Learning Curves
How do Data Imbalances affect the Learning Curves using Nearest Mean Model?

Jia Jie Feng
Supervisors: Dr. Tom Viering, Mr. Taylan Turan

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
January 28, 2024

Name of the student: Jia Jie Feng
Final project course: CSE3000 Research Project
Thesis Committee: Responsible Professor, Supervisor, Examinar

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

This research investigates the impact of data imbalances on the learning curve using the nearest mean model. Learning curves are useful to represent the performance of the model as the training size increases. Imbalanced data sets are often encountered in real-life scenarios and pose challenges to standard classifier models impacting their performance. Thus, the research question is "How do data imbalances affect the learning curves using the nearest mean model?". To answer the question, an experiment is conducted using data from a multivariate Gaussian distribution to sample varying levels of imbalances. The imbalance ratio explored is [0.1, 0.2, 0.3, 0.4, 0.5], representing the percentage of the dataset that consists of the minority class.

The findings indicated that as the data becomes more imbalanced, the learning curves reach the accuracy plateau at a later rate. The analysis of the curve parameter which follows the logistic function suggests that imbalances have an impact on the maximum achievable accuracy and rightward shift of the curves. However, the maximum achievable accuracy is non-significant and the shape of the curves remains similar. Additionally, false negatives have a significant impact on the learning curves.

1 Introduction

Learning curves are visual representations that show the progression of a learning model's performance as the train size increases [1]. It serves as a tool for model selection, forecasting the impact of additional training data, and mitigating the computational complexity associated with model training and hyperparameter tuning [2].

This report explores the impact of the dataset's characteristics on the learning curve. Namely the class imbalances in the datasets. Imbalance datasets are characterized by skewed class proportions where classes that make up a large portion of the data are called the majority and the smaller portions are called the minority [3]. Imbalance data negatively impacts the performance of most standard classifier learning models which assume relatively balanced distribution [4]. Unfortunately, real-world data collection typically involves imbalanced datasets, and there are instances, especially in the medical industry, where data inadequacy compounds the challenges of imbalanced datasets [5]. Given the prevalent issue of imbalanced datasets, the research question for the paper is "How do data imbalances affect the learning curves using the nearest mean model?".

This paper is structured into the following sections: Related Work(Section 2), where the existing literature is examined; Methodology (Section 3), providing details on the methodologies; Results and Discussions(Section 4), presenting findings and the analysis; Conclusion and Future Work(Section 5), summarizing important insights and future suggestions; Responsible Research(Section 6), discussing the reproducibility and ethics of the paper.

2 Related Works

Banko et al. concluded that classification models can benefit greatly from larger training sizes when dealing with imbalanced data, although they acknowledge that there are other factors such as dimensionality and feature complexity [6]. Their experiment based on millions of data points from a word corpus, utilized the accuracy curve for evaluation. In a complementary study, Samir Al Stouhi et al. extended the conclusion by Banko et Al.using non-language models [7]. They have concluded that the choice of classification model is less important than the size of the training data.

Zheng et al. investigated the relationship between varying performance in machine learning models and imbalance ratio [8]. Their findings suggest that increasing the imbalance ratio results in a decrease in performance by the models. However, they observed that there were results where it contradicted the conclusion but speculated that oversampling caused new data points to not follow the original distribution. They defined the imbalance ratio using $\frac{NumberOfMajority}{NumberOfMinority}$, the value rounded down and decrements the ratio by $\frac{1}{10}$ at a time to produce varying levels of imbalance in the data set. To achieve the imbalance of data required according to the ratio, they used three methods;

SMOTE, undersampling, and hybrid. The experiment involves eight machine-learning models for the experiment, evaluated using AFG, AUC, F1-score and G-mean. AFG is described as the arithmetic mean of AUC. However, the experiment did not take into consideration the difference in sample sizes when changing the imbalances in the data set.

Our paper seeks to validate the conclusions made by Banko et al. and Samir Al Stouhi which state that training size has a bigger impact on performance than the imbalance ratio and the inverse relation of imbalance ratio and performance by Zheng et al [6][7][8]. Our experiment shares similarities with the experiment conducted by Zheng et al but diverges in the choice of imbalance ratios, methods to produce the desired imbalance ratio and consideration for sample size across imbalanced data sets. Furthermore, the experiment is evaluated using the accuracy curve, the area under the accuracy curve, multivariate Gaussian distribution, plateau analysis, parameter analysis, and false positive and negative curves. It will not be evaluated using AFG, F0-score and G-mean.

3 Methodology

This section describes how the experiment was conducted to investigate the impact of class imbalances on the learning curve using the nearest-mean model. It describes the data set used and how the results were produced with the model to create learning curves for analysis.

3.1 Definitions

The ratio of the class imbalance used for the experiment is $[0.1, 0.2, 0.3, 0.4, 0.5]$. The value represents the percentage of the dataset that consists of the minority. 0.5 indicates a balanced data set, while a ratio of 0.1 implies that 10% of the dataset consists of the minority class. There are two classes in the experiment; class 0 (C_0) and class 1(C_1). Class 0 is the majority, the negative class and class 1 is the minority, the positive class.

3.2 Data Set

The source of the data set used for the experiment is sampled from a multivariate Gaussian distribution. For class 0, the parameters are defined by $\mu_0 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$ and $\Sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. For class 1, the parameters are $\mu_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Figure 1 visually shows the distribution that will be used for the experiment.

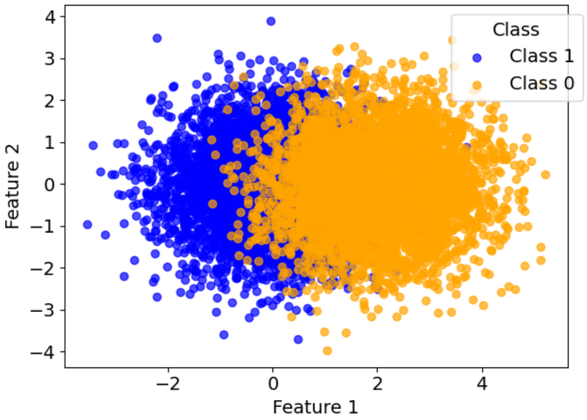


Figure 1: Multivariate Gaussian Distribution. The means are: $\mu_0 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$, $\mu_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$. The covariance is identity matrix.

3.3 Experiment

The experiment begins by sampling 100 data points from the multivariate distribution based on the imbalance ratio. The test set consists of a balanced data set with 10000 samples from the same distribution. The nearest mean model is trained using specific subset sizes, defined as the following: [10, 20, 30, 40, 50, 60, 70, 80, 100]. The experiment is repeated 1000 repetitions to ensure the reliability of the results. After training, the model's accuracy metrics on the test set are averaged and learning curves are produced.

4 Results and Discussion

In this section, the results of the experiment are presented and analysed, focusing on the shapes of the learning curves, the accuracy plateau, performance disparities across imbalance ratios, and the false positive and negative rates.

4.1 Performance

The performance of the nearest mean model decreases with the increase of imbalances in the data set. Illustrated in Figure 2, the learning curves across the imbalance ratio. A notable performance gap is observed in training size 10. However, as the training size increases, all imbalance curves progressively approach the performance level of the balanced dataset. This indicates that with sufficient data irrespective of the imbalance ratios, the model can perform relatively well. To quantify the model's performance throughout the training process, the area under the curves was calculated, this is shown in Figure 4. It shows that as the level of imbalances in the data set increases, the overall performance during the training decreases. This suggests the significant role of the training size on the performance of the model regardless of the imbalance ratios specified in this paper.

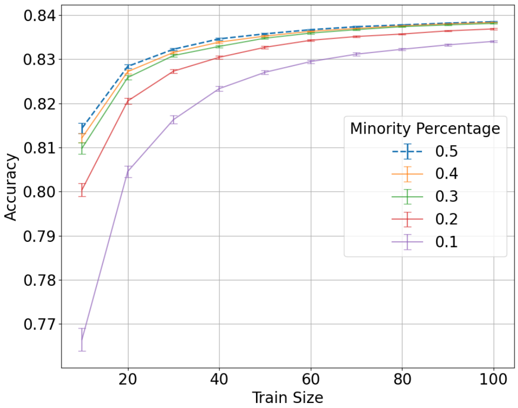


Figure 2: Comparison of Learning Curves of various imbalance ratios. The performance of the model increases as the level of imbalance in the data decreases

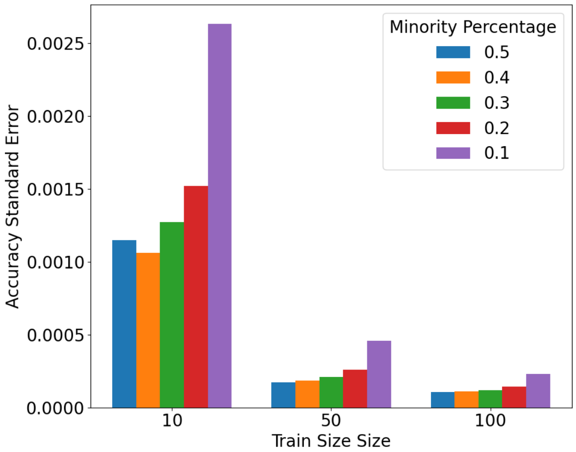


Figure 3: Partial representation of the standard error during training. The standard errors are relatively non-significant across the imbalance ratios. As the train size increases, the standard error increases as the imbalance increases

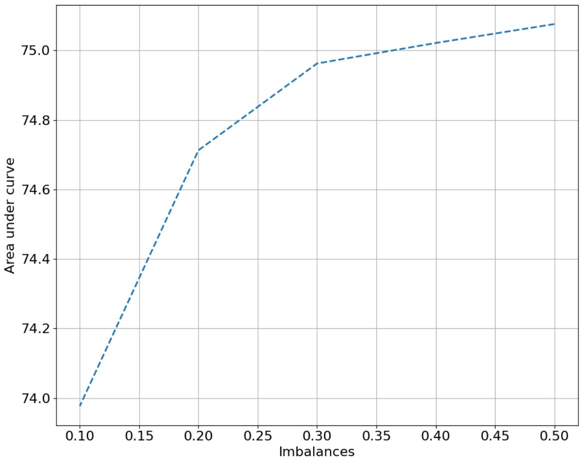


Figure 4: Area under the curve across the imbalance ratio. It indicates that as the imbalances of the data set increases, the overall training performance decreases

4.2 Shape of the Curves

An analysis of the shape of the learning curves was conducted by fitting the logistic function onto the existing curve, providing approximations for parameters a, b, and c. The logistic function is defined by $f(x) = \frac{a}{1+e^{-b\cdot(x-c)}}$ and parameters are defined as the following: a is the maximum achievable accuracy; b is growth rate; c is the midpoint of the logistic curve; x is the train size. Figure 5 shows the curve fitting result on the balanced curve. Despite a few missed points, the relationship among the fitted curves mirrors that of the original curves, as depicted in 6. Additionally, the square error was calculated between the original and fitted curves as shown in 7. The square error indicates that the fitted curves are less accurate when performed on curves resulting in more imbalance data but the errors are not significant, indicating that the fitted curves are relatively reliable. Therefore, the approximated values of the parameters can provide valuable insights into the shape of the curves.

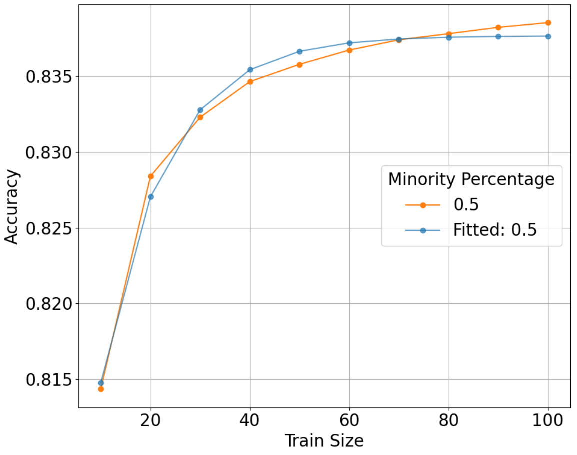


Figure 5: Fitted Curve vs Original Curve of Balanced Data Set. The fitted curve did not capture every data point but the squared error is minimal. The square error for this curve is 5.01335×10^{-06}.

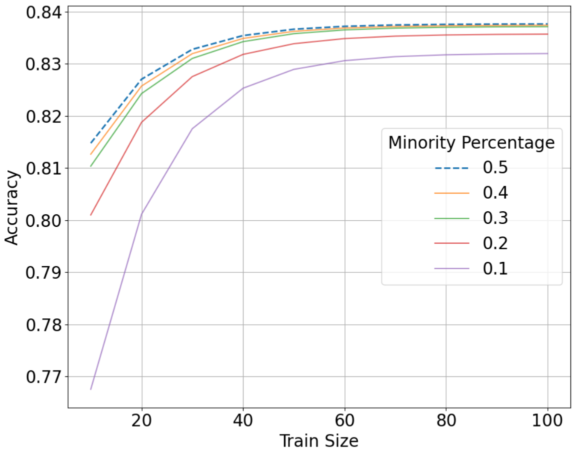


Figure 6: Fitted curves of all original curves. The fitted curves seem to capture similar patterns from the original curves.

The estimated parameters obtained from curve fitting were compared across different imbalance ratios, this is presented in Figure 8, 9, and 10 which shows a, b and c, respectively.

Parameter a decreases as the imbalances in the data set increase. This indicates that the maximum accuracy achievable by the fitted curve decreases with increasing class imbalances. No observable relationship can be identified for parameter b. Parameter c increases as the imbalances increase, implying the rightward shift of the function with increasing imbalance.

Further analysis reveals that while parameters a and b have no significant differences between the imbalance ratios, parameter c displays significant disparities. This suggests that the impact of imbalances on the learning curves majorly affects the rightward shift of the fitting function and has little impact on the maximum achievable accuracy and growth rate.

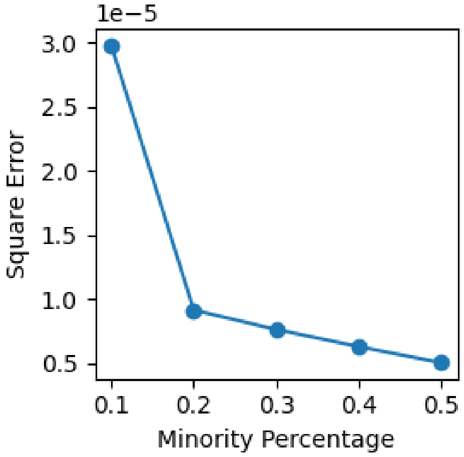


Figure 7: The square errors of the fitted curve and the original curves. The square error increases as the curves are made from more imbalanced datasets. The errors are non-significant, indicating that it is relatively reliable.

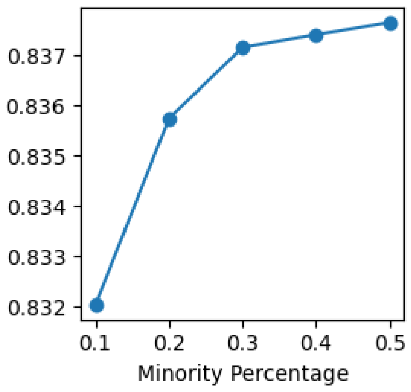


Figure 8: Parameter a comparison, maximum accuracy the fitted curve can obtain. Increased imbalances result in a decrease of a. However, the difference is non-significant.

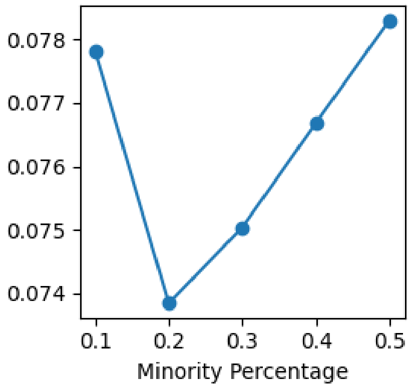


Figure 9: Parameter b comparison, the steepness of the fitted curve. No conclusion can be made from the figure. However, the differences between the imbalance ratios are non-significant.

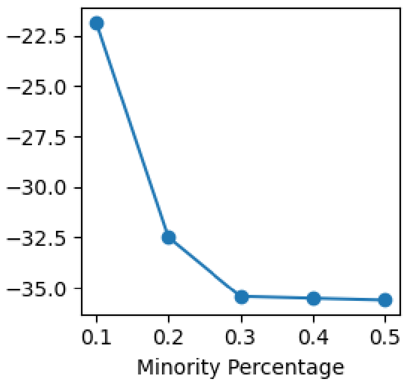


Figure 10: Parameter c comparison, the horizontal shift of the logistic function. As the imbalances in data increase, the logistic function shifts to the right. The differences are significant.

4.3 Plateau

The plateau of the curves provides insights into when increasing the training size yields no further significant advancement in accuracy. The derivatives of the curves were calculated and a threshold of 0.0001 was applied to determine the plateau point. Figure 11 shows the change in derivatives along the curves. Curves with imbalance ratios 0.5, 0.4 and 0.3 plateau at train size 50, while curves with imbalance ratios 0.2 and 0.1 plateau at train sizes 60 and 80 respectively.The observation implies that as the data set becomes more imbalanced, more training samples are required to reach their plateau. The values of the derivatives for the curves can be found in Appendix **TODO Appendix**

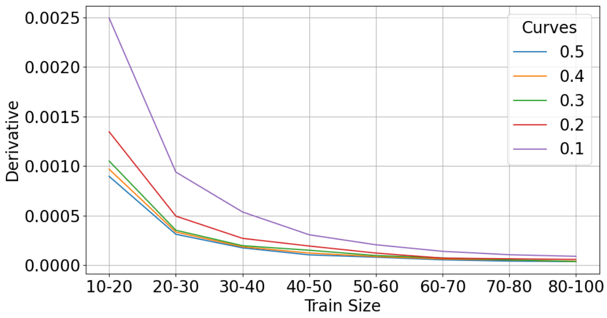


Figure 11: The derivatives of the learning curves. With a threshold of 0.0001, curves 0.5, 0.4 and 0.3 reaches a plateau after 50 train size. Curves 0.2 and 0.1 reach the plateau after 60 and 80 train sizes respectively.

4.4 False Positives and Negatives

The false positive and negative rates of the curve provide details on how the model handles imbalanced data sets and how it impacts the learning curve dynamics. Figure 12 and 13 illustrate the false positive rate and false negative rate across the imbalance ratios, respectively. The positive class signifies the minority, while the negative class signifies the majority. The false positive rate curves show increased noise as imbalances in the data increase. However, the curves behaved as expected since there are more majority instances in more imbalanced data sets, and the rate of false positives is decreased. On the other hand, the false negative rate curves exhibited no visible noise and behaved as expected as higher imbalances lead to higher false negative rate. During the early training phase, the maximum difference among the false positive curves is approximately 3% whereas the false negative is approximately 12%. Toward the end of the training, the maximum difference for the false positive rate is approximately 0.5% and the false negative rate is approximately 1%. This suggests that the learning curves are more significantly impacted by the false negatives, emphasizing their importance over the false positives. As a result, the balanced learning curve performed the best.

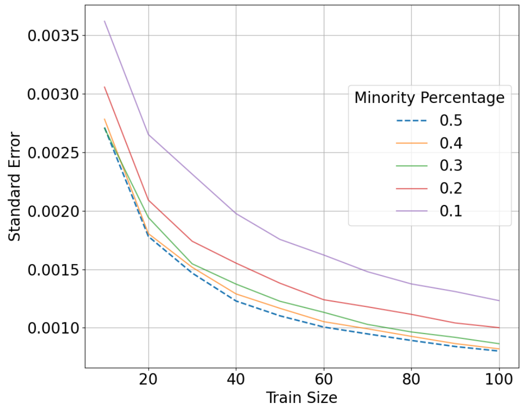


Figure 12: False Positive Rate Curves. The balanced curve showed no visible noise, but noise began to appear with increase of imbalances. However, it behaved as expected, that is lower false positive rate as imbalances decrease.

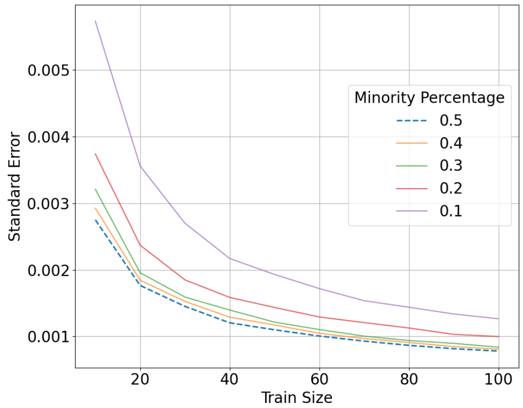


Figure 13: False Negative Rate Curves. The curves behaved as expected. The mislabelling of the minority class decreases as the curve becomes less imbalance.

4.5 Limitations

The experiment conducted is not perfect, it has some limitations which are discussed in this section. The limitations discussed are the choice of subset size and imbalance ratios.

Due to the simplicity of the data used, the analysis is only conducted with a maximum of 100 training samples, which is the main contributor to the limitation of this experiment. As a result, the choice of imbalance ratios and subset sizes for the training is limited to maintain consistency throughout the training of the model and sufficient data points for the curves. To ensure that the desired imbalance ratio is maintained throughout the training, the subset size can only be intervals of 10 and the imbalance to be intervals of 0.1. Additionally, the nearest mean model from sklearn requires that there is at least one instance of each class, contributing to the choice of the subset size and imbalance ratio intervals.

5 Conclusion and Future Works

In conclusion, this research aims to investigate the impact of data imbalances on the learning curve using the nearest mean model. The experiment uses a multivariate Gaussian distribution to generate the samples for the training of the model with various imbalances in the samples. The analysis of curve parameters using the logistic function reveals that the maximum achievable accuracy decreases with increased imbalances and shifts the curves to the right. However, the difference in achievable accuracy is not significantly big. Furthermore, the plateau analysis demonstrated that with increased imbalances, more training sample is needed to reach a plateau in accuracy. The examination of the false negative and positive rate curves suggests that false negatives have a significant impact on the learning curves. Additionally, the variability of the curves is greater with increased imbalances, though insignificant. Overall, the findings support the conclusion made by Banko et al. and Samir Al Stouhi et al., emphasizing the significance of training size over imbalance ratio in the data set. Additionally, it supports Zheng et al. findings that conclude the decrease in performance is due to increased imbalances in the data.

While the finding of this study provides valuable insights into the impact of data imbalances on the learning curve, it is of utmost importance to acknowledge that this paper is based on simple synthetic data and specific constraints imposed by the chosen model and experimental design. To ensure generalisability, future research should be done. This includes more models and diversity in data used.

6 Responsible Research

In order to ensure reproducibility, this paper provides the methodology of the experiment, including all relevant data such as the distribution, repetition size, train size, imbalance ratio and model. The experiment source code can be found at `https://gitlab.com/jiajiefeng000/research-project.git`. Additionally, various limitations of the experiment is stated.

Large language models were used during the production of this research paper. It was used to assist in ideas for writing in some sections, latex, producing code to plot the analysis of the data and debugging code. All related prompts with the large language model can be found in Appendix **TODO**.

References

[1] Julia Kagan. "What is a Learning Curve?" In: *Investopedia* (2023). URL: `https://www.investopedia.com/terms/l/learning-curve.asp`.

[2] Tom J. Viering and Marco Loog. "The Shape of Learning Curves: a Review". In: *CoRR* abs/2103.10948 (2021). URL: `https://arxiv.org/abs/2103.10948`.

[3] Google Developer. "Imbalanced Data". In: *Machine Learning* (address). URL: `https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data`.

[4] Bartosz Krawczyk. "Learning from imbalanced data: open challenges and future directions". In: *Progress in Artificial Intelligence* (2016). DOI: `10.1007/s13748-016-0094-0`. URL: `https://doi.org/10.1007/s13748-016-0094-0`.

[5] Pradeep Kumar and et al. "Classification of Imbalanced Data: Review of Methods and Applications". In: *IOP Science* (2021). URL: `https://iopscience.iop.org/article/10.1088/1757-899X/1099/1/012077/pdf#:~:text=Data%20imbalance%20occurs%20when%20sample,major%20negative%20impact%20on%20performance.`.

[6] Michele Banko and Eric Brill. "Scaling to very very large corpora for natural language disambiguation". In: *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*. 2001, pp. 26–33.

[7] Samir Al-Stouhi and Chandan K. Reddy. "Transfer Learning for Class Imbalance Problems with Inadequate Data". In: *National Library of Medicine* (2015). URL: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4929860/`.

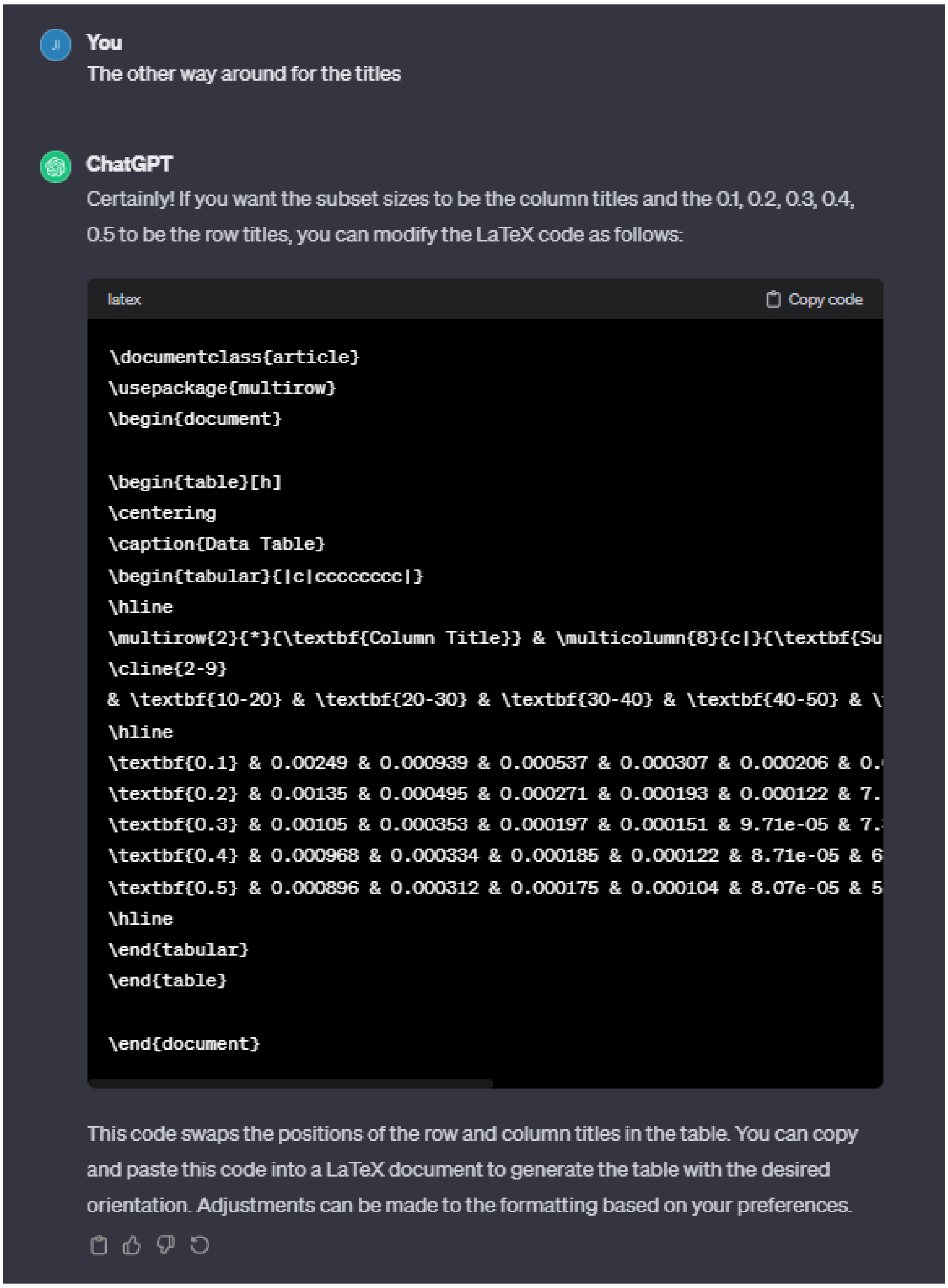
[8] Ming Zheng et al. "A Method for Analyzing the Performance Impact of Imbalanced Binary Data on Machine Learning Models". In: *Axioms* 11.11 (2022). ISSN: 2075-1680. URL: `https://www.mdpi.com/2075-1680/11/11/607`.

A Derivatives

Table 1: The derivatives of the fitted curves

	Train Size							
	10-20	**20-30**	**30-40**	**40-50**	**50-60**	**60-70**	**70-80**	**80-100**
0.1	0.00249	0.000939	0.000537	0.000307	0.000206	0.00014	0.000105	8.94e-05
0.2	0.00135	0.000495	0.000271	0.000193	0.000122	7.1e-05	6.45e-05	5.8e-05
0.3	0.00105	0.000353	0.000197	0.000151	9.71e-05	7.31e-05	5.36e-05	3.7e-05
0.4	0.000968	0.000334	0.000185	0.000122	8.71e-05	6.27e-05	5.17e-05	3.9e-05
0.5	0.000896	0.000312	0.000175	0.000104	8.07e-05	5.4e-05	4.11e-05	3.64e-05

B Large Language Model Prompts



You

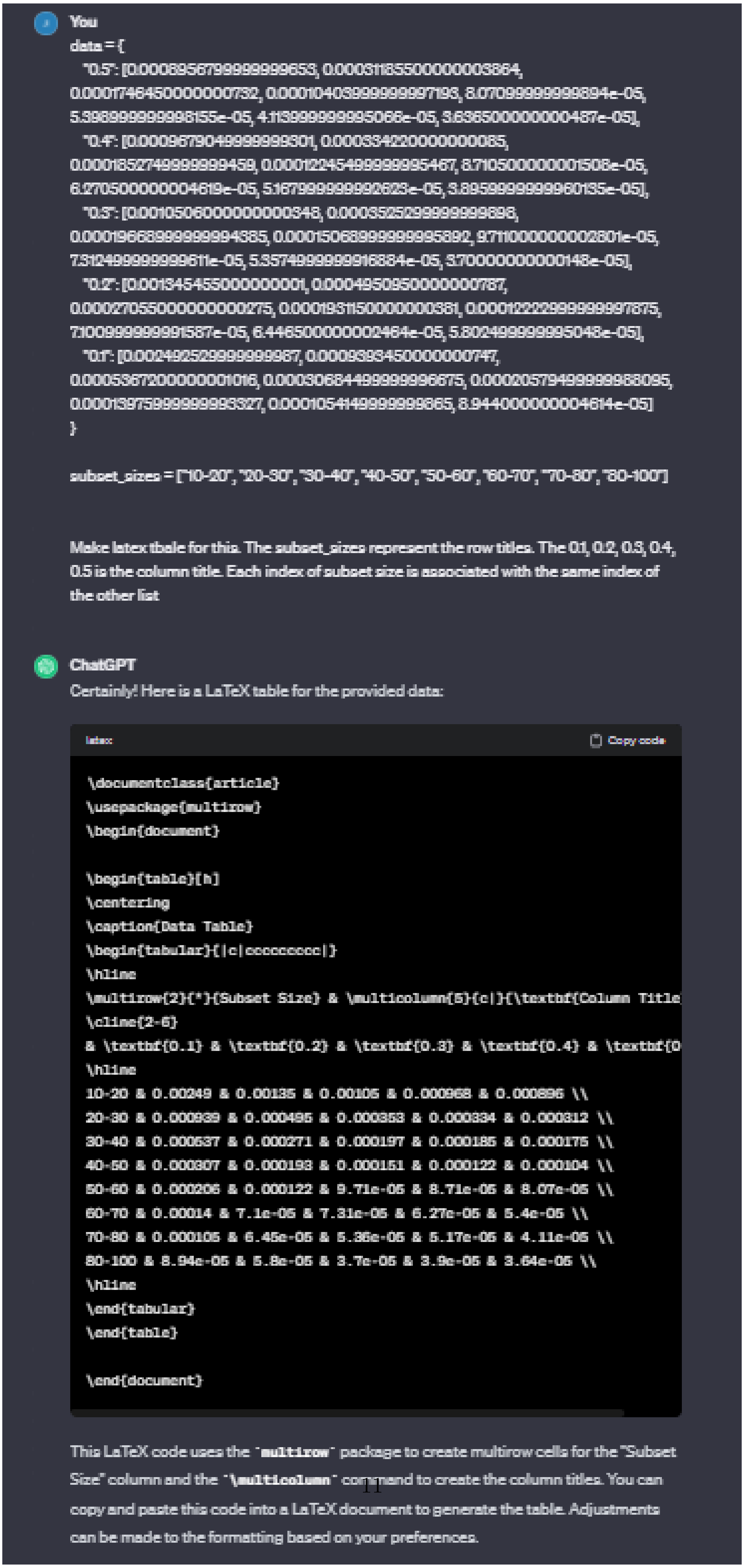
The other way around for the titles

ChatGPT

Certainly! If you want the subset sizes to be the column titles and the 0.1, 0.2, 0.3, 0.4, 0.5 to be the row titles, you can modify the LaTeX code as follows:

```latex
\documentclass{article}
\usepackage{multirow}
\begin{document}

\begin{table}[h]
\centering
\caption{Data Table}
\begin{tabular}{|c|cccccccc|}
\hline
\multirow{2}{*}{\textbf{Column Title}} & \multicolumn{8}{c|}{\textbf{Su
\cline{2-9}
& \textbf{10-20} & \textbf{20-30} & \textbf{30-40} & \textbf{40-50} & \
\hline
\textbf{0.1} & 0.00249 & 0.000939 & 0.000537 & 0.000307 & 0.000206 & 0.
\textbf{0.2} & 0.00135 & 0.000495 & 0.000271 & 0.000193 & 0.000122 & 7.
\textbf{0.3} & 0.00105 & 0.000353 & 0.000197 & 0.000151 & 9.71e-05 & 7.
\textbf{0.4} & 0.000968 & 0.000334 & 0.000185 & 0.000122 & 8.71e-05 & 6
\textbf{0.5} & 0.000896 & 0.000312 & 0.000175 & 0.000104 & 8.07e-05 & 5
\hline
\end{tabular}
\end{table}

\end{document}
```

This code swaps the positions of the row and column titles in the table. You can copy and paste this code into a LaTeX document to generate the table with the desired orientation. Adjustments can be made to the formatting based on your preferences.



You

data = {
  "0.5": [0.0008956799999999653, 0.0003118550000000003864, 0.0001746450000000732, 0.0001040399999997193, 8.0709999999999894e-05, 5.3989999999998155e-05, 4.1139999999995066e-05, 3.6365000000000487e-05],
  "0.4": [0.0009679049999999301, 0.000334220000000085, 0.0001852749999999459, 0.0001224549999995467, 8.7105000000001508e-05, 6.2705000000004619e-05, 5.1679999999992623e-05, 3.8959999999960135e-05],
  "0.3": [0.0010506000000000348, 0.0003525299999999898, 0.0001966399999994385, 0.0001506399999995392, 9.7110000000002801e-05, 7.3124999999999611e-05, 5.3574999999916384e-05, 3.70000000000148e-05],
  "0.2": [0.0013454550000000001, 0.0004950950000000787, 0.0002705500000000275, 0.0001931150000000381, 0.0001222999999997875, 7.1009999999991587e-05, 6.4465000000002464e-05, 5.8024999999995048e-05],
  "0.1": [0.0024925299999999987, 0.0009393450000000747, 0.0005367200000001016, 0.0003068449999996675, 0.0002057949999999368095, 0.0001397599999993327, 0.0001054149999999365, 8.9440000000004614e-05]
}

subset_sizes = ["10-20", "20-30", "30-40", "40-50", "50-60", "60-70", "70-80", "80-100"]

Make latex tbale for this. The subset_sizes represent the row titles. The 0.1, 0.2, 0.3, 0.4, 0.5 is the column title. Each index of subset size is associated with the same index of the other list

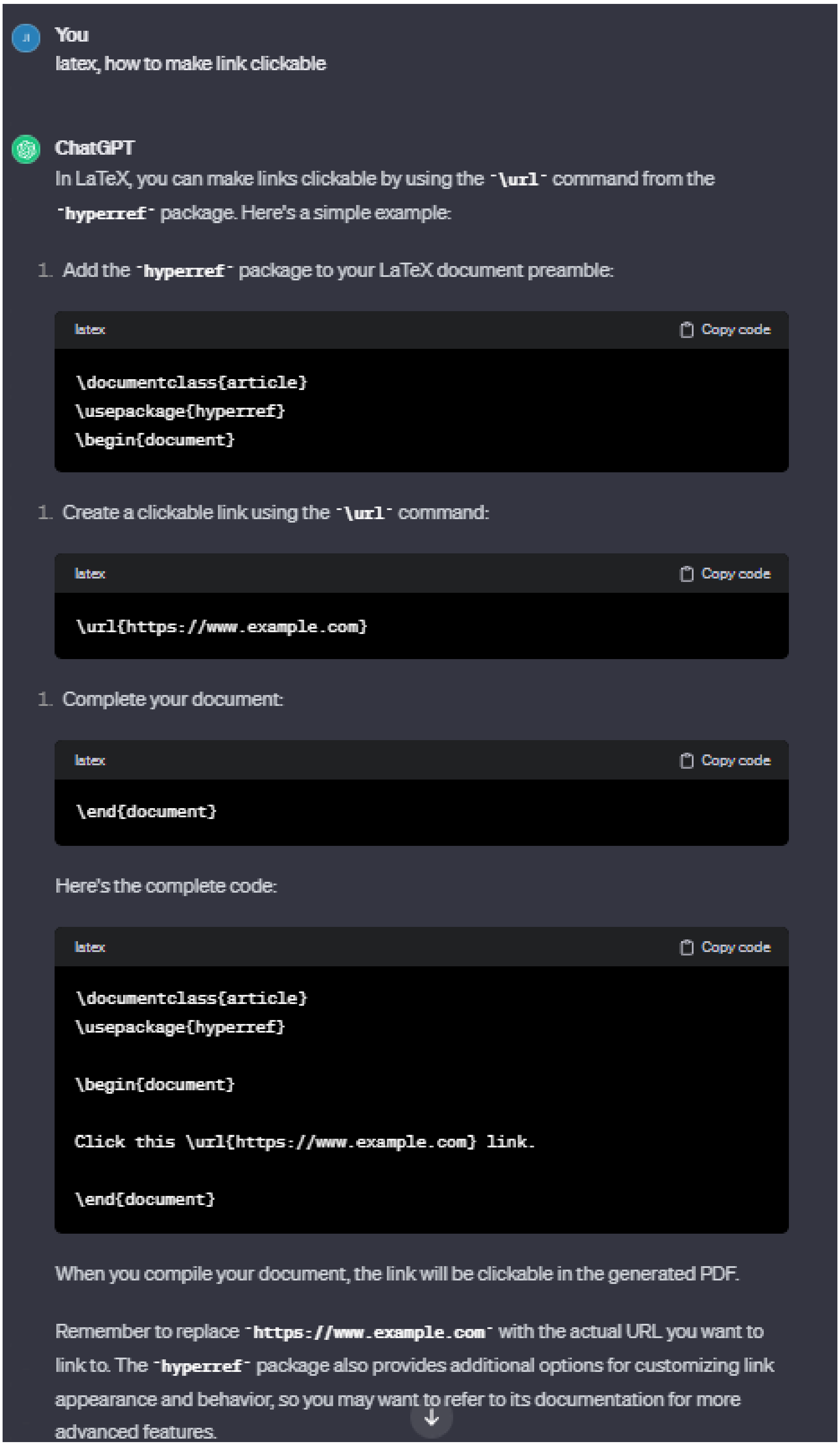
ChatGPT

Certainly! Here is a LaTeX table for the provided data:

latex Copy code

```latex
\documentclass{article}
\usepackage{multirow}
\begin{document}

\begin{table}[h]
\centering
\caption{Data Table}
\begin{tabular}{|c|ccccccccc|}
\hline
\multirow{2}{*}{Subset Size} & \multicolumn{5}{c|}{\textbf{Column Title
\cline{2-6}
& \textbf{0.1} & \textbf{0.2} & \textbf{0.3} & \textbf{0.4} & \textbf{0
\hline
10-20 & 0.00249 & 0.00135 & 0.00106 & 0.000968 & 0.000896 \\
20-30 & 0.000939 & 0.000495 & 0.000353 & 0.000334 & 0.000312 \\
30-40 & 0.000537 & 0.000271 & 0.000197 & 0.000185 & 0.000175 \\
40-50 & 0.000307 & 0.000193 & 0.000151 & 0.000122 & 0.000104 \\
50-60 & 0.000206 & 0.000122 & 9.71e-05 & 8.71e-05 & 8.07e-05 \\
60-70 & 0.00014 & 7.1e-05 & 7.31e-05 & 6.27e-05 & 5.4e-05 \\
70-80 & 0.000105 & 6.45e-05 & 5.36e-05 & 5.17e-05 & 4.11e-05 \\
80-100 & 8.94e-05 & 5.8e-05 & 3.7e-05 & 3.9e-05 & 3.64e-05 \\
\hline
\end{tabular}
\end{table}

\end{document}
```

This LaTeX code uses the `multirow` package to create multirow cells for the "Subset Size" column and the `\multicolumn` command to create the column titles. You can copy and paste this code into a LaTeX document to generate the table. Adjustments can be made to the formatting based on your preferences.



You

latex, how to make link clickable

ChatGPT

In LaTeX, you can make links clickable by using the `\url` command from the `hyperref` package. Here's a simple example:

1. Add the `hyperref` package to your LaTeX document preamble:

```latex
\documentclass{article}
\usepackage{hyperref}
\begin{document}
```

1. Create a clickable link using the `\url` command:

```latex
\url{https://www.example.com}
```

1. Complete your document:

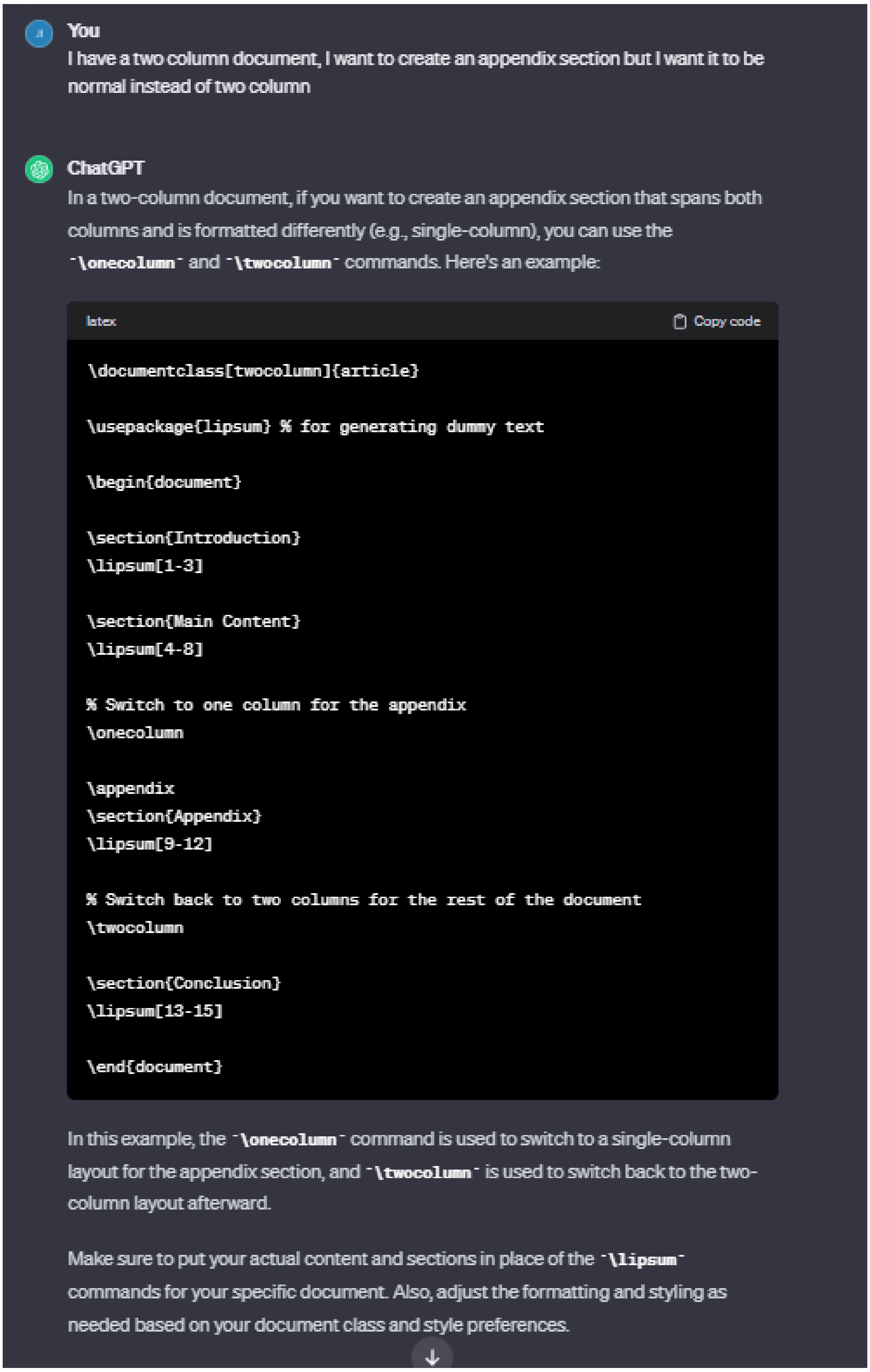
```latex
\end{document}
```

Here's the complete code:

```latex
\documentclass{article}
\usepackage{hyperref}

\begin{document}

Click this \url{https://www.example.com} link.

\end{document}
```

When you compile your document, the link will be clickable in the generated PDF.

Remember to replace `https://www.example.com` with the actual URL you want to link to. The `hyperref` package also provides additional options for customizing link appearance and behavior, so you may want to refer to its documentation for more advanced features.



You

I have a two column document, I want to create an appendix section but I want it to be normal instead of two column

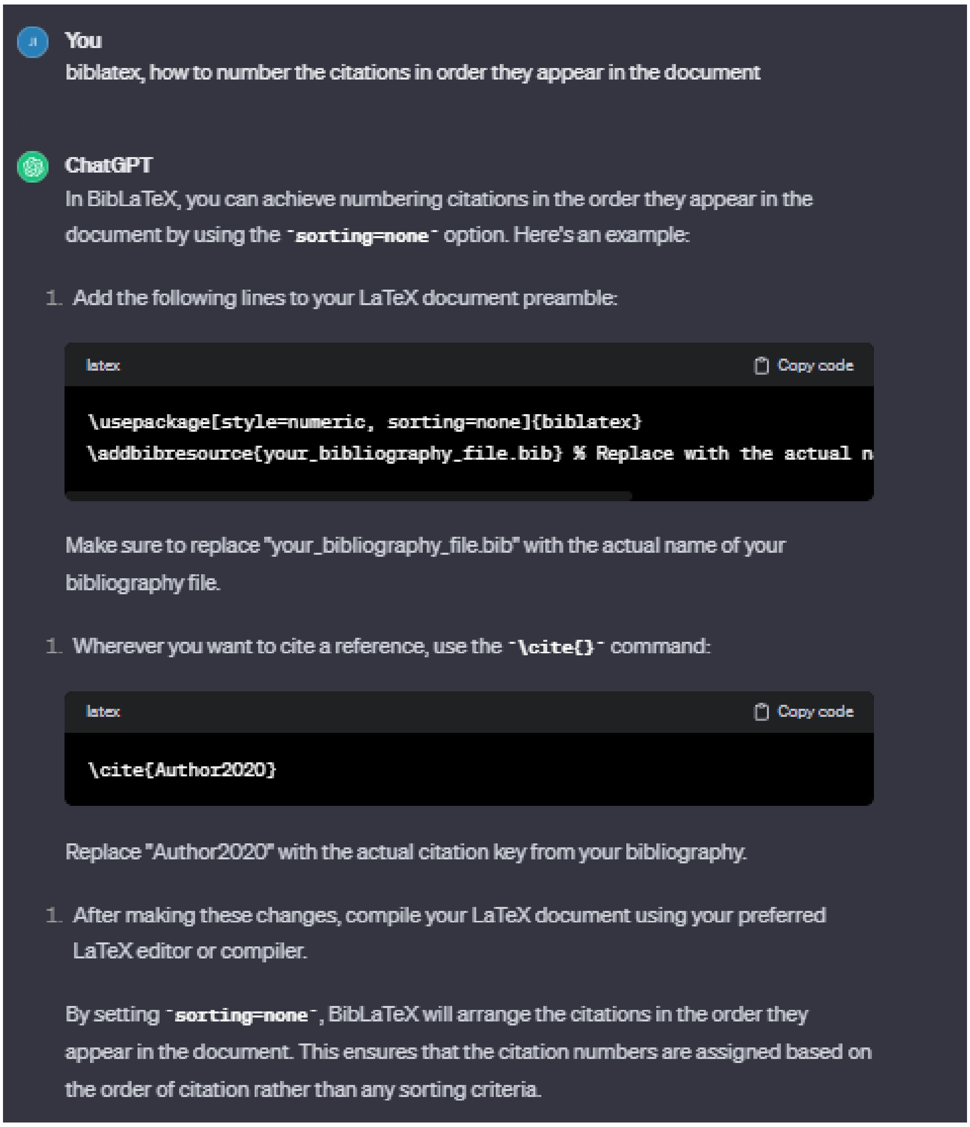
ChatGPT

In a two-column document, if you want to create an appendix section that spans both columns and is formatted differently (e.g., single-column), you can use the `\onecolumn` and `\twocolumn` commands. Here's an example:

```latex
\documentclass[twocolumn]{article}

\usepackage{lipsum} % for generating dummy text

\begin{document}

\section{Introduction}
\lipsum[1-3]

\section{Main Content}
\lipsum[4-8]

% Switch to one column for the appendix
\onecolumn

\appendix
\section{Appendix}
\lipsum[9-12]

% Switch back to two columns for the rest of the document
\twocolumn

\section{Conclusion}
\lipsum[13-15]

\end{document}
```

In this example, the `\onecolumn` command is used to switch to a single-column layout for the appendix section, and `\twocolumn` is used to switch back to the two-column layout afterward.

Make sure to put your actual content and sections in place of the `\lipsum` commands for your specific document. Also, adjust the formatting and styling as needed based on your document class and style preferences.



You

biblatex, how to number the citations in order they appear in the document

ChatGPT

In BibLaTeX, you can achieve numbering citations in the order they appear in the document by using the `sorting=none` option. Here's an example:

1. Add the following lines to your LaTeX document preamble:

```latex
\usepackage[style=numeric, sorting=none]{biblatex}
\addbibresource{your_bibliography_file.bib} % Replace with the actual n
```

Make sure to replace "your_bibliography_file.bib" with the actual name of your bibliography file.

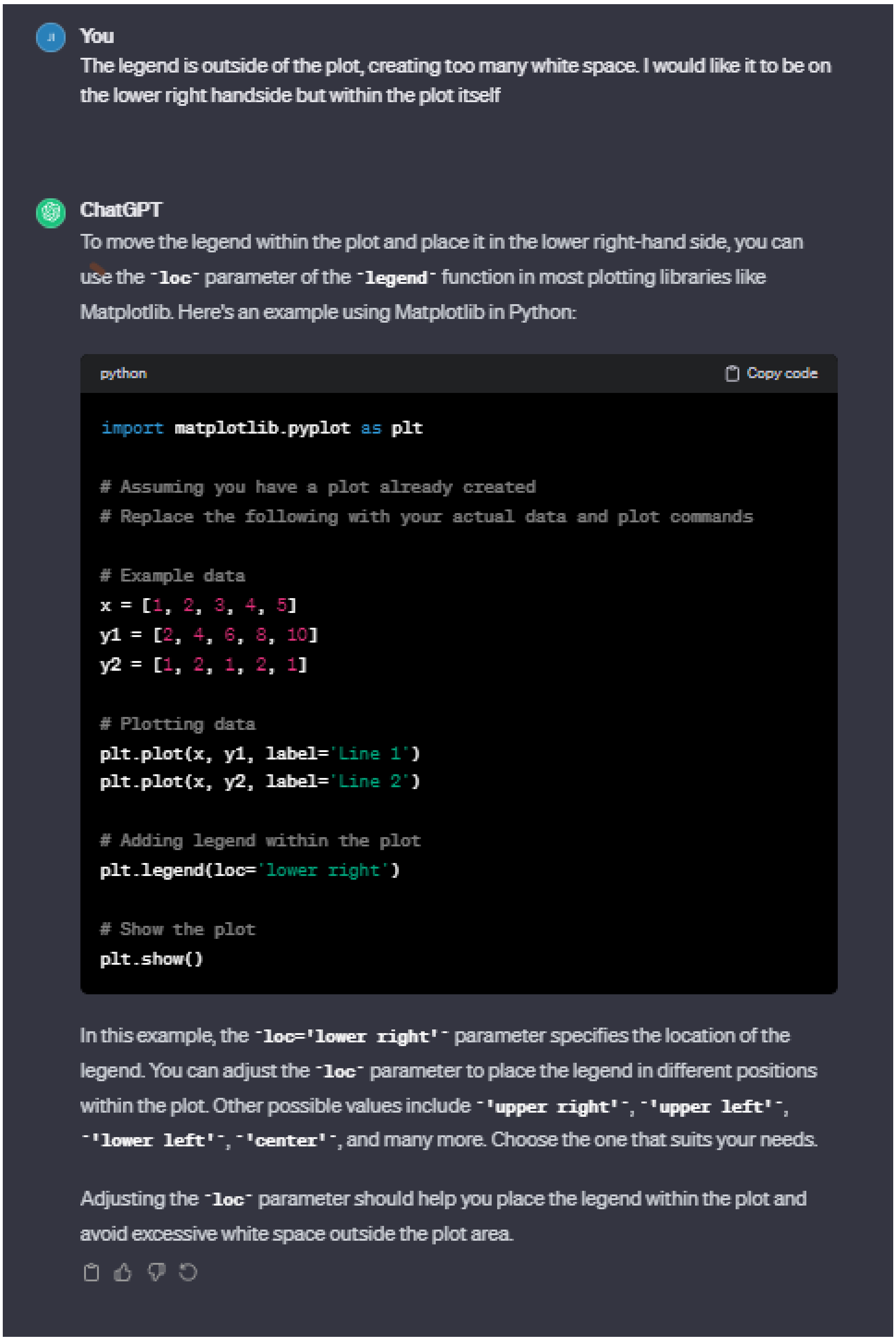
1. Wherever you want to cite a reference, use the `\cite{}` command:

```latex
\cite{Author2020}
```

Replace "Author2020" with the actual citation key from your bibliography.

1. After making these changes, compile your LaTeX document using your preferred LaTeX editor or compiler.

By setting `sorting=none`, BibLaTeX will arrange the citations in the order they appear in the document. This ensures that the citation numbers are assigned based on the order of citation rather than any sorting criteria.



You

The legend is outside of the plot, creating too many white space. I would like it to be on the lower right handside but within the plot itself

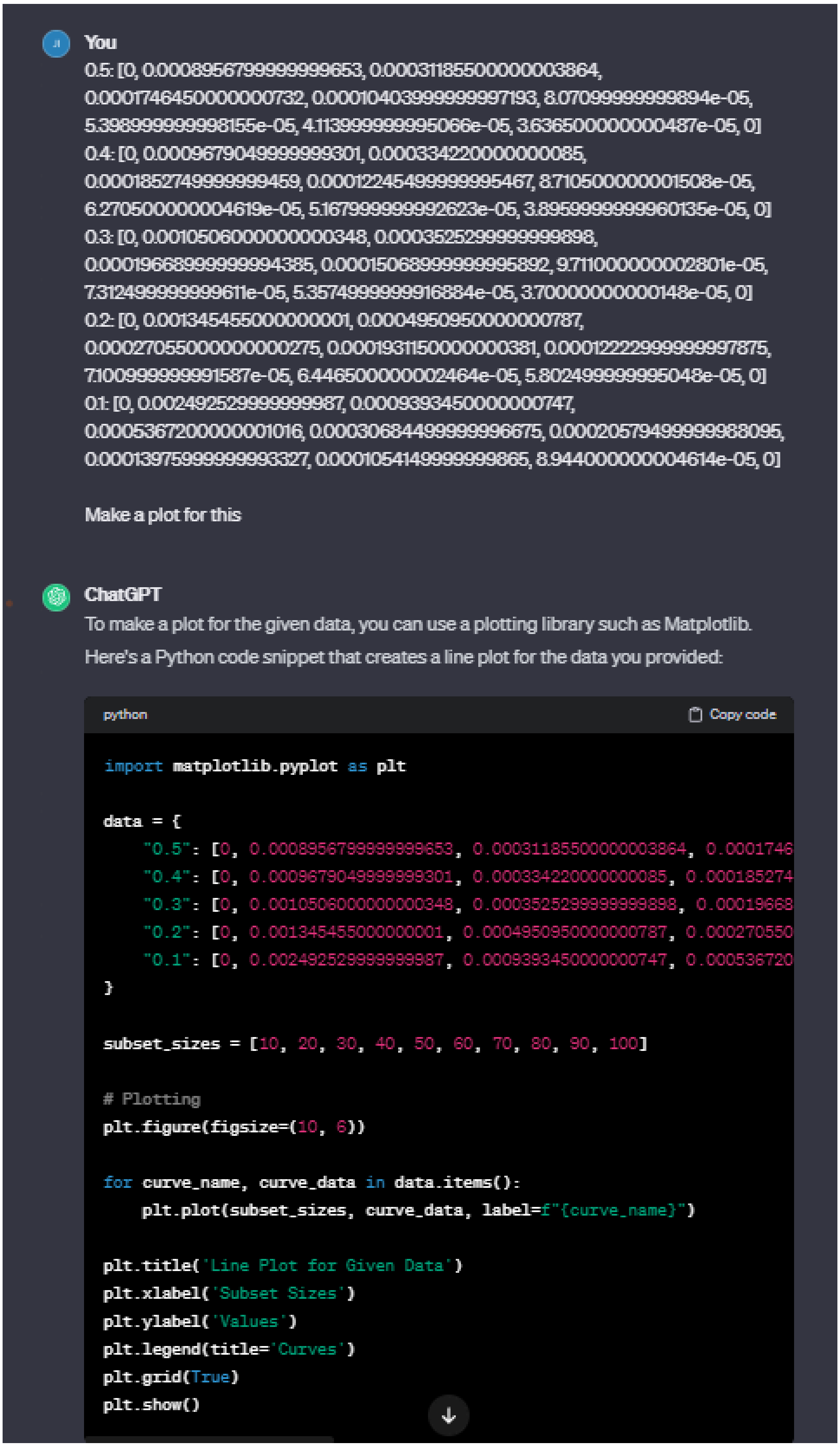
ChatGPT

To move the legend within the plot and place it in the lower right-hand side, you can use the `loc` parameter of the `legend` function in most plotting libraries like Matplotlib. Here's an example using Matplotlib in Python:

```python
import matplotlib.pyplot as plt

# Assuming you have a plot already created
# Replace the following with your actual data and plot commands

# Example data
x = [1, 2, 3, 4, 5]
y1 = [2, 4, 6, 8, 10]
y2 = [1, 2, 1, 2, 1]

# Plotting data
plt.plot(x, y1, label='Line 1')
plt.plot(x, y2, label='Line 2')

# Adding legend within the plot
plt.legend(loc='lower right')

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

In this example, the `loc='lower right'` parameter specifies the location of the legend. You can adjust the `loc` parameter to place the legend in different positions within the plot. Other possible values include `'upper right'`, `'upper left'`, `'lower left'`, `'center'`, and many more. Choose the one that suits your needs.

Adjusting the `loc` parameter should help you place the legend within the plot and avoid excessive white space outside the plot area.



You

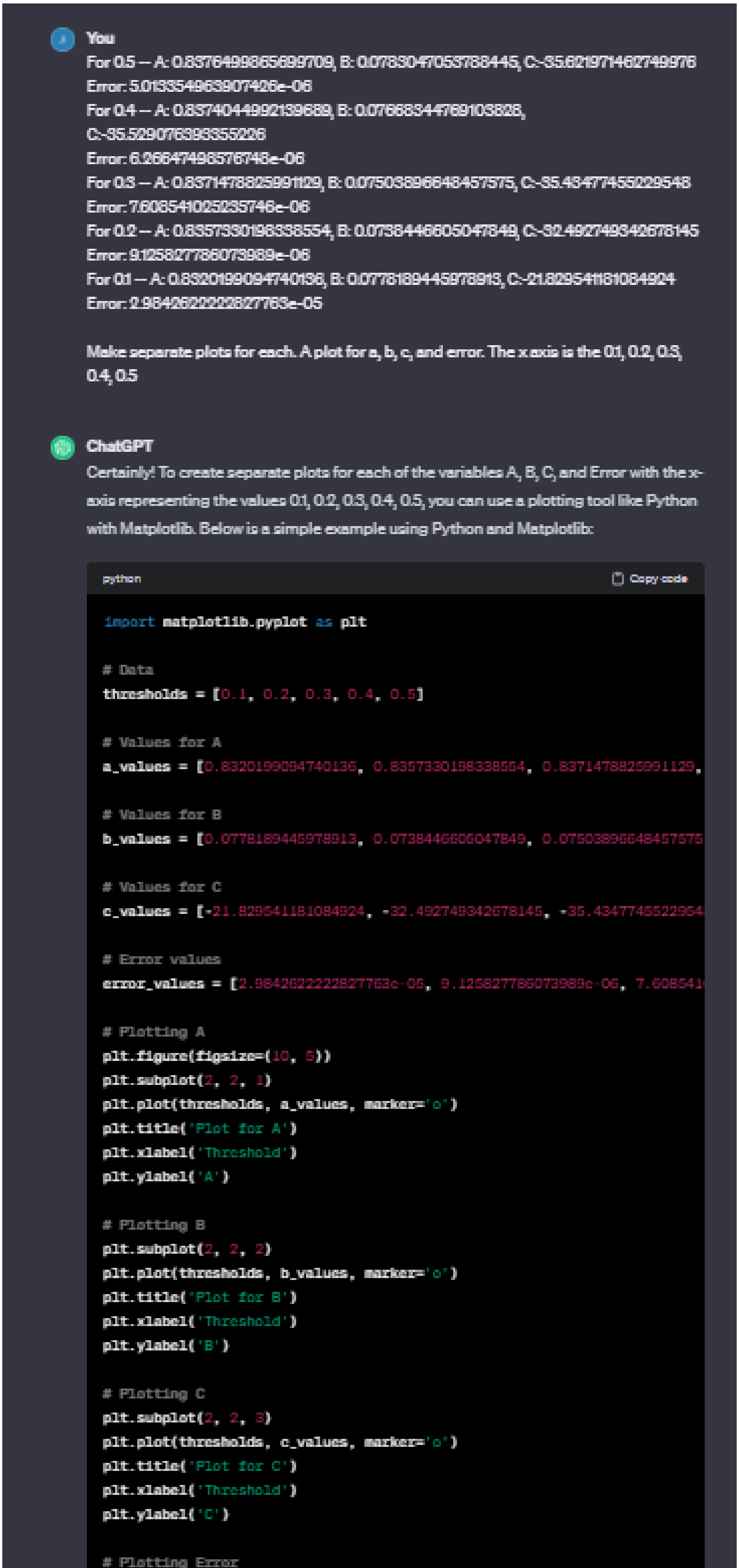
0.5: [0, 0.0008956799999999653, 0.0003118550000000003864,
0.0001746450000000000732, 0.00010403999999997193, 8.07099999999894e-05,
5.398999999998155e-05, 4.113999999995066e-05, 3.636500000000487e-05, 0]
0.4: [0, 0.0009679049999999301, 0.000334220000000000085,
0.0001852749999999459, 0.00012245499999995467, 8.710500000001508e-05,
6.270500000004619e-05, 5.167999999992623e-05, 3.895999999960135e-05, 0]
0.3: [0, 0.0010506000000000000348, 0.00035252999999999898,
0.00019668999999994385, 0.00015068999999995892, 9.711000000002801e-05,
7.312499999999611e-05, 5.357499999916884e-05, 3.700000000000148e-05, 0]
0.2: [0, 0.001345455000000001, 0.000495095000000000787,
0.000270550000000000275, 0.00019311500000000381, 0.00012222999999997875,
7.100999999991587e-05, 6.446500000002464e-05, 5.802499999995048e-05, 0]
0.1: [0, 0.0024925299999999987, 0.000939345000000000747,
0.000536720000000001016, 0.00030684499999996675, 0.00020579499999988095,
0.00013975999999993327, 0.00010541499999999865, 8.944000000004614e-05, 0]

Make a plot for this

ChatGPT

To make a plot for the given data, you can use a plotting library such as Matplotlib. Here's a Python code snippet that creates a line plot for the data you provided:

```python
import matplotlib.pyplot as plt

data = {
    "0.5": [0, 0.0008956799999999653, 0.000311855000000003864, 0.0001746
    "0.4": [0, 0.0009679049999999301, 0.00033422000000000085, 0.000185274
    "0.3": [0, 0.00105060000000000348, 0.00035252999999999898, 0.00019668
    "0.2": [0, 0.00134545500000000001, 0.000495095000000000787, 0.000270550
    "0.1": [0, 0.0024925299999999987, 0.000939345000000000747, 0.000536720
}

subset_sizes = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

# Plotting
plt.figure(figsize=(10, 6))

for curve_name, curve_data in data.items():
    plt.plot(subset_sizes, curve_data, label=f"{curve_name}")

plt.title('Line Plot for Given Data')
plt.xlabel('Subset Sizes')
plt.ylabel('Values')
plt.legend(title='Curves')
plt.grid(True)
plt.show()
```



You

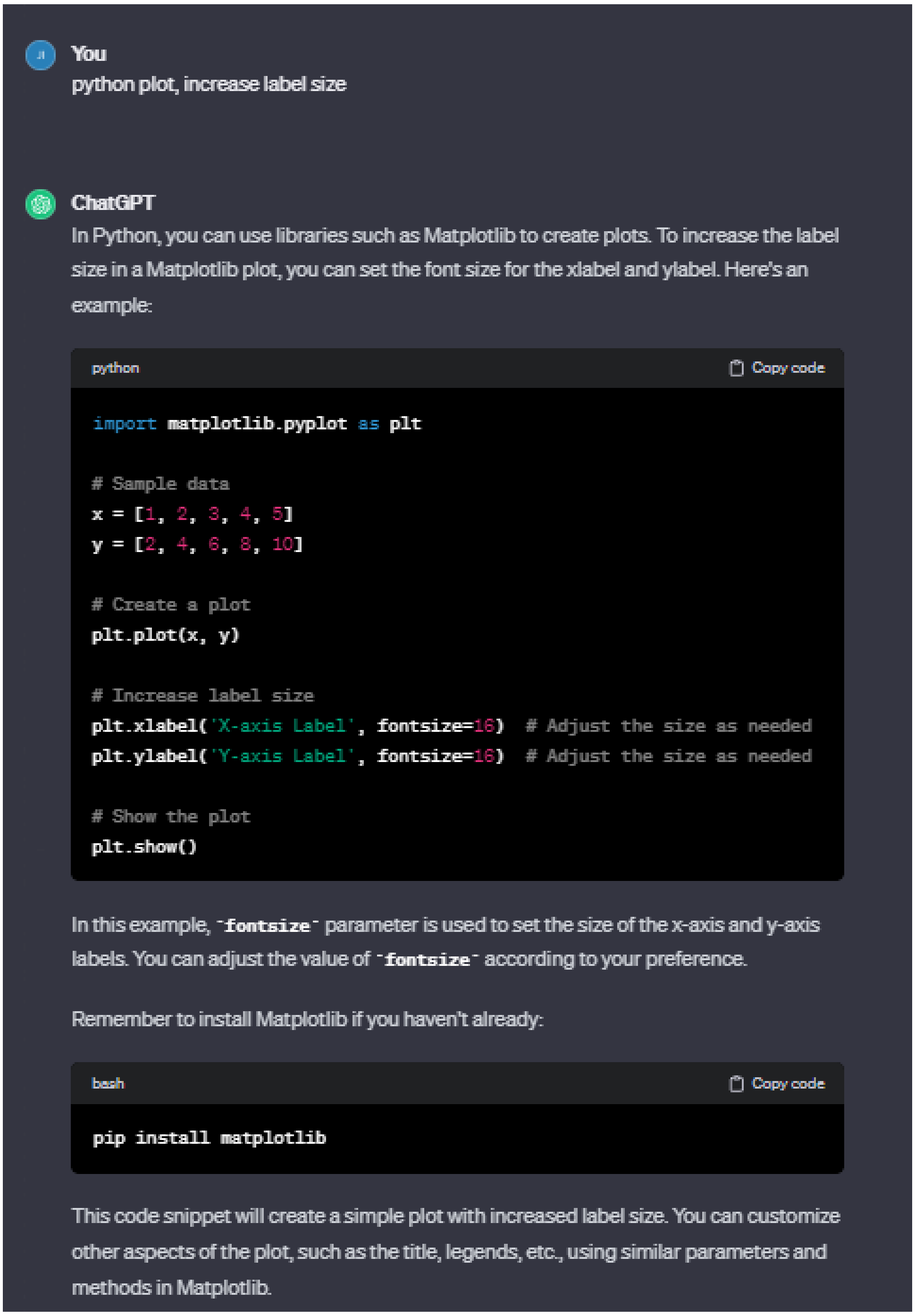
For 0.5 -- A: 0.8376499865699709, B: 0.07830470537788445, C:-35.621971462749976
Error: 5.013354963907426e-06
For 0.4 -- A: 0.8374044992139689, B: 0.0766834476910382 8,
C:-35.52907639335 5226
Error: 6.266474985767 48e-06
For 0.3 -- A: 0.8371478825991129, B: 0.07503896648457575, C:-35.43477455229548
Error: 7.608541025235746e-06
For 0.2 -- A: 0.8357330198338554, B: 0.07384466050478 49, C:-32.492749342678145
Error: 9.125827786073989e-06
For 0.1 -- A: 0.8320199094740136, B: 0.07781894459 78913, C:-21.829541181084924
Error: 2.9842622228277 63e-05

Make separate plots for each. A plot for a, b, c, and error. The x axis is the 0.1, 0.2, 0.3, 0.4, 0.5

ChatGPT

Certainly! To create separate plots for each of the variables A, B, C, and Error with the x-axis representing the values 0.1, 0.2, 0.3, 0.4, 0.5, you can use a plotting tool like Python with Matplotlib. Below is a simple example using Python and Matplotlib:

```python
import matplotlib.pyplot as plt

# Data
thresholds = [0.1, 0.2, 0.3, 0.4, 0.5]

# Values for A
a_values = [0.8320199094740136, 0.8357330198338554, 0.8371478825991129,

# Values for B
b_values = [0.0778189445978913, 0.0738446605047849, 0.07503896648457575

# Values for C
c_values = [-21.829541181084924, -32.492749342678145, -35.434774552295

# Error values
error_values = [2.984262222828 7763e-05, 9.125827786073989e-06, 7.60854

# Plotting A
plt.figure(figsize=(10, 5))
plt.subplot(2, 2, 1)
plt.plot(thresholds, a_values, marker='o')
plt.title('Plot for A')
plt.xlabel('Threshold')
plt.ylabel('A')

# Plotting B
plt.subplot(2, 2, 2)
plt.plot(thresholds, b_values, marker='o')
plt.title('Plot for B')
plt.xlabel('Threshold')
plt.ylabel('B')

# Plotting C
plt.subplot(2, 2, 3)
plt.plot(thresholds, c_values, marker='o')
plt.title('Plot for C')
plt.xlabel('Threshold')
plt.ylabel('C')

# Plotting Error
```



You

python plot, increase label size

ChatGPT

In Python, you can use libraries such as Matplotlib to create plots. To increase the label size in a Matplotlib plot, you can set the font size for the xlabel and ylabel. Here's an example:

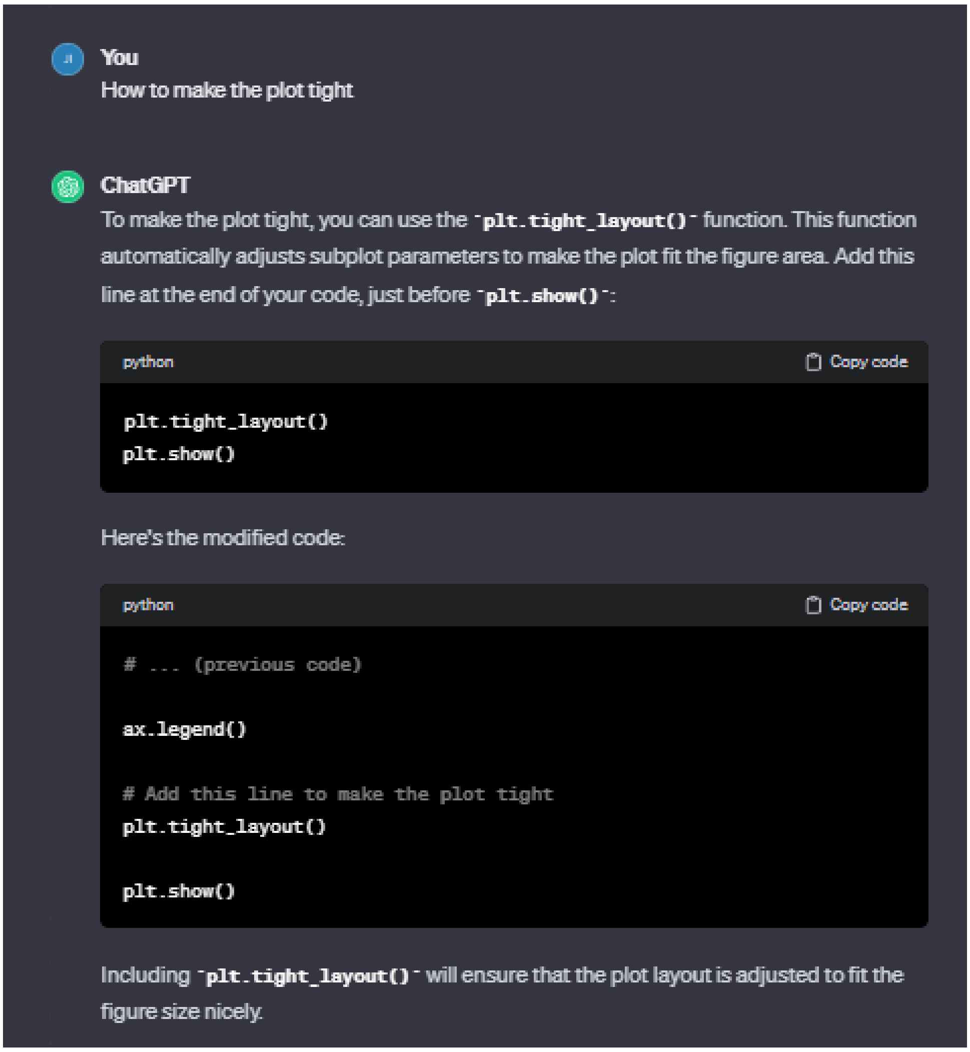
```python
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Create a plot
plt.plot(x, y)

# Increase label size
plt.xlabel('X-axis Label', fontsize=16)  # Adjust the size as needed
plt.ylabel('Y-axis Label', fontsize=16)  # Adjust the size as needed

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

In this example, `fontsize` parameter is used to set the size of the x-axis and y-axis labels. You can adjust the value of `fontsize` according to your preference.

Remember to install Matplotlib if you haven't already:

```bash
pip install matplotlib
```

This code snippet will create a simple plot with increased label size. You can customize other aspects of the plot, such as the title, legends, etc., using similar parameters and methods in Matplotlib.


You

How to make the plot tight

ChatGPT

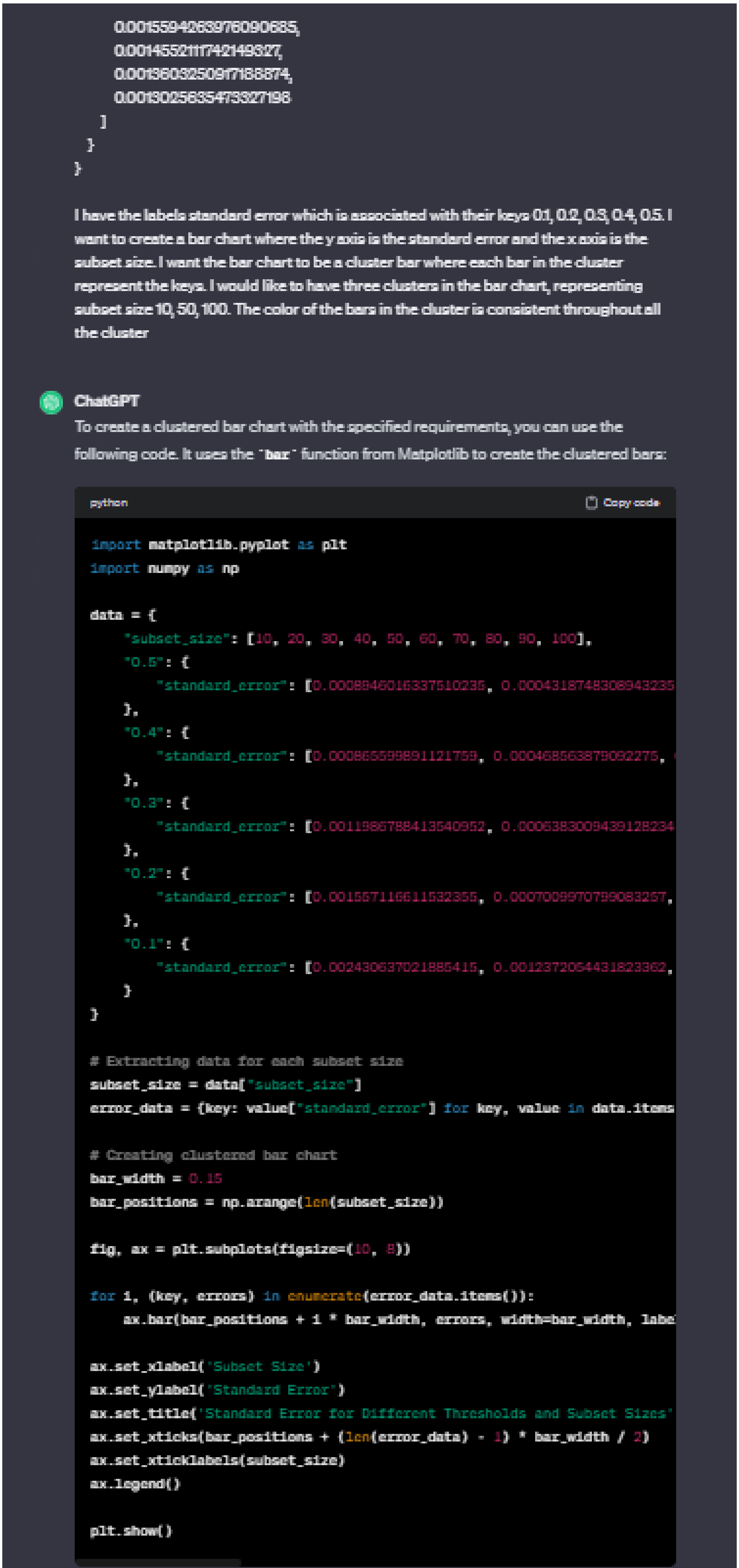
To make the plot tight, you can use the `plt.tight_layout()` function. This function automatically adjusts subplot parameters to make the plot fit the figure area. Add this line at the end of your code, just before `plt.show()`:

```python
plt.tight_layout()
plt.show()
```

Here's the modified code:

```python
# ... (previous code)

ax.legend()

# Add this line to make the plot tight
plt.tight_layout()

plt.show()
```

Including `plt.tight_layout()` will ensure that the plot layout is adjusted to fit the figure size nicely.



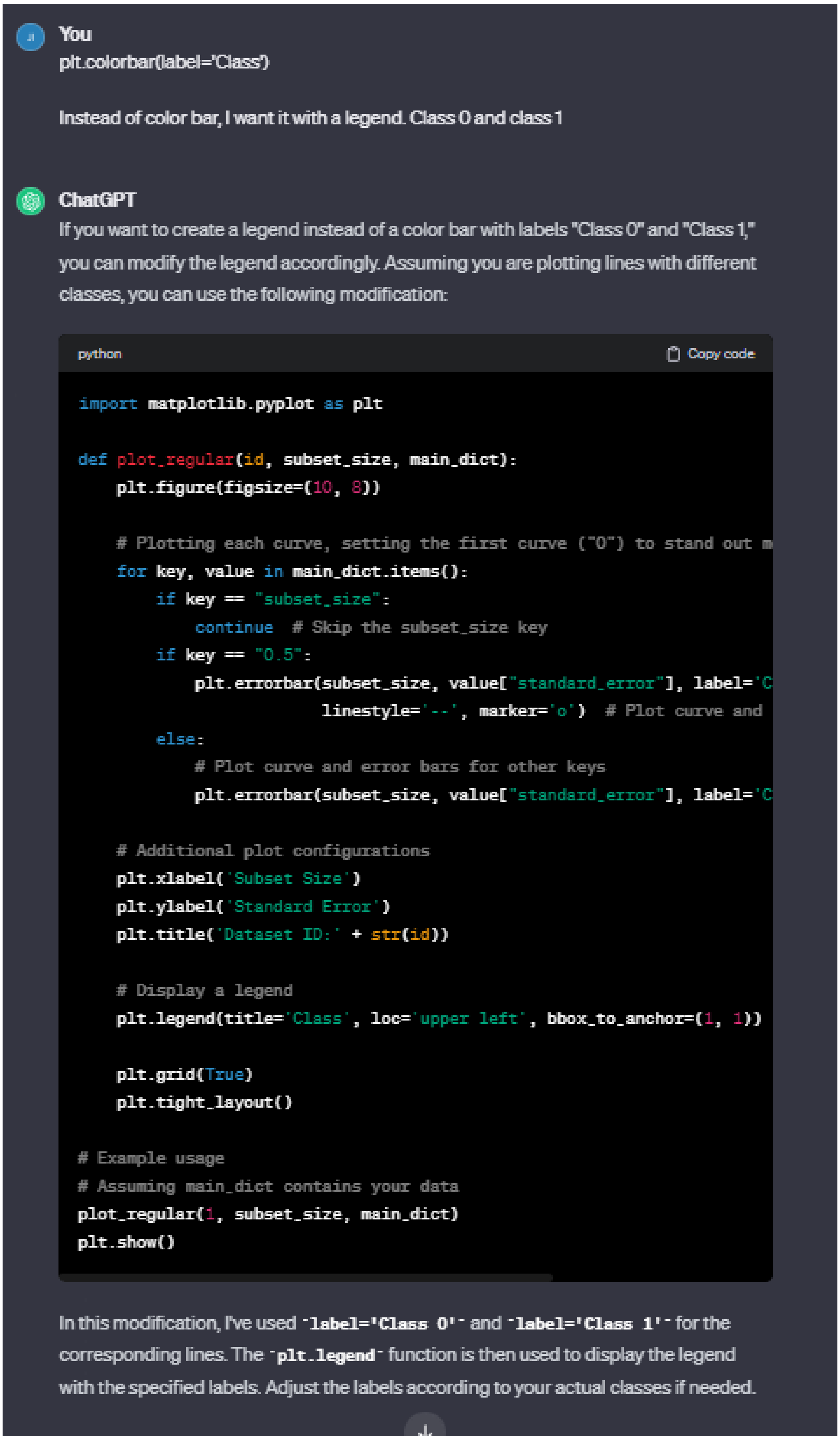
```
        0.0015594263976090685,
        0.0014552111742149327,
        0.0013603250917188874,
        0.0013025635473327198
      ]
    }
}
```

I have the labels standard error which is associated with their keys 0.1, 0.2, 0.3, 0.4, 0.5. I want to create a bar chart where the y axis is the standard error and the x axis is the subset size. I want the bar chart to be a cluster bar where each bar in the cluster represent the keys. I would like to have three clusters in the bar chart, representing subset size 10, 50, 100. The color of the bars in the cluster is consistent throughout all the cluster

ChatGPT

To create a clustered bar chart with the specified requirements, you can use the following code. It uses the `bar` function from Matplotlib to create the clustered bars:

```python
import matplotlib.pyplot as plt
import numpy as np

data = {
    "subset_size": [10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
    "0.5": {
        "standard_error": [0.0008946016337510235, 0.0004318748308943235
    },
    "0.4": {
        "standard_error": [0.0008655998891121759, 0.000468563879092275,
    },
    "0.3": {
        "standard_error": [0.0011986788413540952, 0.0006383009439128234
    },
    "0.2": {
        "standard_error": [0.001557116611532355, 0.0007009970799083257,
    },
    "0.1": {
        "standard_error": [0.002430637021885415, 0.0012372054431823362,
    }
}

# Extracting data for each subset size
subset_size = data["subset_size"]
error_data = {key: value["standard_error"] for key, value in data.items

# Creating clustered bar chart
bar_width = 0.15
bar_positions = np.arange(len(subset_size))

fig, ax = plt.subplots(figsize=(10, 8))

for i, (key, errors) in enumerate(error_data.items()):
    ax.bar(bar_positions + i * bar_width, errors, width=bar_width, labe

ax.set_xlabel('Subset Size')
ax.set_ylabel('Standard Error')
ax.set_title('Standard Error for Different Thresholds and Subset Sizes'
ax.set_xticks(bar_positions + (len(error_data) - 1) * bar_width / 2)
ax.set_xticklabels(subset_size)
ax.legend()

plt.show()
```



You

plt.colorbar(label='Class')

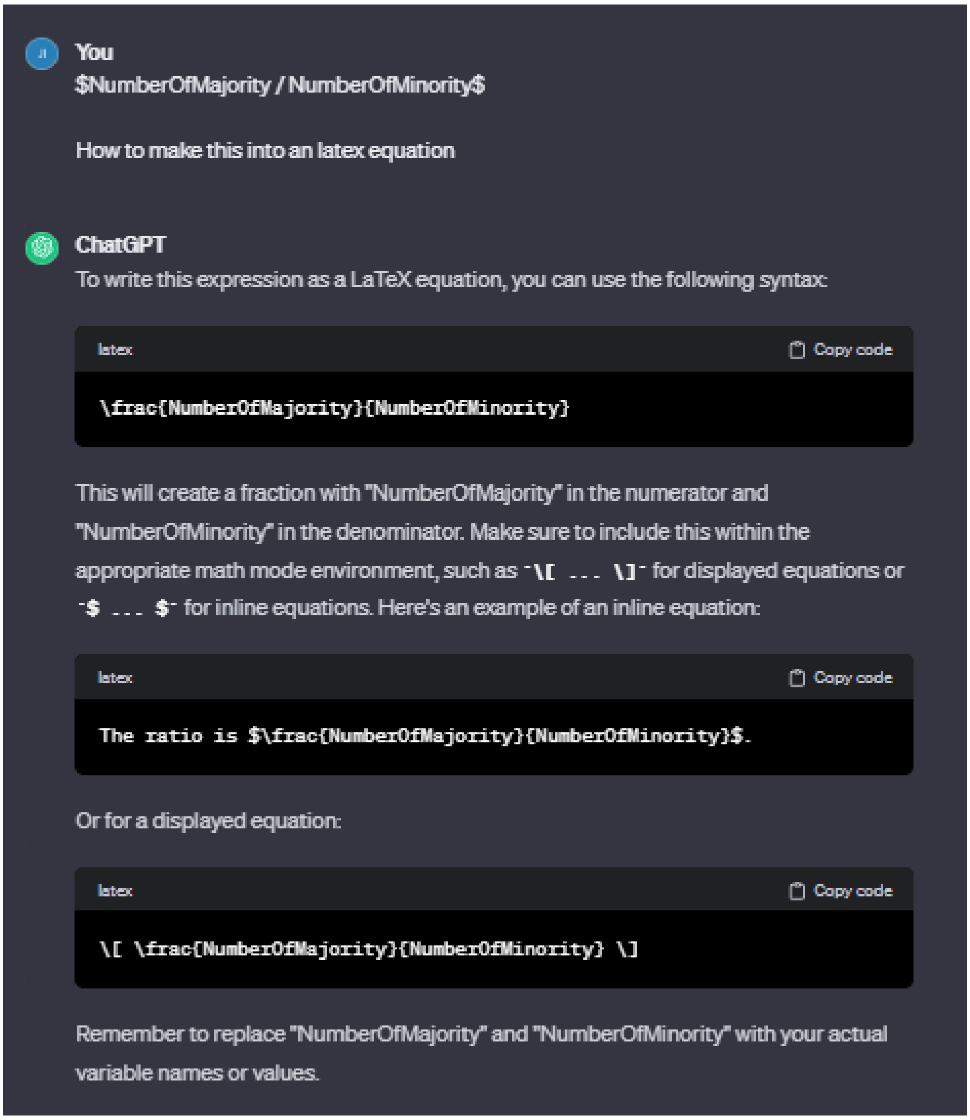
Instead of color bar, I want it with a legend. Class 0 and class 1

ChatGPT

If you want to create a legend instead of a color bar with labels "Class 0" and "Class 1," you can modify the legend accordingly. Assuming you are plotting lines with different classes, you can use the following modification:

```python
import matplotlib.pyplot as plt

def plot_regular(id, subset_size, main_dict):
    plt.figure(figsize=(10, 8))

    # Plotting each curve, setting the first curve ("0") to stand out m
    for key, value in main_dict.items():
        if key == "subset_size":
            continue  # Skip the subset_size key
        if key == "0.5":
            plt.errorbar(subset_size, value["standard_error"], label='C
                         linestyle='--', marker='o')  # Plot curve and
        else:
            # Plot curve and error bars for other keys
            plt.errorbar(subset_size, value["standard_error"], label='C

    # Additional plot configurations
    plt.xlabel('Subset Size')
    plt.ylabel('Standard Error')
    plt.title('Dataset ID:' + str(id))

    # Display a legend
    plt.legend(title='Class', loc='upper left', bbox_to_anchor=(1, 1))

    plt.grid(True)
    plt.tight_layout()

# Example usage
# Assuming main_dict contains your data
plot_regular(1, subset_size, main_dict)
plt.show()
```

In this modification, I've used `label='Class 0'` and `label='Class 1'` for the corresponding lines. The `plt.legend` function is then used to display the legend with the specified labels. Adjust the labels according to your actual classes if needed.



You

$NumberOfMajority / NumberOfMinority$

How to make this into an latex equation

ChatGPT

To write this expression as a LaTeX equation, you can use the following syntax:

```latex
\frac{NumberOfMajority}{NumberOfMinority}
```

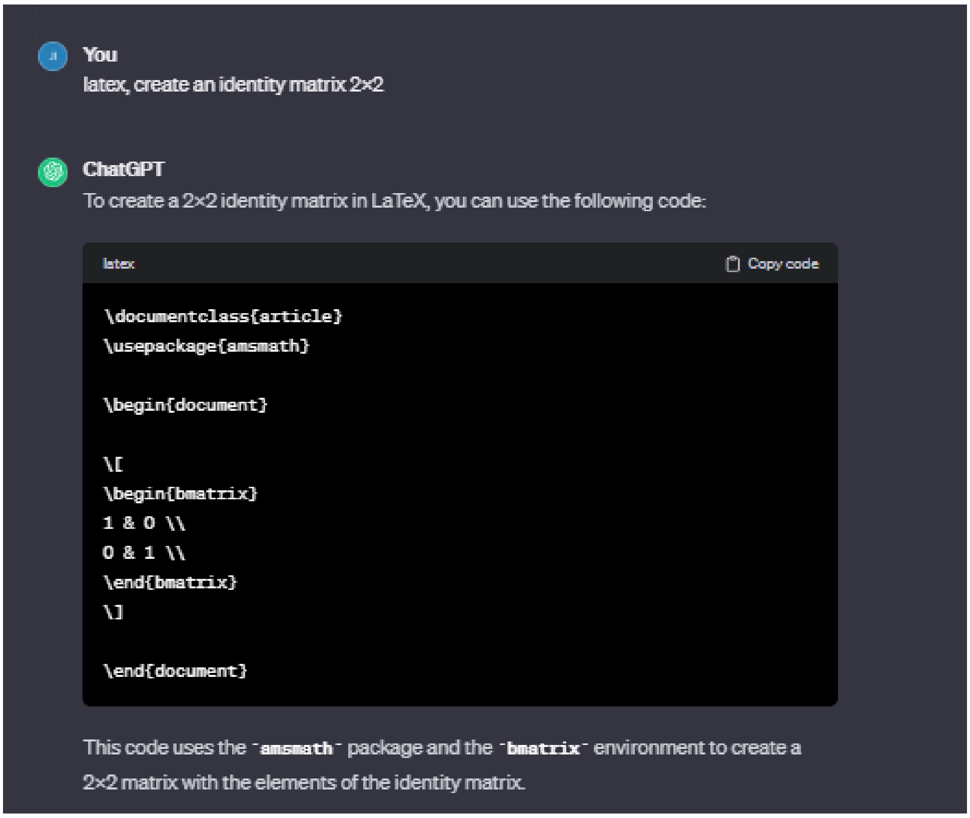
This will create a fraction with "NumberOfMajority" in the numerator and "NumberOfMinority" in the denominator. Make sure to include this within the appropriate math mode environment, such as `\[ ... \]` for displayed equations or `$ ... $` for inline equations. Here's an example of an inline equation:

```latex
The ratio is $\frac{NumberOfMajority}{NumberOfMinority}$.
```

Or for a displayed equation:

```latex
\[ \frac{NumberOfMajority}{NumberOfMinority} \]
```

Remember to replace "NumberOfMajority" and "NumberOfMinority" with your actual variable names or values.



You

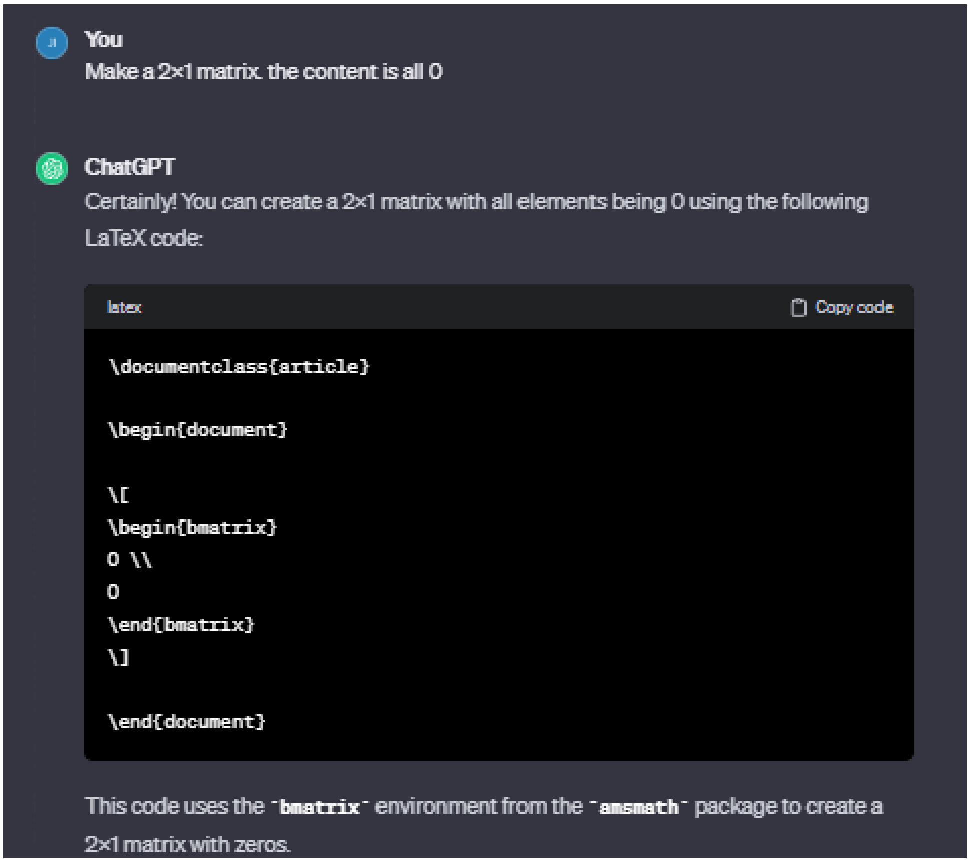
latex, create an identity matrix 2×2

ChatGPT

To create a 2×2 identity matrix in LaTeX, you can use the following code:

```latex
\documentclass{article}
\usepackage{amsmath}

\begin{document}

\[
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
\end{bmatrix}
\]

\end{document}
```

This code uses the `amsmath` package and the `bmatrix` environment to create a 2×2 matrix with the elements of the identity matrix.



You

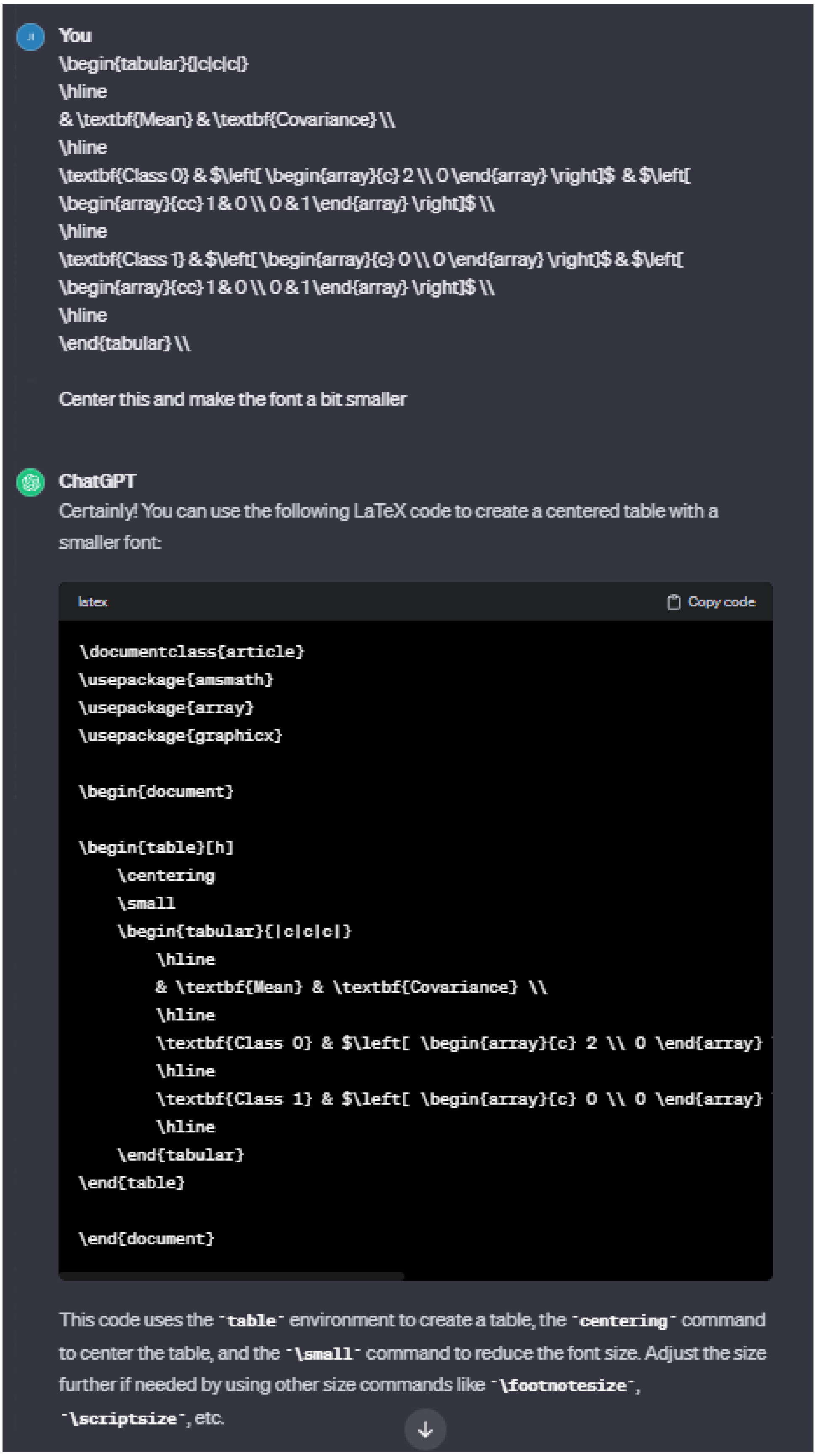
Make a 2×1 matrix. the content is all 0

ChatGPT

Certainly! You can create a 2×1 matrix with all elements being 0 using the following LaTeX code:

```latex
\documentclass{article}

\begin{document}

\[
\begin{bmatrix}
0 \\
0
\end{bmatrix}
\]

\end{document}
```

This code uses the `bmatrix` environment from the `amsmath` package to create a 2×1 matrix with zeros.



You

\begin{tabular}{|c|c|c|}
\hline
& \textbf{Mean} & \textbf{Covariance} \\
\hline
\textbf{Class 0} & $\left[\begin{array}{c} 2 \\ 0 \end{array} \right]$ & $\left[
\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right]$ \\
\hline
\textbf{Class 1} & $\left[\begin{array}{c} 0 \\ 0 \end{array} \right]$ & $\left[
\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right]$ \\
\hline
\end{tabular} \\

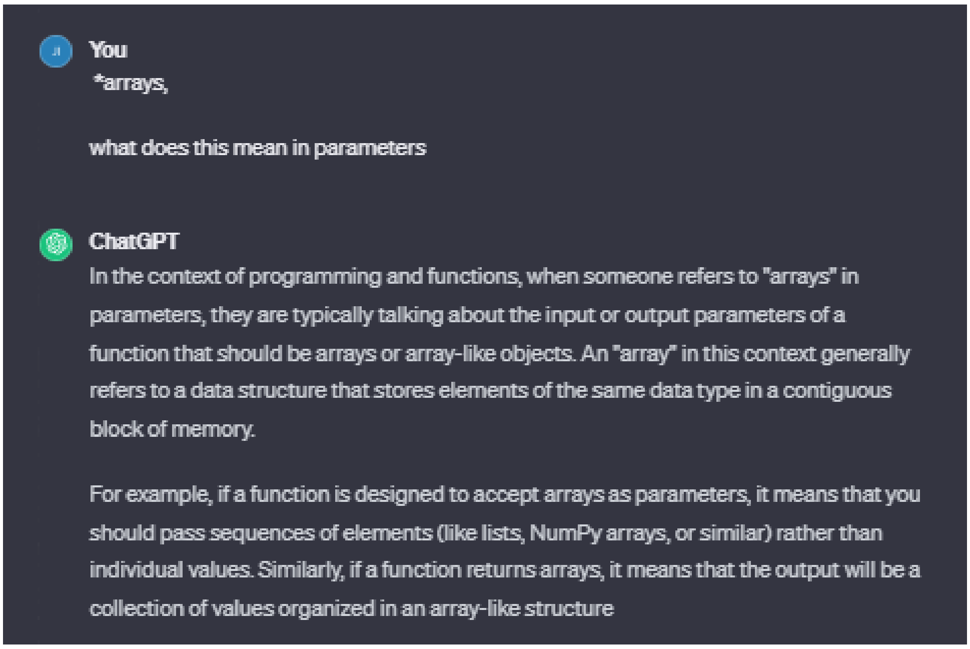
Center this and make the font a bit smaller

ChatGPT

Certainly! You can use the following LaTeX code to create a centered table with a smaller font:

```latex
\documentclass{article}
\usepackage{amsmath}
\usepackage{array}
\usepackage{graphicx}

\begin{document}

\begin{table}[h]
    \centering
    \small
    \begin{tabular}{|c|c|c|}
        \hline
        & \textbf{Mean} & \textbf{Covariance} \\
        \hline
        \textbf{Class 0} & $\left[ \begin{array}{c} 2 \\ 0 \end{array}
        \hline
        \textbf{Class 1} & $\left[ \begin{array}{c} 0 \\ 0 \end{array}
        \hline
    \end{tabular}
\end{table}

\end{document}
```

This code uses the `table` environment to create a table, the `centering` command to center the table, and the `\small` command to reduce the font size. Adjust the size further if needed by using other size commands like `\footnotesize`, `\scriptsize`, etc.



You

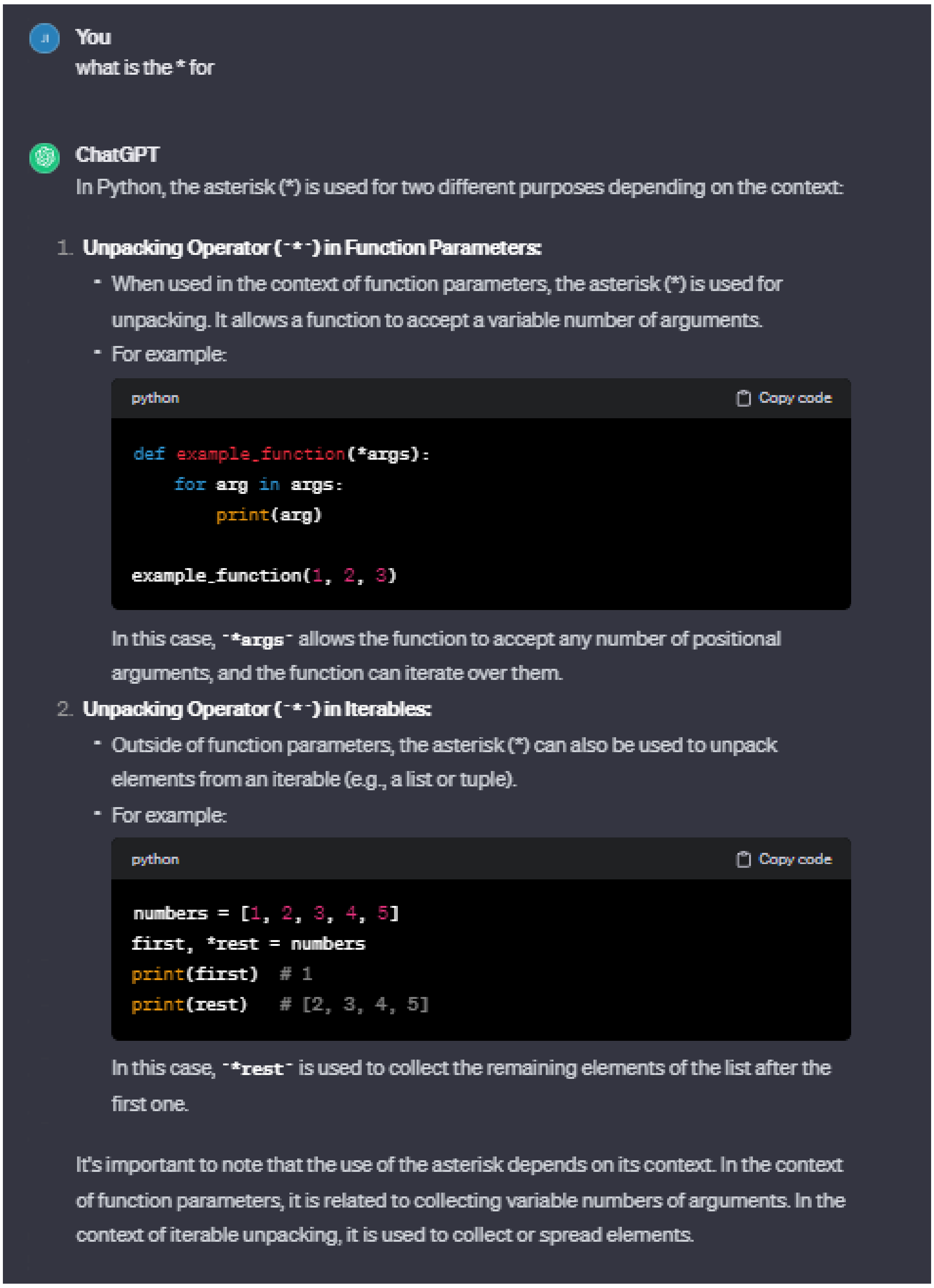
*arrays,

what does this mean in parameters

ChatGPT

In the context of programming and functions, when someone refers to "arrays" in parameters, they are typically talking about the input or output parameters of a function that should be arrays or array-like objects. An "array" in this context generally refers to a data structure that stores elements of the same data type in a contiguous block of memory.

For example, if a function is designed to accept arrays as parameters, it means that you should pass sequences of elements (like lists, NumPy arrays, or similar) rather than individual values. Similarly, if a function returns arrays, it means that the output will be a collection of values organized in an array-like structure



You

what is the * for

ChatGPT

In Python, the asterisk (*) is used for two different purposes depending on the context:

1. **Unpacking Operator (`*`) in Function Parameters:**
   - When used in the context of function parameters, the asterisk (*) is used for unpacking. It allows a function to accept a variable number of arguments.
   - For example:

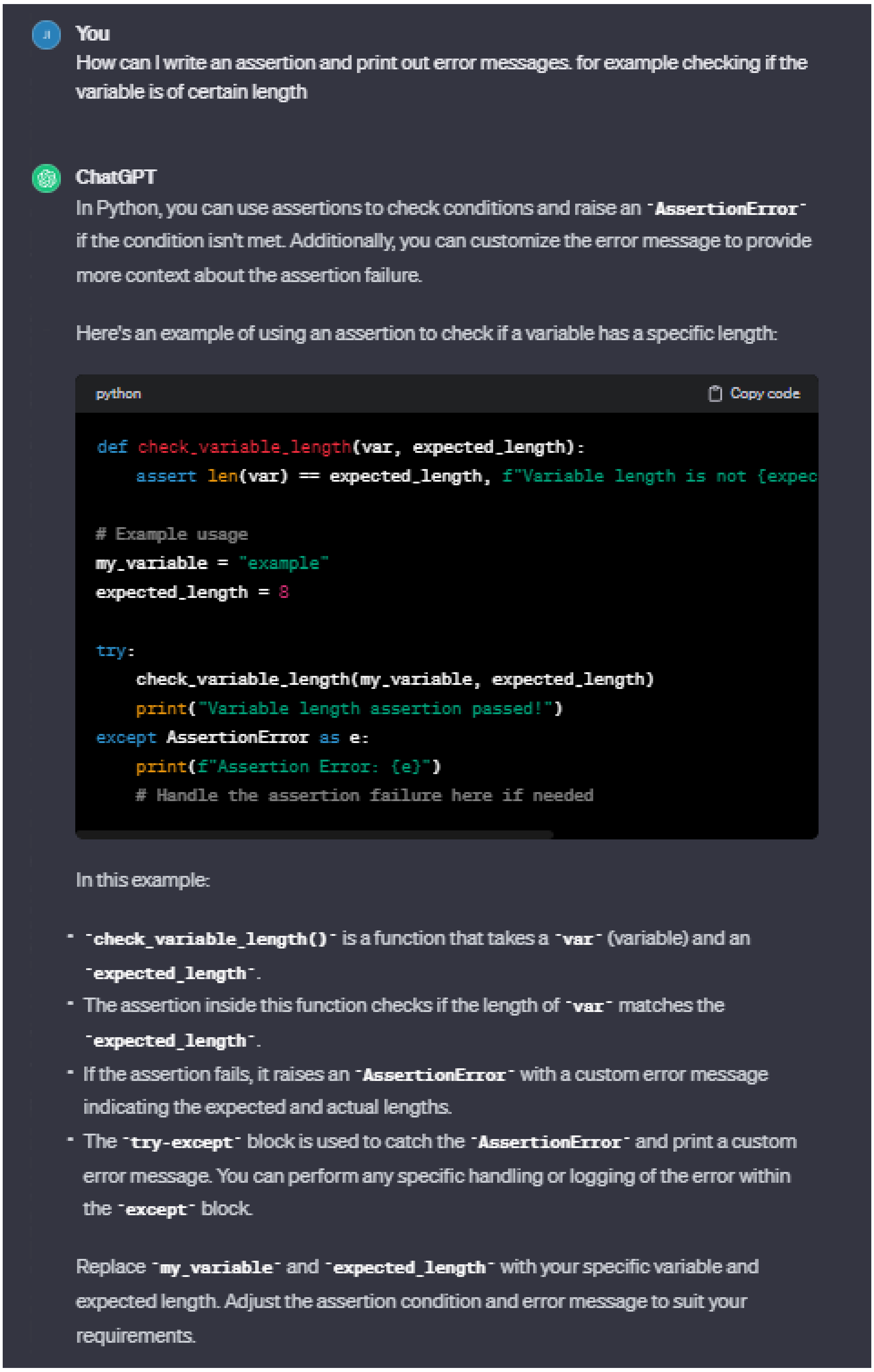
   ```python
   def example_function(*args):
       for arg in args:
           print(arg)

   example_function(1, 2, 3)
   ```

   In this case, `*args` allows the function to accept any number of positional arguments, and the function can iterate over them.
2. **Unpacking Operator (`*`) in Iterables:**
   - Outside of function parameters, the asterisk (*) can also be used to unpack elements from an iterable (e.g., a list or tuple).
   - For example:

   ```python
   numbers = [1, 2, 3, 4, 5]
   first, *rest = numbers
   print(first)  # 1
   print(rest)   # [2, 3, 4, 5]
   ```

   In this case, `*rest` is used to collect the remaining elements of the list after the first one.

It's important to note that the use of the asterisk depends on its context. In the context of function parameters, it is related to collecting variable numbers of arguments. In the context of iterable unpacking, it is used to collect or spread elements.



You

How can I write an assertion and print out error messages. for example checking if the variable is of certain length

ChatGPT

In Python, you can use assertions to check conditions and raise an `AssertionError` if the condition isn't met. Additionally, you can customize the error message to provide more context about the assertion failure.

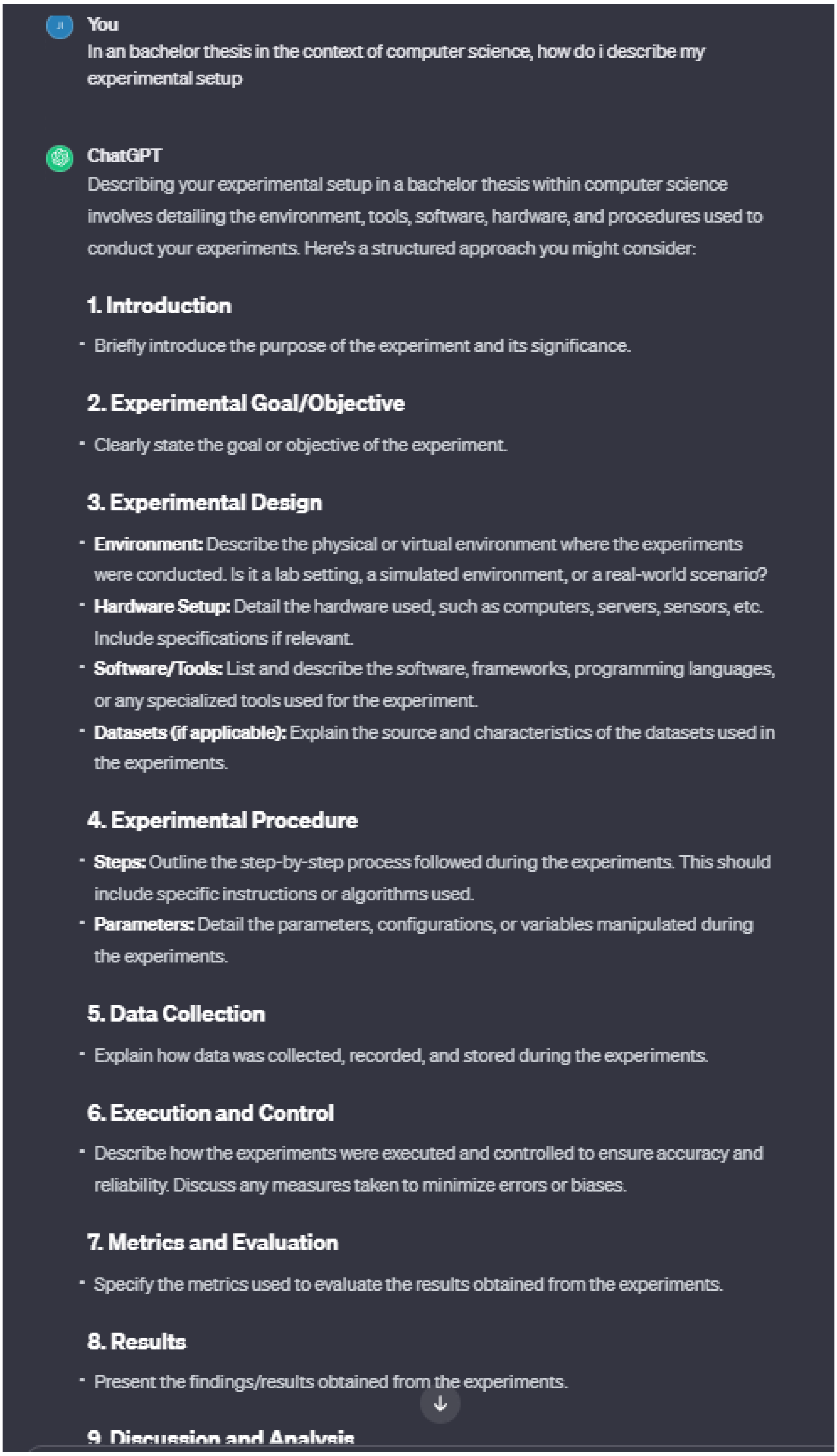
Here's an example of using an assertion to check if a variable has a specific length:

```python
def check_variable_length(var, expected_length):
    assert len(var) == expected_length, f"Variable length is not {expec

# Example usage
my_variable = "example"
expected_length = 8

try:
    check_variable_length(my_variable, expected_length)
    print("Variable length assertion passed!")
except AssertionError as e:
    print(f"Assertion Error: {e}")
    # Handle the assertion failure here if needed
```

In this example:

- `check_variable_length()` is a function that takes a `var` (variable) and an `expected_length`.
- The assertion inside this function checks if the length of `var` matches the `expected_length`.
- If the assertion fails, it raises an `AssertionError` with a custom error message indicating the expected and actual lengths.
- The `try-except` block is used to catch the `AssertionError` and print a custom error message. You can perform any specific handling or logging of the error within the `except` block.

Replace `my_variable` and `expected_length` with your specific variable and expected length. Adjust the assertion condition and error message to suit your requirements.



You

In an bachelor thesis in the context of computer science, how do i describe my experimental setup

ChatGPT

Describing your experimental setup in a bachelor thesis within computer science involves detailing the environment, tools, software, hardware, and procedures used to conduct your experiments. Here's a structured approach you might consider:

1. Introduction

- Briefly introduce the purpose of the experiment and its significance.

2. Experimental Goal/Objective

- Clearly state the goal or objective of the experiment.

3. Experimental Design

- **Environment:** Describe the physical or virtual environment where the experiments were conducted. Is it a lab setting, a simulated environment, or a real-world scenario?
- **Hardware Setup:** Detail the hardware used, such as computers, servers, sensors, etc. Include specifications if relevant.
- **Software/Tools:** List and describe the software, frameworks, programming languages, or any specialized tools used for the experiment.
- **Datasets (if applicable):** Explain the source and characteristics of the datasets used in the experiments.

4. Experimental Procedure

- **Steps:** Outline the step-by-step process followed during the experiments. This should include specific instructions or algorithms used.
- **Parameters:** Detail the parameters, configurations, or variables manipulated during the experiments.

5. Data Collection

- Explain how data was collected, recorded, and stored during the experiments.

6. Execution and Control

- Describe how the experiments were executed and controlled to ensure accuracy and reliability. Discuss any measures taken to minimize errors or biases.

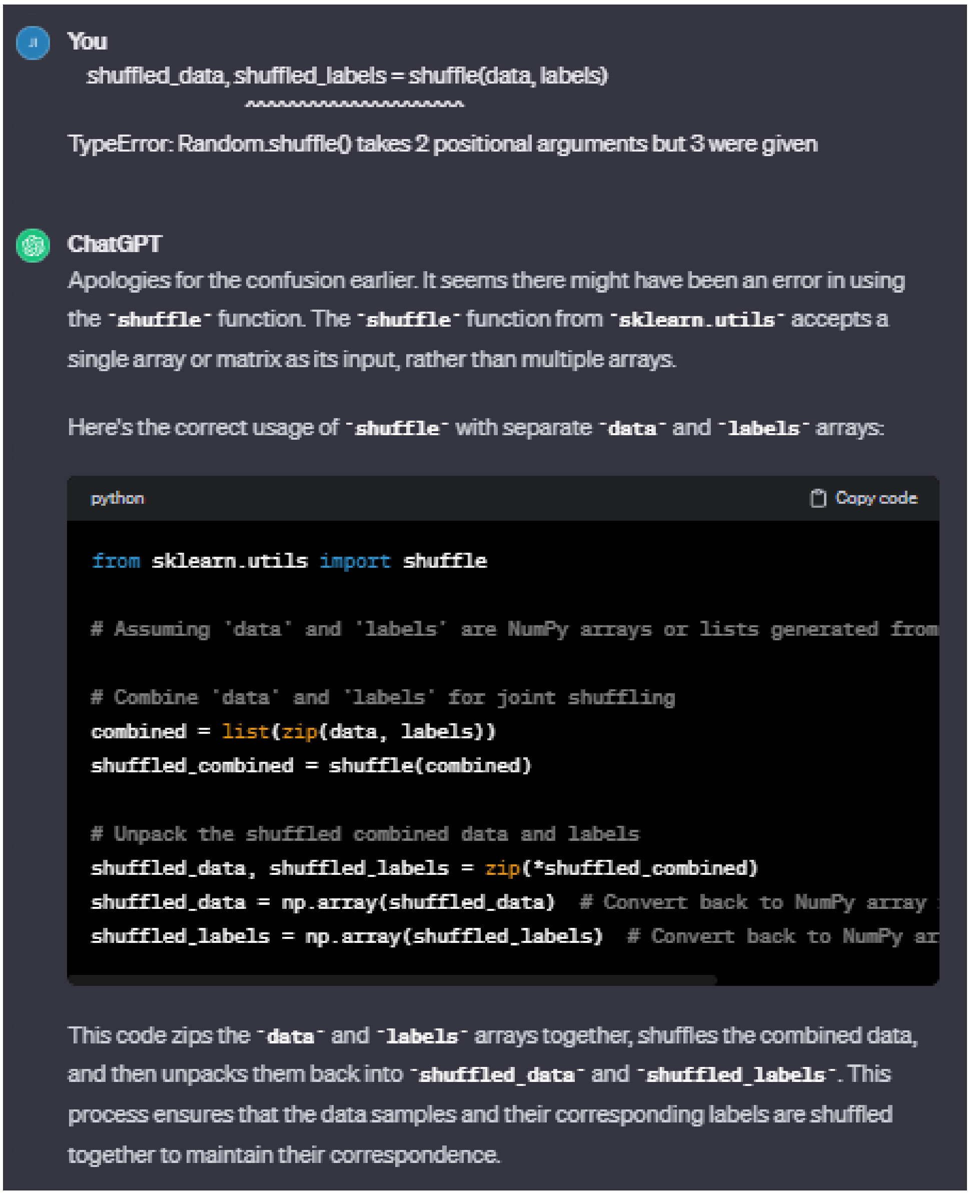
7. Metrics and Evaluation

- Specify the metrics used to evaluate the results obtained from the experiments.

8. Results

- Present the findings/results obtained from the experiments.

9. Discussion and Analysis



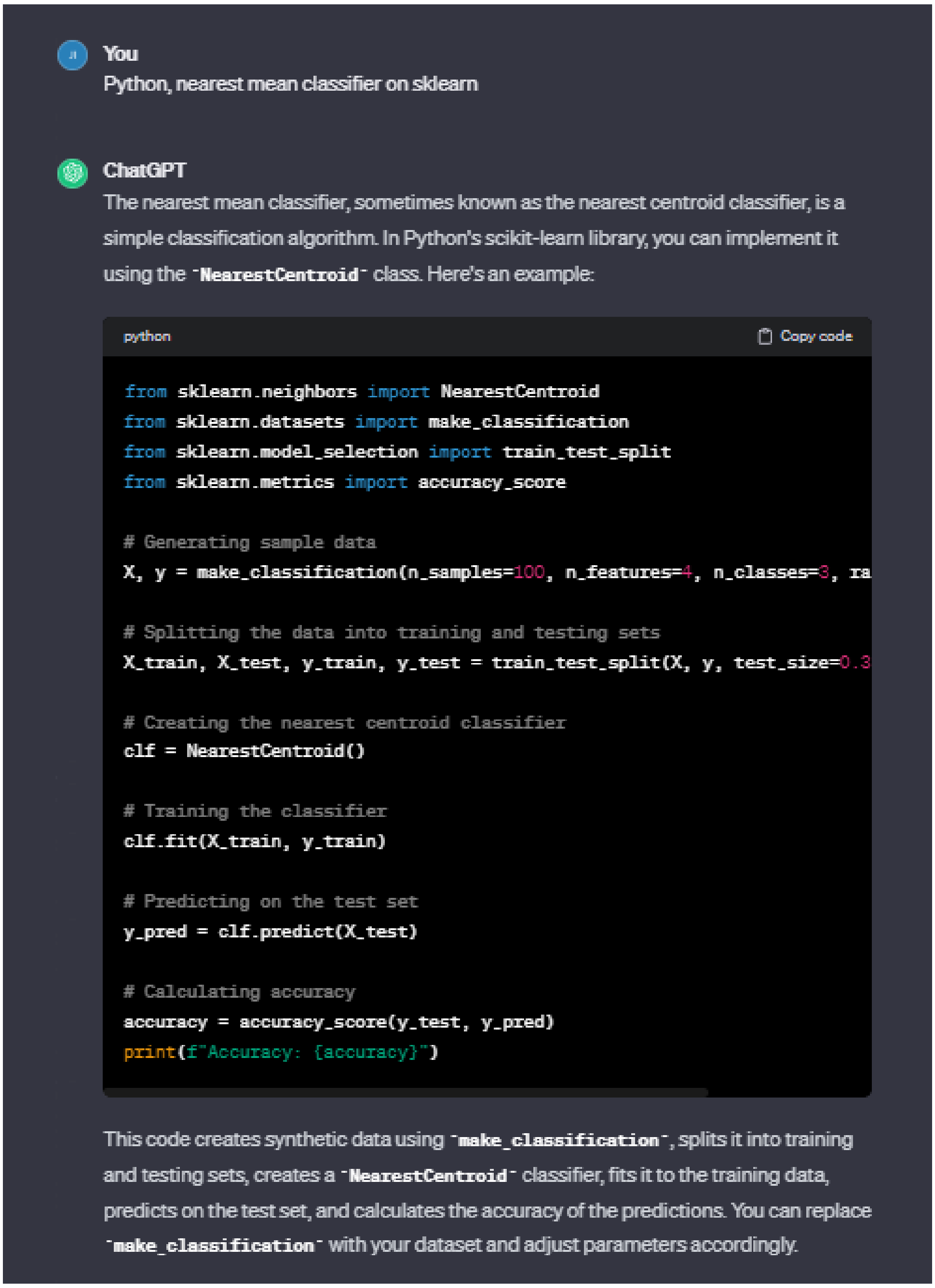
```
shuffled_data, shuffled_labels = shuffle(data, labels)
                                 ^^^^^^^^^^^^^^^^^^^^^

TypeError: Random.shuffle() takes 2 positional arguments but 3 were given
```

Apologies for the confusion earlier. It seems there might have been an error in using the `shuffle` function. The `shuffle` function from `sklearn.utils` accepts a single array or matrix as its input, rather than multiple arrays.

Here's the correct usage of `shuffle` with separate `data` and `labels` arrays:

```python
from sklearn.utils import shuffle

# Assuming `data` and `labels` are NumPy arrays or lists generated from

# Combine `data` and `labels` for joint shuffling
combined = list(zip(data, labels))
shuffled_combined = shuffle(combined)

# Unpack the shuffled combined data and labels
shuffled_data, shuffled_labels = zip(*shuffled_combined)
shuffled_data = np.array(shuffled_data)  # Convert back to NumPy array
shuffled_labels = np.array(shuffled_labels)  # Convert back to NumPy ar
```

This code zips the `data` and `labels` arrays together, shuffles the combined data, and then unpacks them back into `shuffled_data` and `shuffled_labels`. This process ensures that the data samples and their corresponding labels are shuffled together to maintain their correspondence.



You

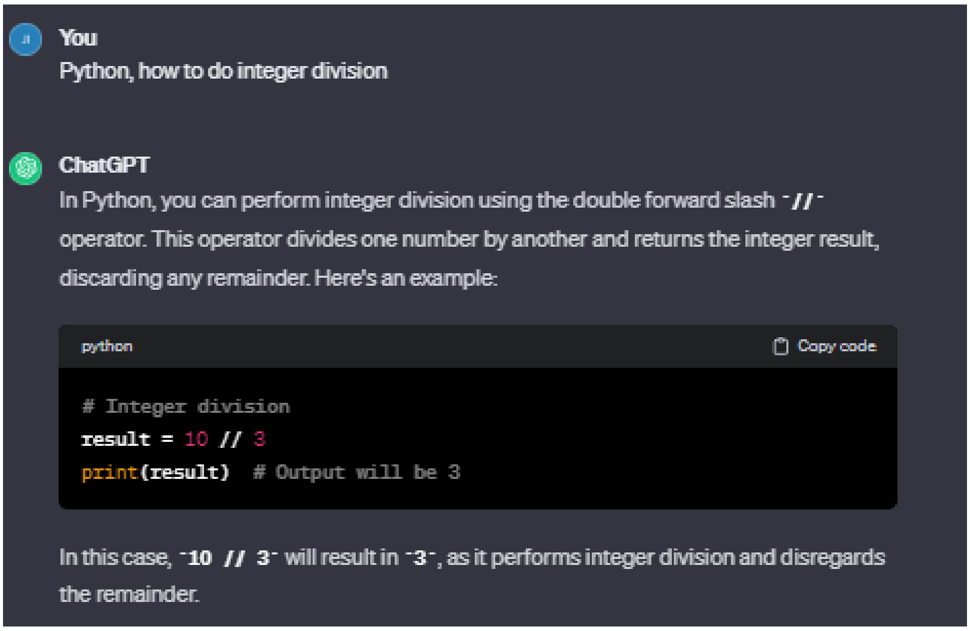
Python, nearest mean classifier on sklearn

ChatGPT

The nearest mean classifier, sometimes known as the nearest centroid classifier, is a simple classification algorithm. In Python's scikit-learn library, you can implement it using the `NearestCentroid` class. Here's an example:

```python
from sklearn.neighbors import NearestCentroid
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Generating sample data
X, y = make_classification(n_samples=100, n_features=4, n_classes=3, ra

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3

# Creating the nearest centroid classifier
clf = NearestCentroid()

# Training the classifier
clf.fit(X_train, y_train)

# Predicting on the test set
y_pred = clf.predict(X_test)

# Calculating accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

This code creates synthetic data using `make_classification`, splits it into training and testing sets, creates a `NearestCentroid` classifier, fits it to the training data, predicts on the test set, and calculates the accuracy of the predictions. You can replace `make_classification` with your dataset and adjust parameters accordingly.



You

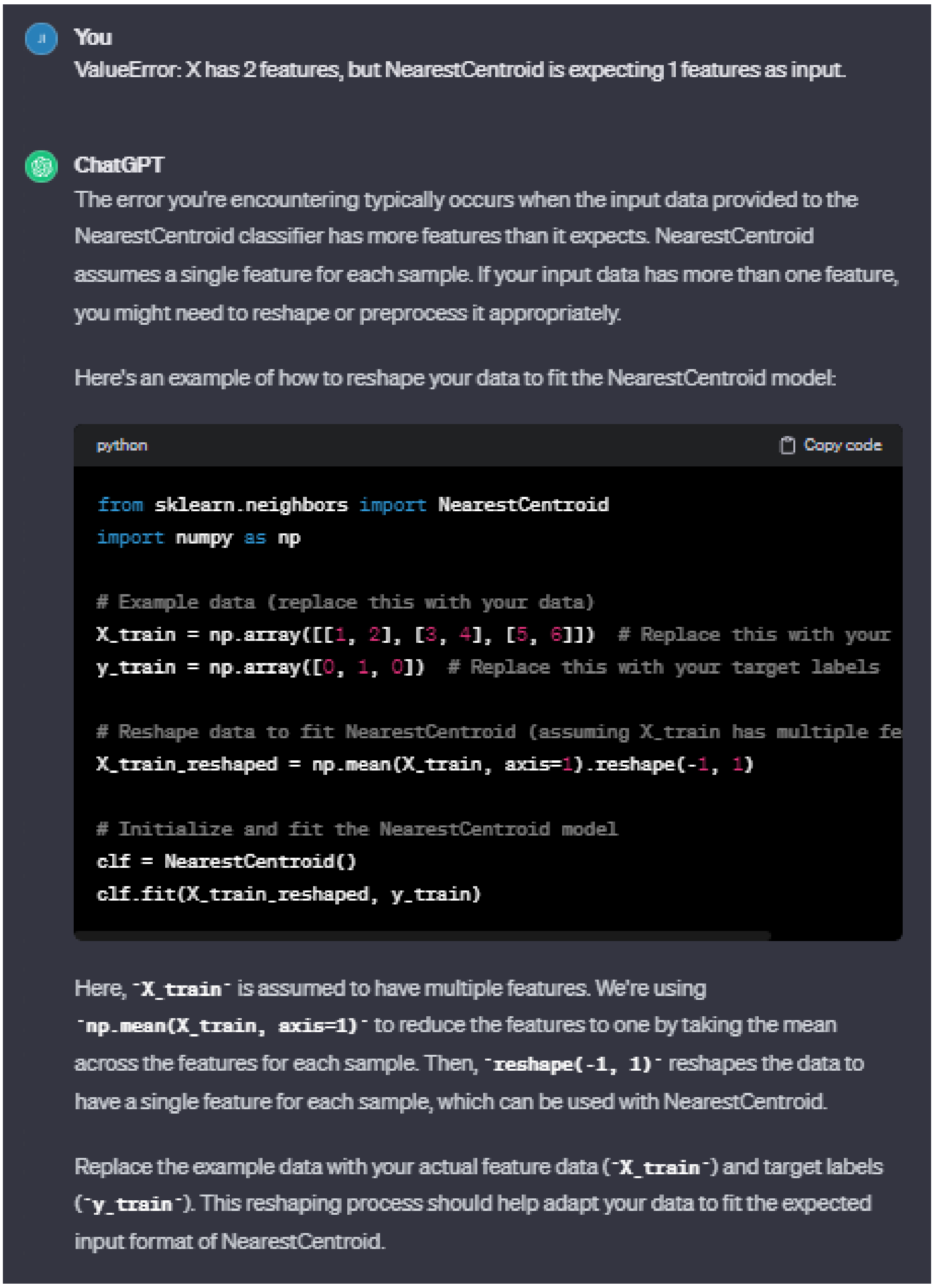
Python, how to do integer division

ChatGPT

In Python, you can perform integer division using the double forward slash `//` operator. This operator divides one number by another and returns the integer result, discarding any remainder. Here's an example:

```python
# Integer division
result = 10 // 3
print(result)  # Output will be 3
```

In this case, `10 // 3` will result in `3`, as it performs integer division and disregards the remainder.



ValueError: X has 2 features, but NearestCentroid is expecting 1 features as input.

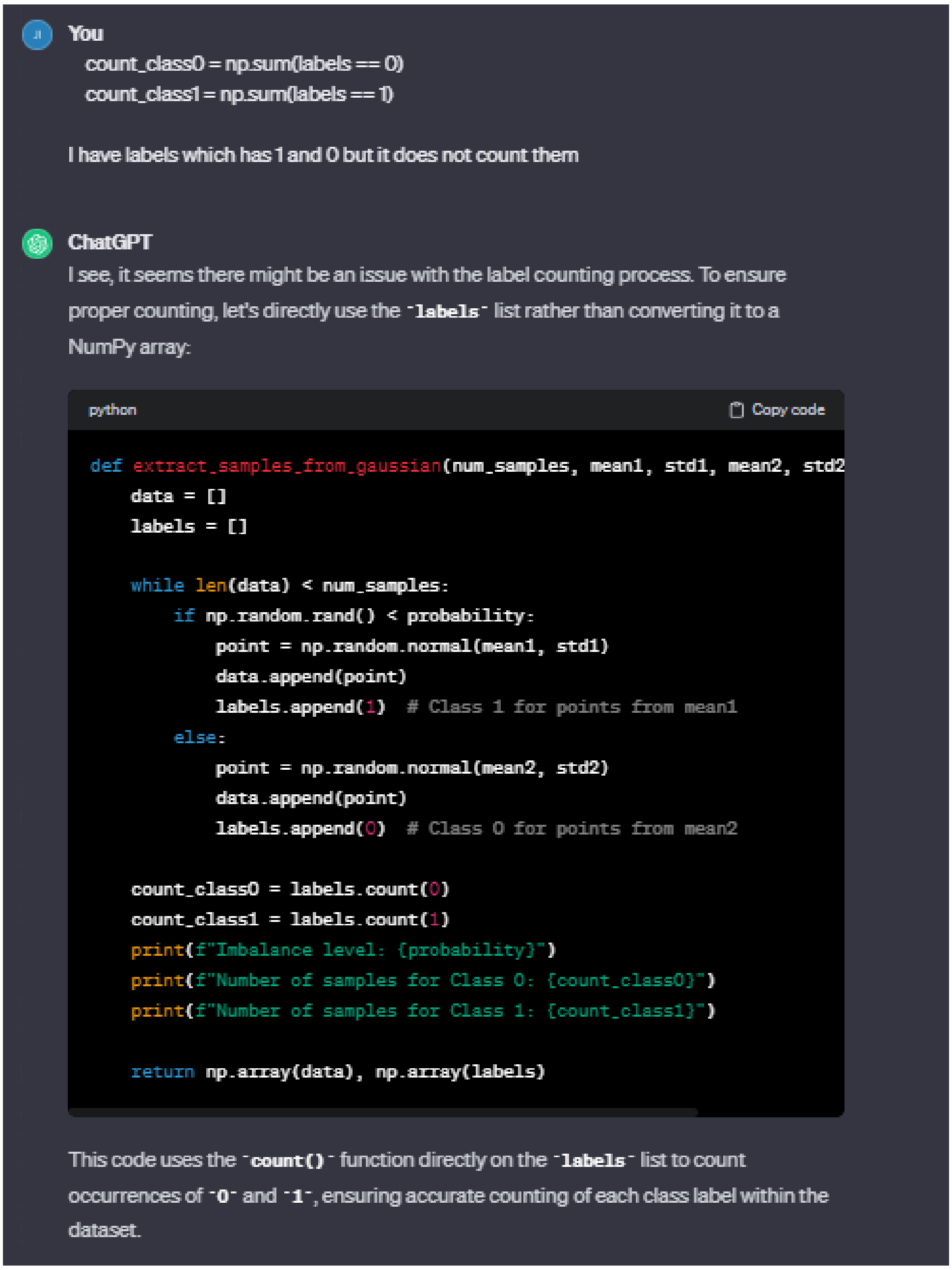
The error you're encountering typically occurs when the input data provided to the NearestCentroid classifier has more features than it expects. NearestCentroid assumes a single feature for each sample. If your input data has more than one feature, you might need to reshape or preprocess it appropriately.

Here's an example of how to reshape your data to fit the NearestCentroid model:

```python
from sklearn.neighbors import NearestCentroid
import numpy as np

# Example data (replace this with your data)
X_train = np.array([[1, 2], [3, 4], [5, 6]])  # Replace this with your
y_train = np.array([0, 1, 0])  # Replace this with your target labels

# Reshape data to fit NearestCentroid (assuming X_train has multiple fe
X_train_reshaped = np.mean(X_train, axis=1).reshape(-1, 1)

# Initialize and fit the NearestCentroid model
clf = NearestCentroid()
clf.fit(X_train_reshaped, y_train)
```

Here, `X_train` is assumed to have multiple features. We're using `np.mean(X_train, axis=1)` to reduce the features to one by taking the mean across the features for each sample. Then, `reshape(-1, 1)` reshapes the data to have a single feature for each sample, which can be used with NearestCentroid.

Replace the example data with your actual feature data (`X_train`) and target labels (`y_train`). This reshaping process should help adapt your data to fit the expected input format of NearestCentroid.



You

count_class0 = np.sum(labels == 0)
count_class1 = np.sum(labels == 1)

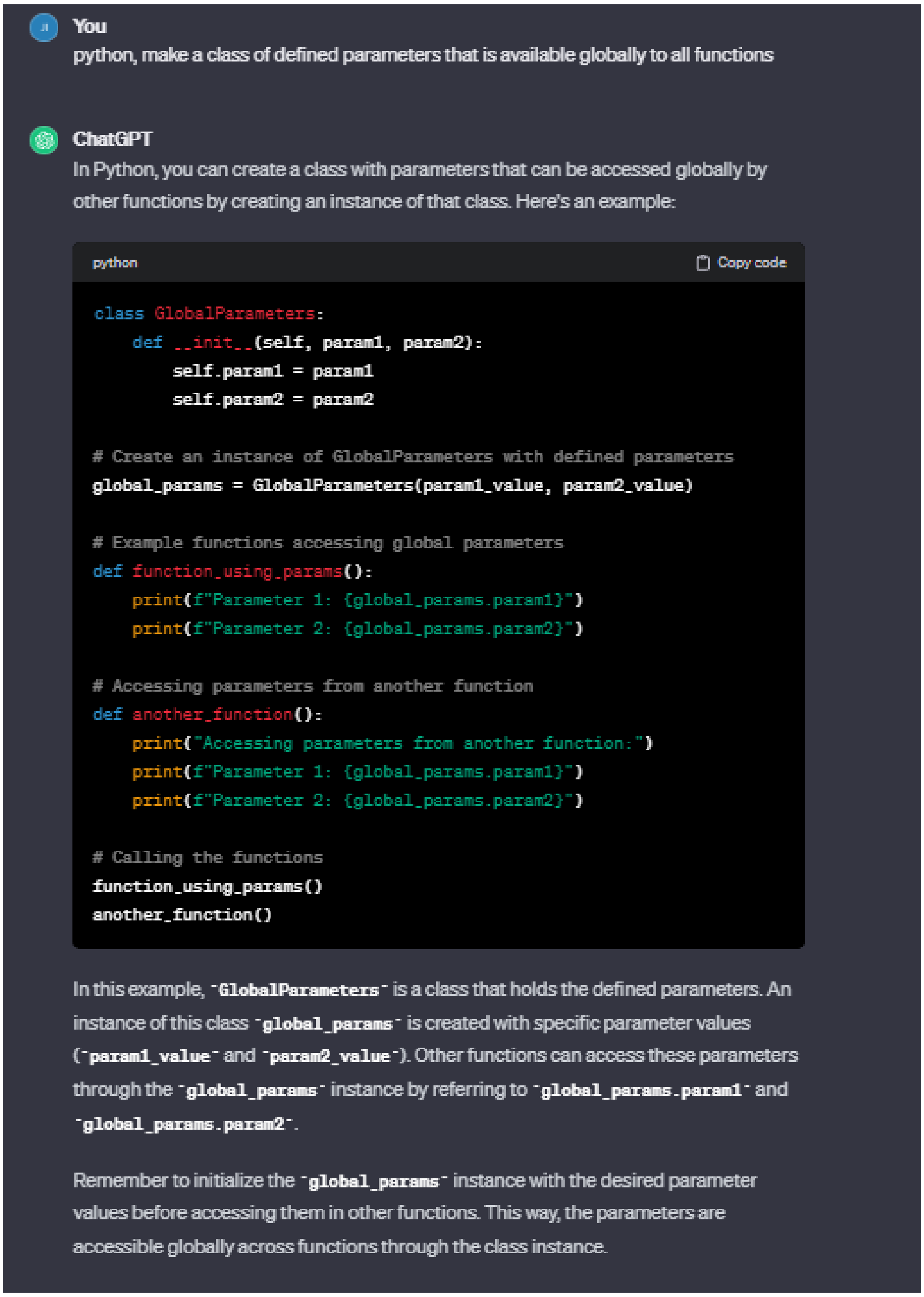
I have labels which has 1 and 0 but it does not count them

ChatGPT

I see, it seems there might be an issue with the label counting process. To ensure proper counting, let's directly use the `labels` list rather than converting it to a NumPy array:

```python
def extract_samples_from_gaussian(num_samples, mean1, std1, mean2, std2
    data = []
    labels = []

    while len(data) < num_samples:
        if np.random.rand() < probability:
            point = np.random.normal(mean1, std1)
            data.append(point)
            labels.append(1)  # Class 1 for points from mean1
        else:
            point = np.random.normal(mean2, std2)
            data.append(point)
            labels.append(0)  # Class 0 for points from mean2

    count_class0 = labels.count(0)
    count_class1 = labels.count(1)
    print(f"Imbalance level: {probability}")
    print(f"Number of samples for Class 0: {count_class0}")
    print(f"Number of samples for Class 1: {count_class1}")

    return np.array(data), np.array(labels)
```

This code uses the `count()` function directly on the `labels` list to count occurrences of `0` and `1`, ensuring accurate counting of each class label within the dataset.



You

python, make a class of defined parameters that is available globally to all functions

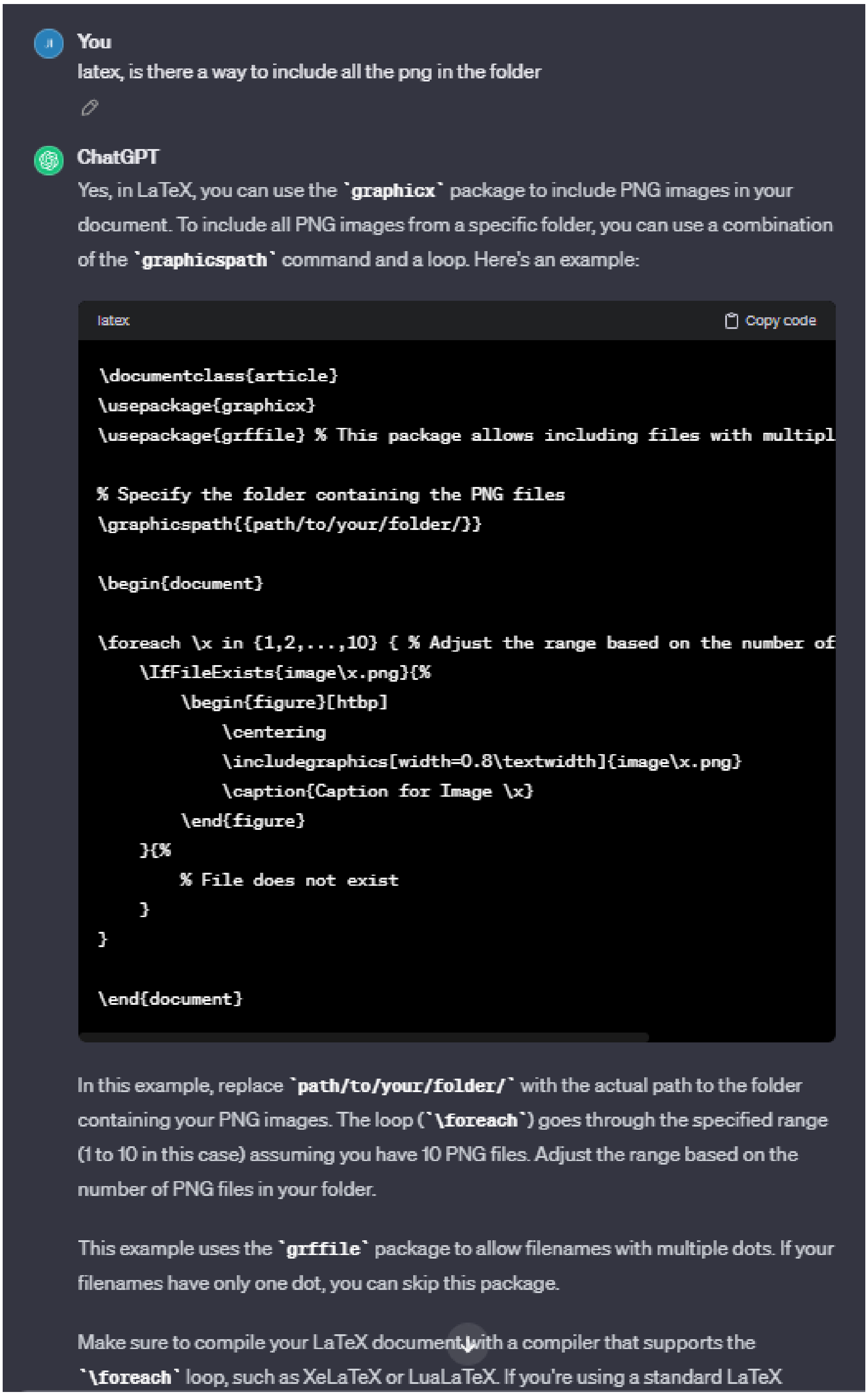
ChatGPT

In Python, you can create a class with parameters that can be accessed globally by other functions by creating an instance of that class. Here's an example:

```python
class GlobalParameters:
    def __init__(self, param1, param2):
        self.param1 = param1
        self.param2 = param2

# Create an instance of GlobalParameters with defined parameters
global_params = GlobalParameters(param1_value, param2_value)

# Example functions accessing global parameters
def function_using_params():
    print(f"Parameter 1: {global_params.param1}")
    print(f"Parameter 2: {global_params.param2}")

# Accessing parameters from another function
def another_function():
    print("Accessing parameters from another function:")
    print(f"Parameter 1: {global_params.param1}")
    print(f"Parameter 2: {global_params.param2}")

# Calling the functions
function_using_params()
another_function()
```

In this example, `GlobalParameters` is a class that holds the defined parameters. An instance of this class `global_params` is created with specific parameter values (`param1_value` and `param2_value`). Other functions can access these parameters through the `global_params` instance by referring to `global_params.param1` and `global_params.param2`.

Remember to initialize the `global_params` instance with the desired parameter values before accessing them in other functions. This way, the parameters are accessible globally across functions through the class instance.



You

latex, is there a way to include all the png in the folder

ChatGPT

Yes, in LaTeX, you can use the `graphicx` package to include PNG images in your document. To include all PNG images from a specific folder, you can use a combination of the `graphicspath` command and a loop. Here's an example:

```latex
\documentclass{article}
\usepackage{graphicx}
\usepackage{grffile} % This package allows including files with multipl

% Specify the folder containing the PNG files
\graphicspath{{path/to/your/folder/}}

\begin{document}

\foreach \x in {1,2,...,10} { % Adjust the range based on the number of
    \IfFileExists{image\x.png}{%
        \begin{figure}[htbp]
            \centering
            \includegraphics[width=0.8\textwidth]{image\x.png}
            \caption{Caption for Image \x}
        \end{figure}
    }{%
        % File does not exist
    }
}

\end{document}
```

In this example, replace `path/to/your/folder/` with the actual path to the folder containing your PNG images. The loop (`\foreach`) goes through the specified range (1 to 10 in this case) assuming you have 10 PNG files. Adjust the range based on the number of PNG files in your folder.

This example uses the `grffile` package to allow filenames with multiple dots. If your filenames have only one dot, you can skip this package.

Make sure to compile your LaTeX document with a compiler that supports the `\foreach` loop, such as XeLaTeX or LuaLaTeX. If you're using a standard LaTeX