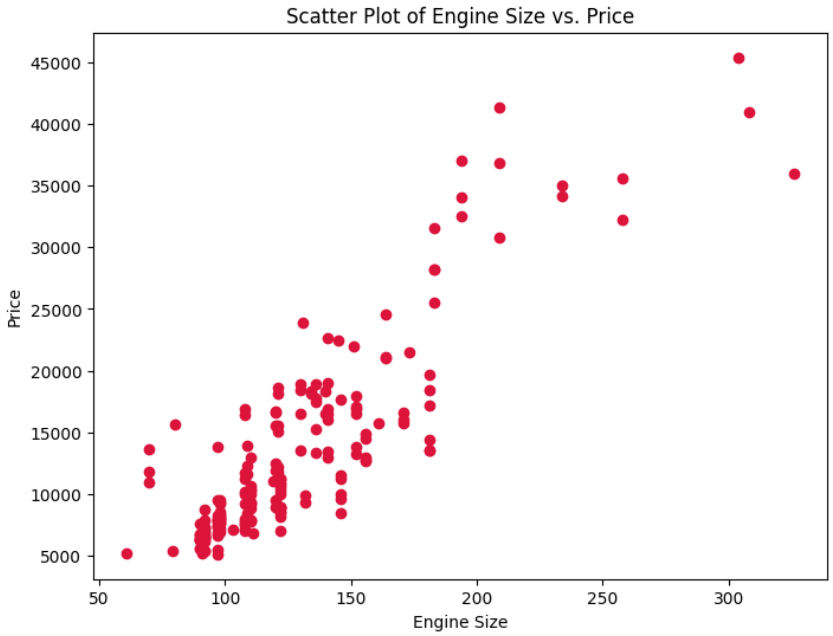

Figure 6: scatter between engine size and price value

- Standard Deviation: 6.886443130941824
- Skewness: 0.5360379305163596
- Kurtosis: 0.40028379176220774

Column: price

- Standard Deviation: 7868.76821236424
- Skewness: 1.8139271903574734
- Kurtosis: 3.2438386421315215

Part 8:

The scatter plot will display how the "engine-size" and "price" values are distributed relative to each other. If the points on the scatter plot show a clear pattern or trend, it suggests an association between the two factors. For example, a positive slope in the scatter plot indicates a positive association, where larger engine sizes tend to have higher prices. In this case, we can observe a generally upward trend in the points, which suggests a positive association between engine size and price. In Figure 6, you can see the results for this part of the experiment.

Part 9:

I drew this plot for 'length,' 'wheelbase,' 'highway-mpg,' and 'city-mpg.' The resulting pair plot will display scatterplots, histograms, and kernel density estimates for these two specific variables, allowing you to analyze their relationships.

- **Scatterplots**: The scatterplots will show how 'highway-mpg' and 'city-mpg' relate to each other. Look for the following patterns:
  - Positive Correlation: If the points form an upward-sloping pattern, it indicates a positive correlation, meaning that as 'wheelbase' increases, 'length' also tends to increase.
  - Negative Correlation: If the points form a downward-sloping pattern, it indicates a negative correlation, meaning that as 'highway-mpg' increases, 'horsepower' tends to decrease.
  - No Correlation: If the points are scattered with no apparent pattern, it suggests little to no correlation between the variables, like 'wheelbase' and 'horsepower.'
- **Histograms**: The histograms along the diagonal of the pair plot show the distribution of each variable individually. Analyze the shapes of the histograms to understand the distribution of 'highway-mpg' and 'city-mpg' values. For example, we can assess if the data is normally distributed, skewed, or exhibits any other characteristic.
- **Kernel Density Estimates**: The plots above the diagonal show kernel density estimates. These provide smoothed representations of the data distribution.
- **Outliers**: Look for any outliers in the scatterplots or histograms. Outliers may indicate unusual data points that deviate significantly from the primary trend. For example, as 'highway-mpg' increases, 'horsepower' has one of these outliers.
- **Density Concentrations**: Identify high-density areas in the scatterplots or kernel density estimates. These regions may indicate where most of the data points are concentrated.

  In Figure 7, you can see the results for this part of the experiment.

Part 10:

The heatmap will visually represent the correlation between numerical columns, with colors indicating the strength and direction of the correlations. Positive correlations are shown in warmer colors, while negative correlations are represented in cooler colors. This allows you to identify relationships between variables in your dataset quickly. In Figure 8, you can see the results for this part of the experiment.

Part 11:

This is a plot for body style and price together. In Figure 9, you can see the results for this part of the experiment.

Percentiles:

body-style	25%	75%
zero: convertible	14246.25	30709.25
one: hardtop	9341.50	32903.00
two: hatchback	6518.75	12044.75
three: sedan	8010.75	17890.00
four: wagon	8013.00	15750.00

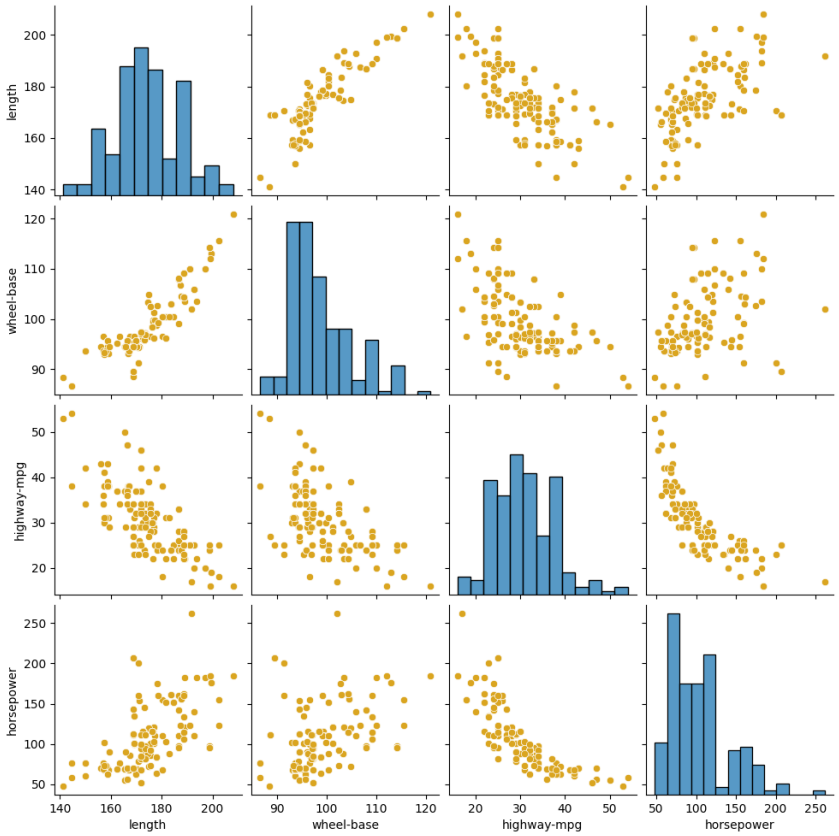

Figure 7: Multivariate analysis

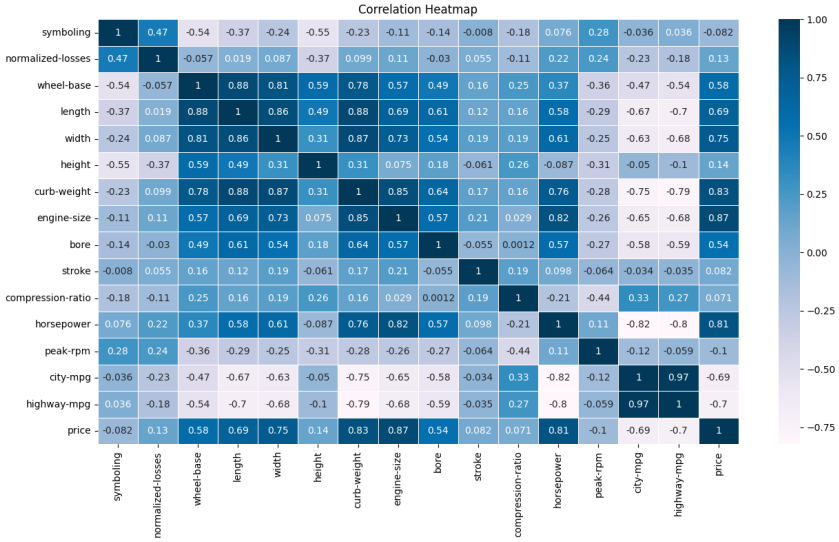

Figure 8: Heat map

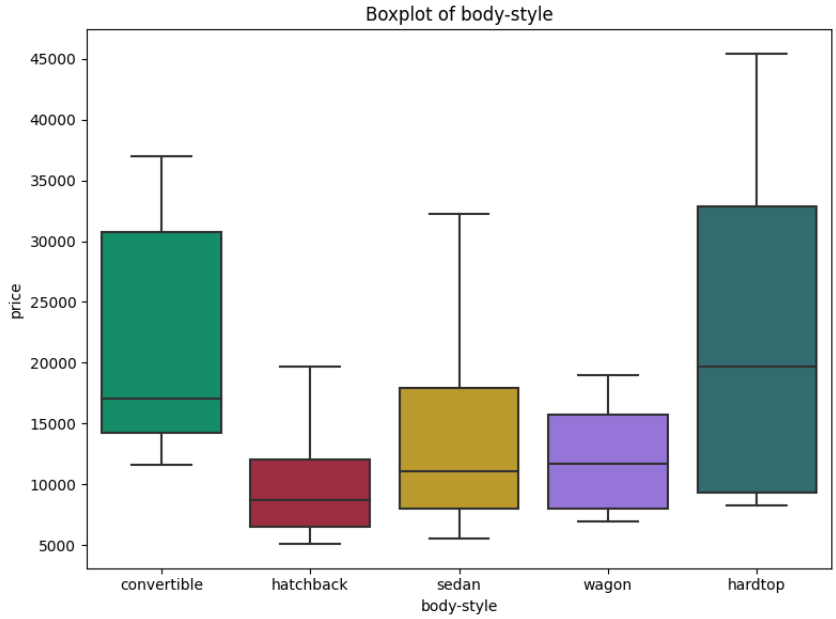

Figure 9: Box plot of body style and price together

IQR (Interquartile Range): 16463.0
Lower Whisker: -10448.25
Upper Whisker: 55403.75

Problem 12

Task 1:

This program generates random points within a square with a side length of one and checks whether each point is inside the quarter-circle (1/4 of a circle with a radius of 1). The ratio of points inside the quarter-circle to the total points should converge to $\frac{\pi}{4}$. By multiplying this ratio by 4, we can estimate the value of π. Here are the steps:

1. Define a square with a side length of 2 units, centered at $(0, 0)$.
2. Draw a quarter of a circle inside the square with a radius of 1 unit.
3. Randomly generate points within the square with x and y coordinates between -1 and 1.
4. Calculate the distance for each point, and if it's less than or equal to 1, consider it inside the circle.
5. The ratio of the number of points inside the circle to the total number of generated points is the estimate of the ratio of the areas of the quarter-circle to the square.
6. The estimated value of π is 4 times the ratio calculated in step 5.

Mathematically, you can express the estimation as follows:

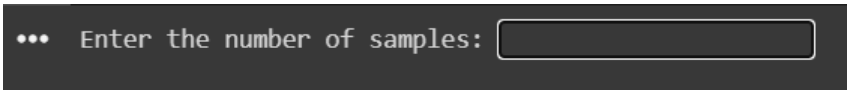

Figure 10: Output from user

```
Enter the number of samples: 500
Estimated Pi: 3.064
Actual Pi: 3.141592653589793
Difference: 0.07759265358979306
```

Figure 11: Estimated Pi number for 500 samples

```
Enter the number of samples: 10000
Estimated Pi: 3.1356
Actual Pi: 3.141592653589793
Difference: 0.005992653589792951
```

Figure 12: Estimated Pi number for 10000 samples

$$\pi \approx 4 \times \left(\frac{\text{Number of points inside the circle}}{\text{Total number of points generated}} \right).$$

When you run the program, it will ask you for the number of samples: (Figure10)

```
Enter the number of samples:
```

Then, the program will calculate this parameter: (In Figure 11 and Figure 12, you can see the results for this part of the experiment.)

```
Estimated value of pi: 3.14159265359
```

In Figure 13, you can see the Monte Carlo Estimation of Pi in the sample of 10000.

Task 2:

We can get a more accurate estimate with a larger sample size. In Figure 14, you can see the results for this part of the experiment.
In Figure 15, you can use the Monte Carlo method to estimate the difference between Pi and the actual value, and as the number of random points increases, the difference approaches zero. .

Problem 13:

Task 1:

Simulates a gambling scenario where a gambler starts with an initial stake and aims to reach a target amount by repeatedly making bets.

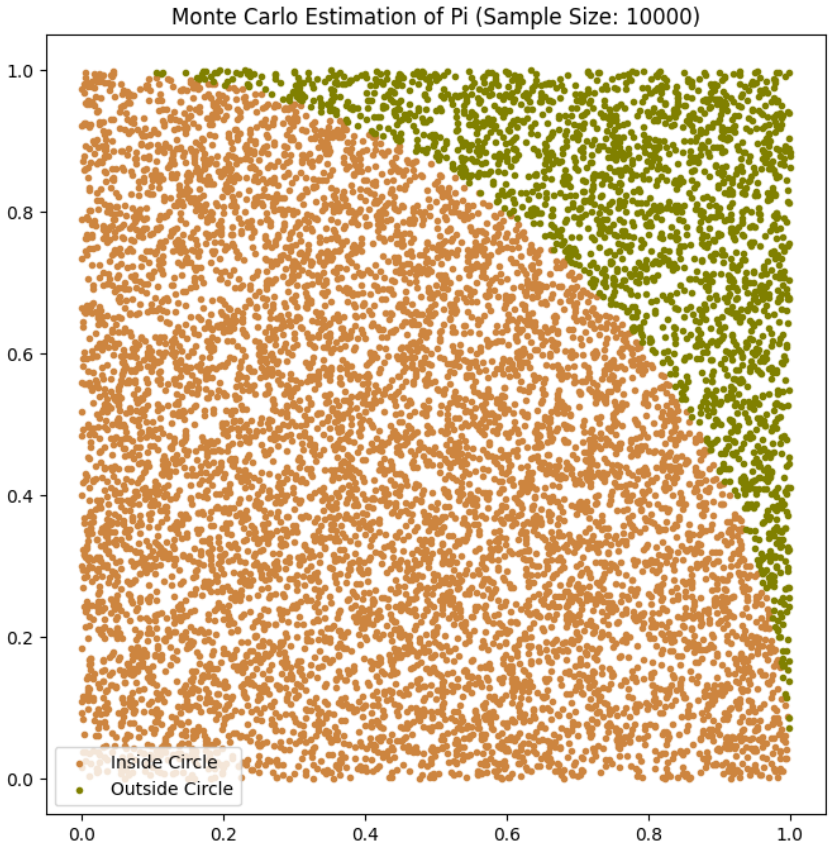

Figure 13: Monte Carlo Estimation of Pi visualization

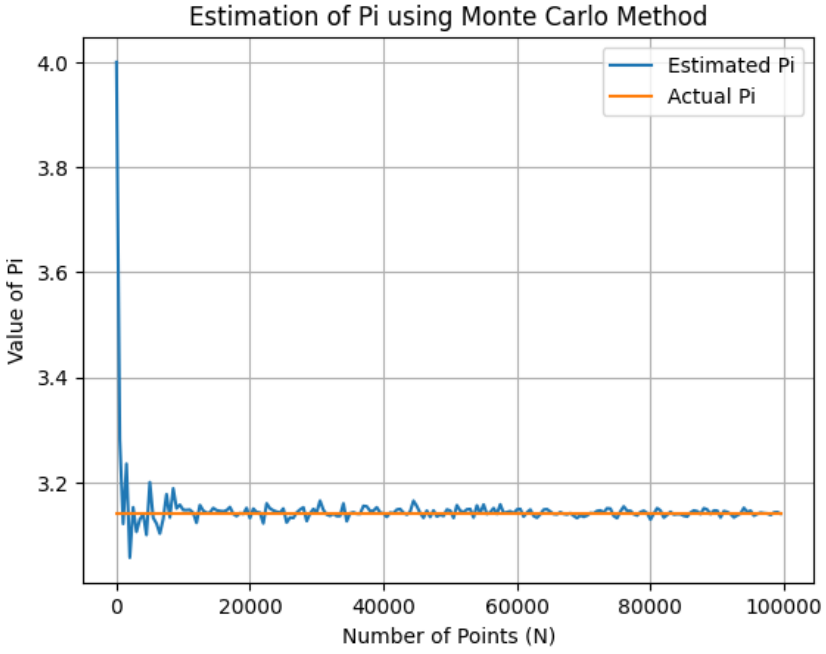

Figure 14: Estimation of Pi using Monte Carlo Method versus Actual Pi

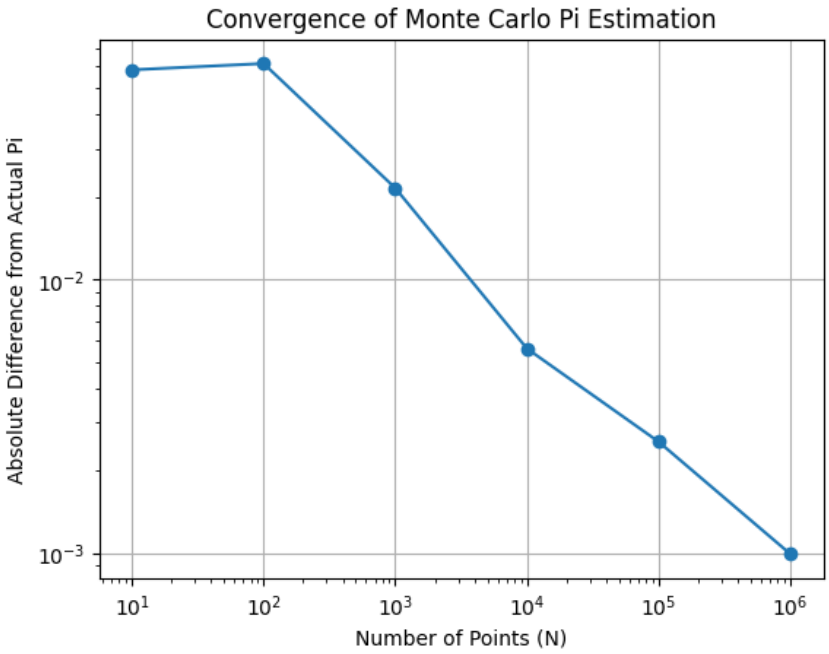

Figure 15: Absolute Difference from Pi using Monte Carlo Method versus Actual Pi

First, the code prompts the user to input parameters for the gambling scenario, including the initial stake, target amount, bet amount, win probability, and the maximum number of rounds. The code includes visualization using Matplotlib. It displays a line graph showing the stake history over time for a single simulation. Additionally, it plots the mean success probabilities for different numbers of iterations, providing insights into the likelihood of reaching the target amount before going broke as the number of simulations increases for the user-inputted parameters.

We have both experienced victories and defeats. In Figure 16 and Figure 17, you can see the results for this part of the experiment.

This plot displays the mean probability of reaching the target amount before going broke based on the inputted program parameters. In Figure 18, you can see the results for this part of the experiment.

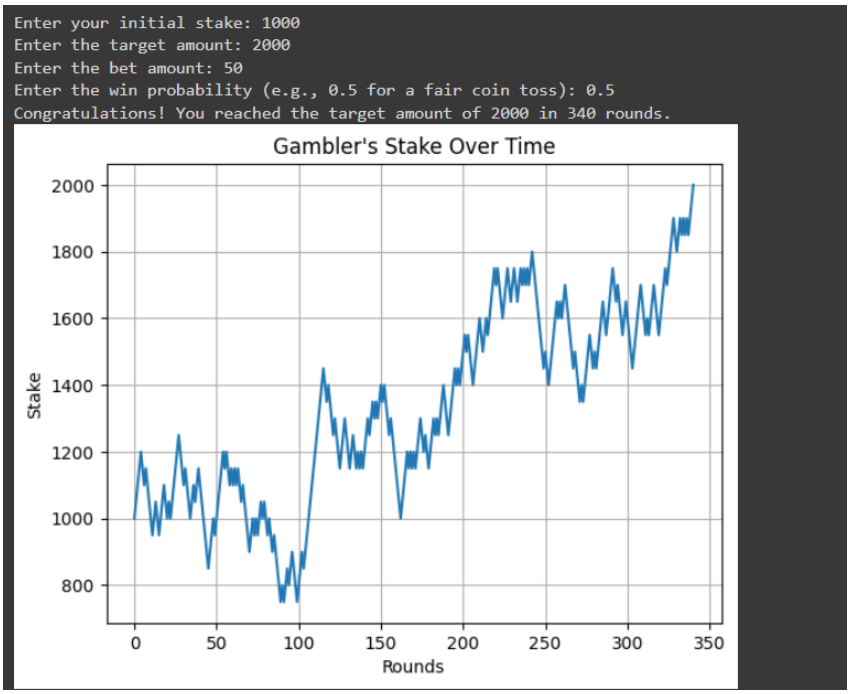

Figure 16: experienced victories for a situation

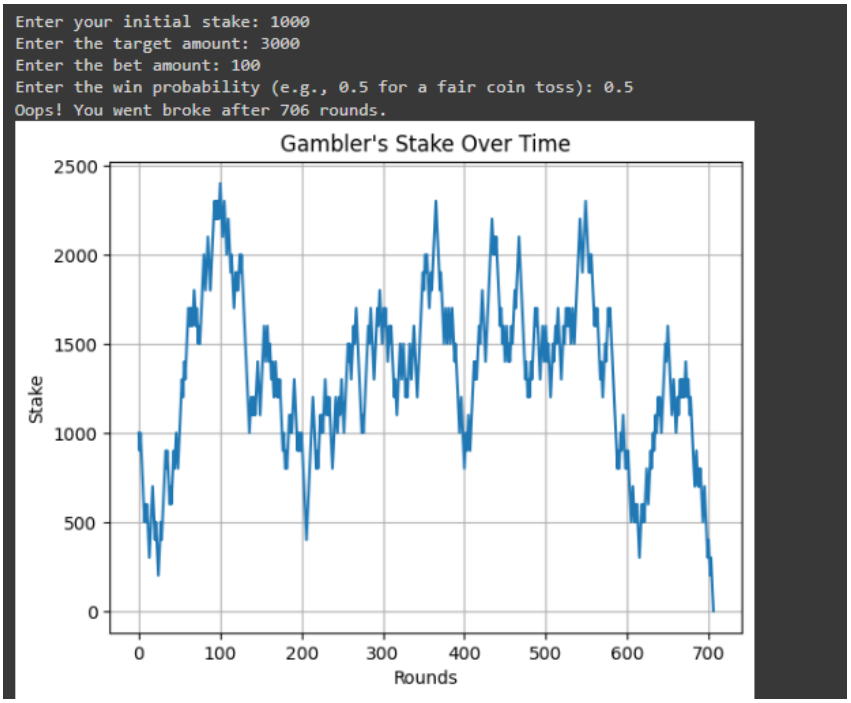

Figure 17: experienced defeat for a situation

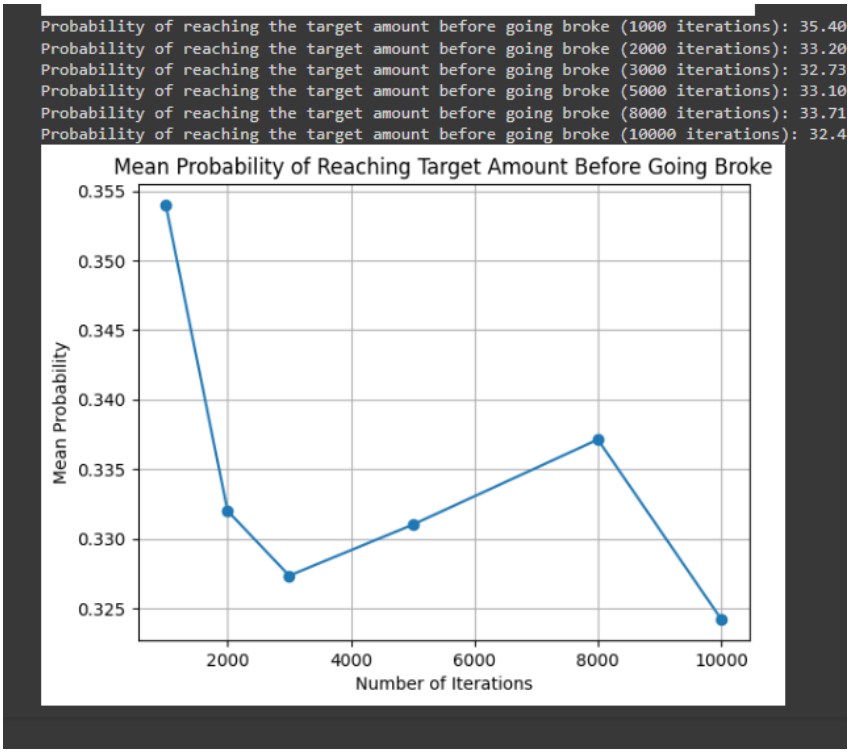

Figure 18: Mean probability of reaching the target amount before going broke