

Tutorial

An Automated Topological Framework for Robust Multi-Scale Data Analysis Using Persistent Homology and Mapper Algorithms

Naseem Zoha Ansari¹, Subhajit Bandyopadhyay², Sudarshan Gogoi¹, Mubashar Rashid³, Amit Chakraborty^{*1}

¹Department of Mathematics, Sikkim University, Gangtok, 737102, Sikkim, India.

²Department of Computer Applications, Sikkim University, Gangtok, 737102, Sikkim, India

³Department of Mathematics, Indian Institute of Technology, Kanpur, India.

- ① How to install Python in Anaconda Navigator
- ② Tutorial for Coordinate points
- ③ Tutorial for Correlation Matrix

① How to install Python in Anaconda Navigator

② Tutorial for Coordinate points

③ Tutorial for Correlation Matrix

Step 1 : Open Anaconda Navigator

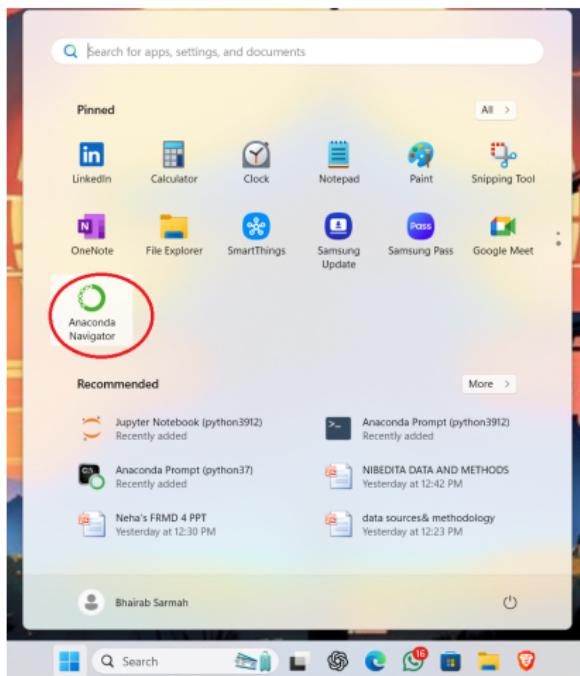


Figure 1: Select the Anaconda Navigator file from the destination or in Strat menu

Step 2 : Selecting Python 3.9.21 for our program use

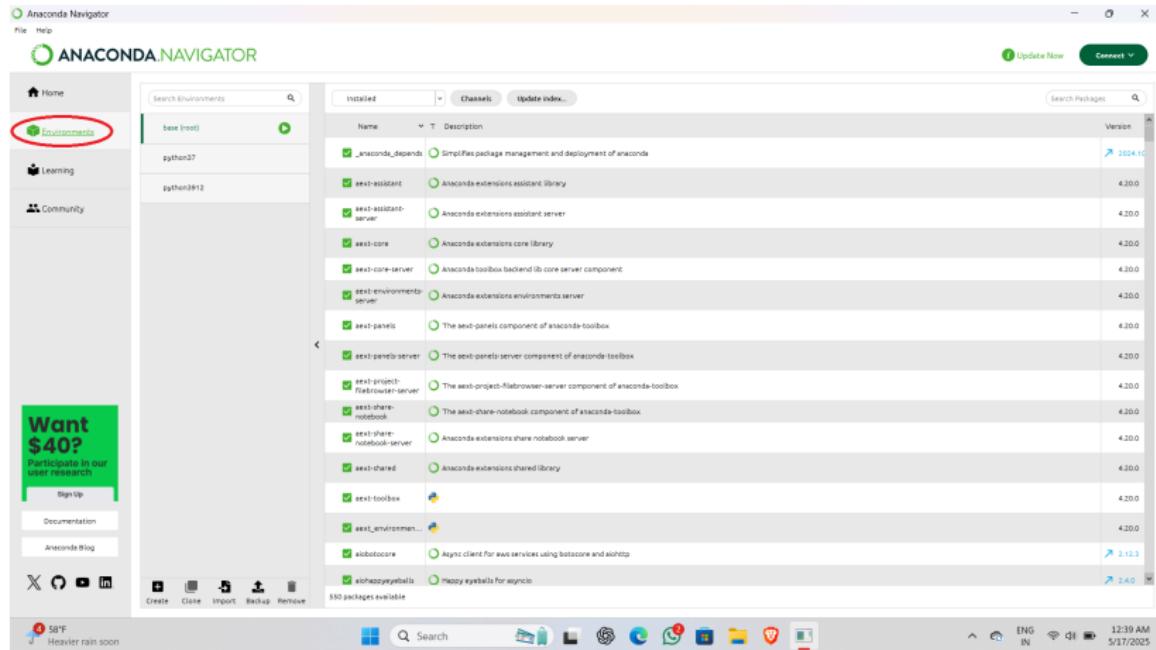


Figure 2: Click on the Environment tab just below Home tab

Step 2 : Selecting Python 3.9.21 for our program use

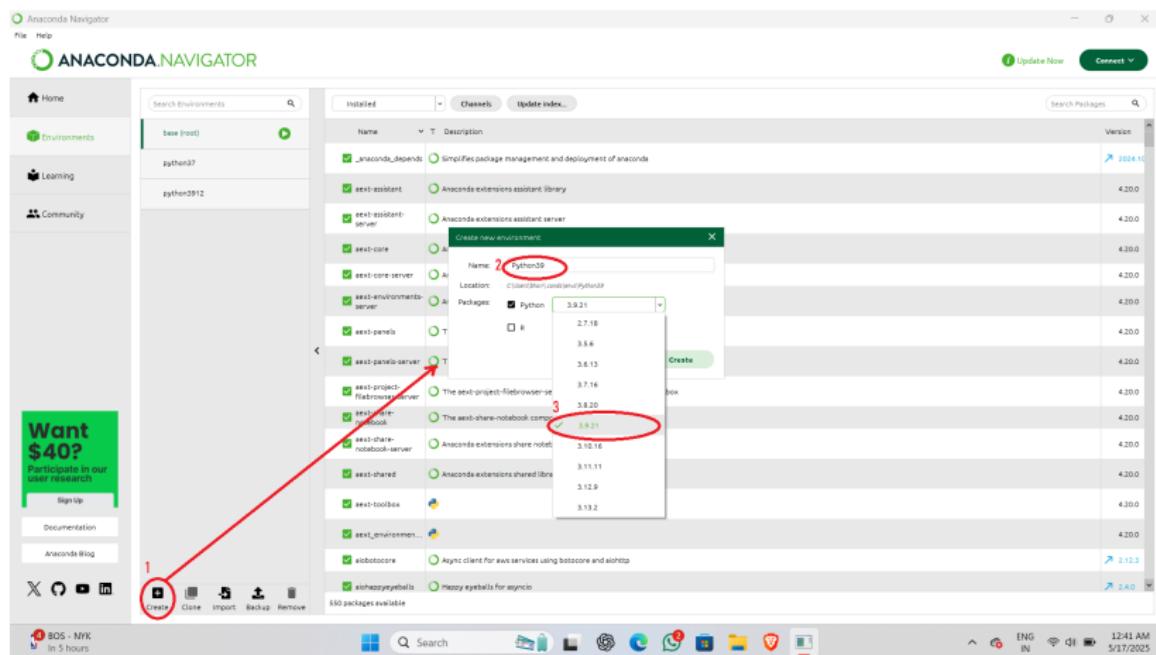


Figure 3: Click Create, name your environment in the dialog box that appears, select Python 3.9.21

Step 2 : Selecting Python 3.9.21 for our program use

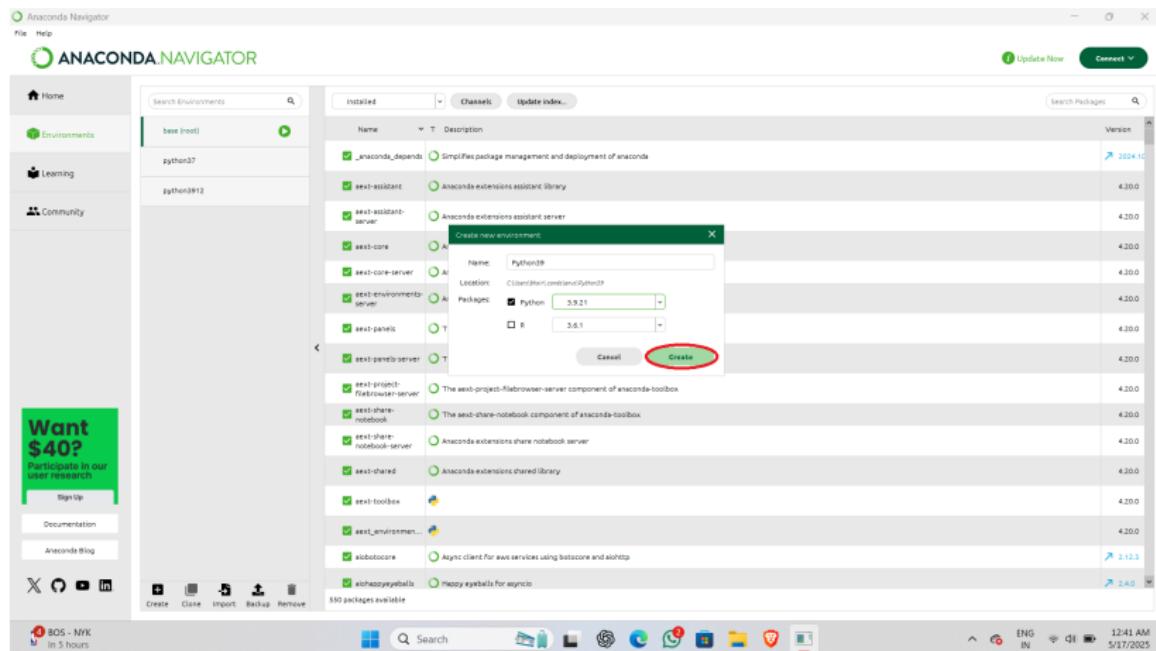


Figure 4: Hit on create button to confirm

Step 2 : Selecting Python 3.9.21 for our program use

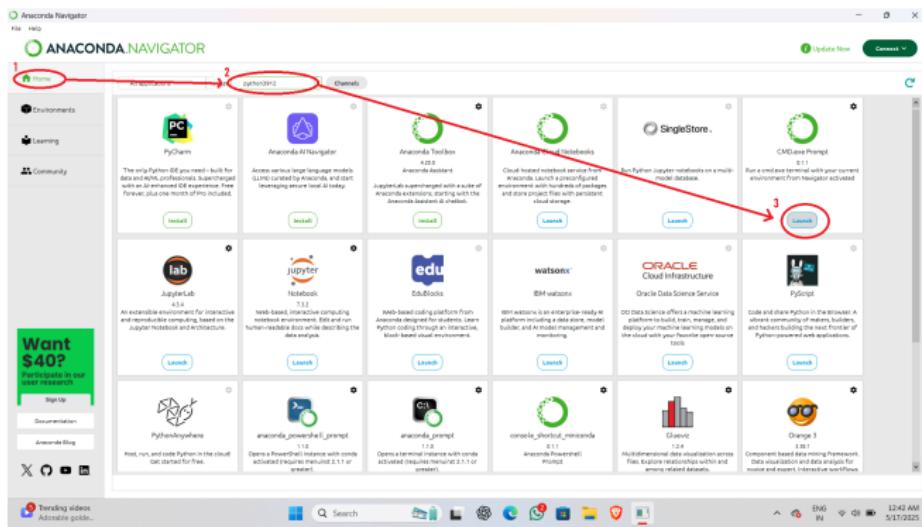


Figure 5: Navigate to the **Home** tab, verify the created environment under the *Applications on* section, and launch **CMD.exe** to install the required packages.

Step 3 : Installation of the required packages

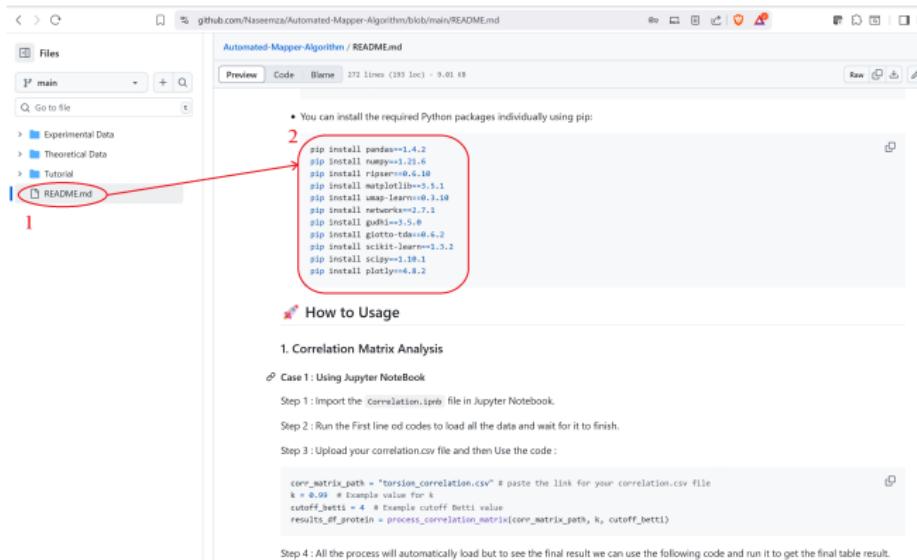


Figure 6: Visit the GitHub repository

<https://github.com/Naseemza/Advance-Mapper-Algorithm>, open the README.md file to identify the required package name, and use the cmd.exe dialog box to install all compatible packages using pip.

Step 3 : Installation of the required packages

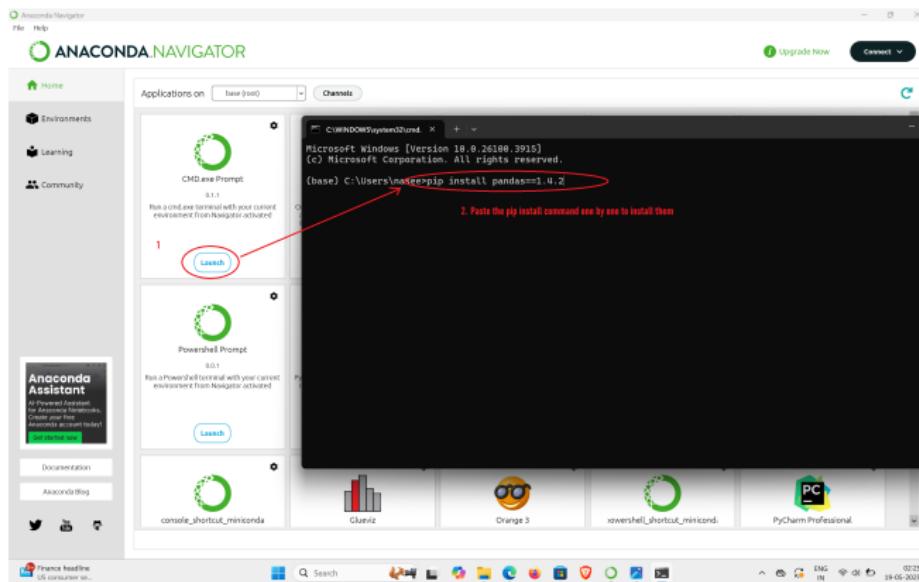


Figure 7: Open Anaconda Navigator, launch the Command Prompt (CMD.exe), and sequentially execute each pip command to install the required packages.

① How to install Python in Anaconda Navigator

② Tutorial for Coordinate points

③ Tutorial for Correlation Matrix

Step 1 : Download the Torus.ipynb file

The screenshot shows a Jupyter Notebook interface with the following details:

- File Explorer:** On the left, it shows a tree view of files. A red circle labeled "1" highlights the "Theoretical Data" folder. Inside "Theoretical Data", there are three files: "Double Torus.ipynb", "Torus.ipynb" (which is selected), and "Triple Overlapping Circle.ipynb". A red circle labeled "2" highlights "Torus.ipynb".
- Code Cell:** In the main area, a code cell titled "In [1]" contains Python code related to Mapper algorithms. The code includes imports for numpy, ripser, matplotlib, vmap, networkx, gdal, gtds, and various plotting functions from matplotlib and plotly.
- Download Button:** In the top right corner of the code cell, there is a "Download" button represented by a downward arrow icon. A red circle labeled "3" highlights this button.

Figure 8: Click on the **Theoretical Data** section, open the **Torus.ipynb** file (or any other), and click **Download** to save it from <https://github.com/Naseemza/Advance-Mapper-Algorithm>.

Step 1 : Download the Torus.ipynb file

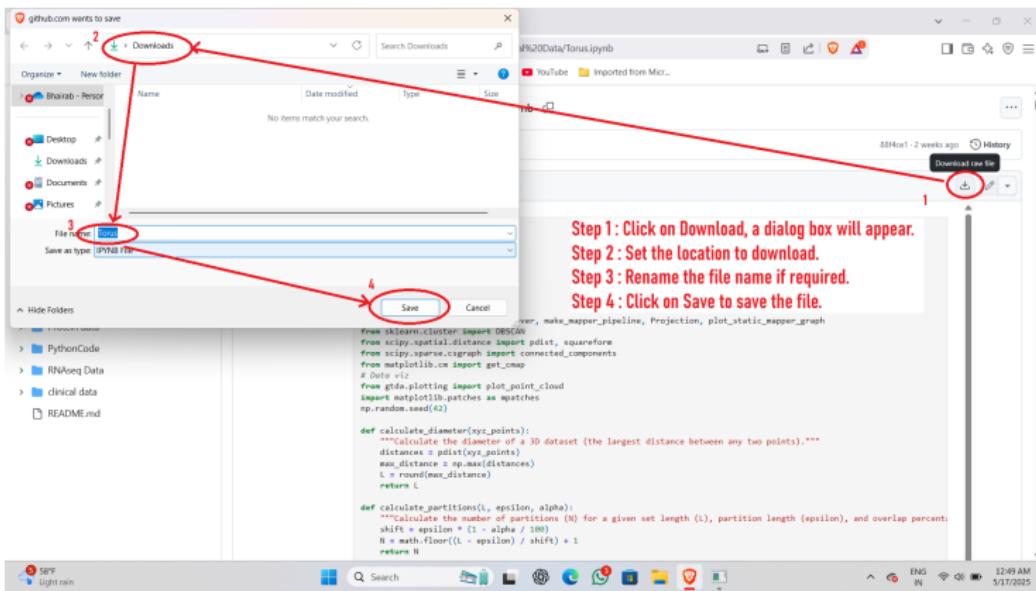


Figure 9: The process for downloading a file: click, locate, rename and save

Step 2 : Uploading the .ipynb file

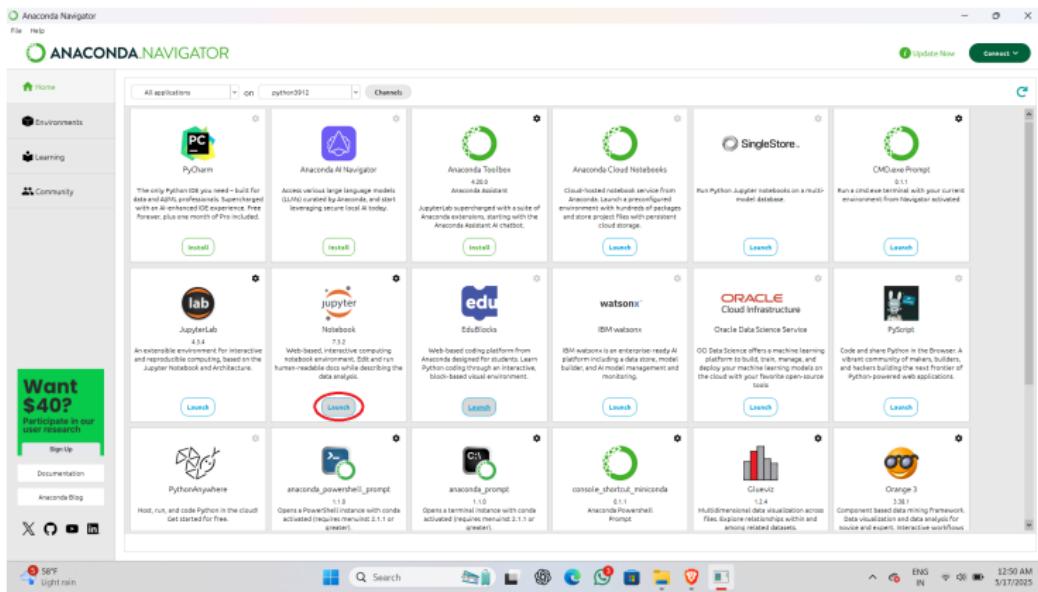


Figure 10: Open Jupyter Notebook and wait for it to load from Anaconda Navigator application

Step 2 : Uploading the .ipynb file

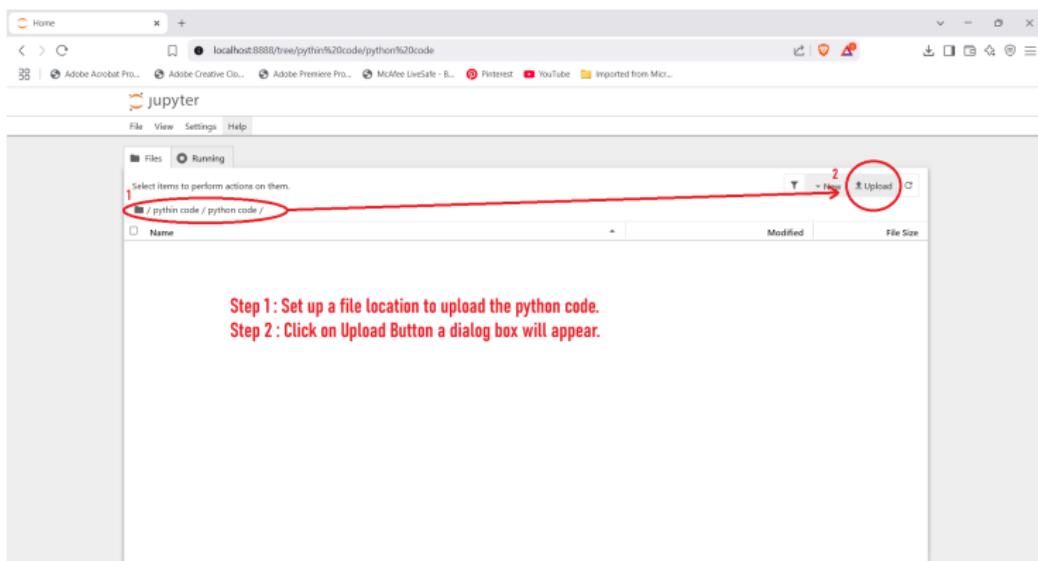


Figure 11: In the **Home** tab of the browser, navigate to the desired location, then click **Upload** to select and upload the previously downloaded file.

Step 3 : Lunch the .ipynb file

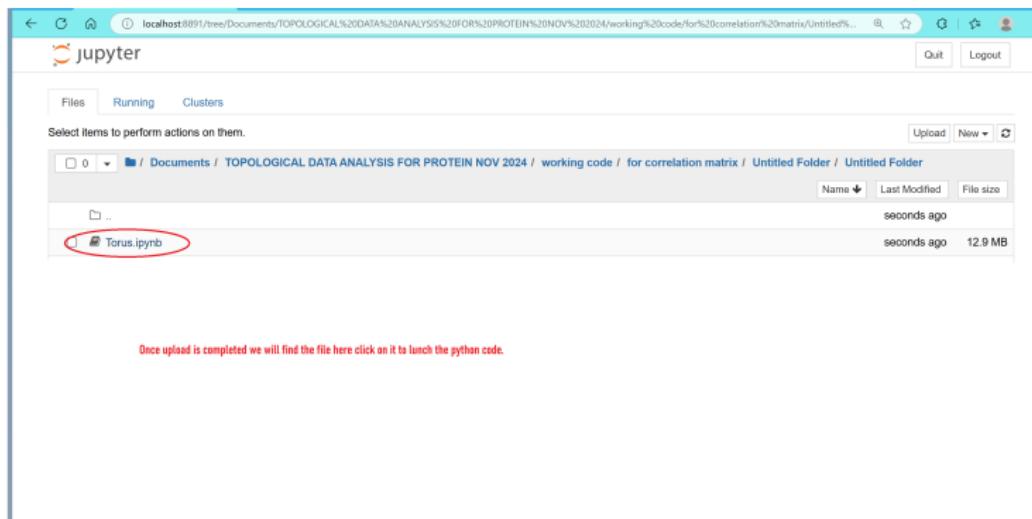


Figure 12: Click on the uploaded file to lunch the python code

Step 4 : Run the python code

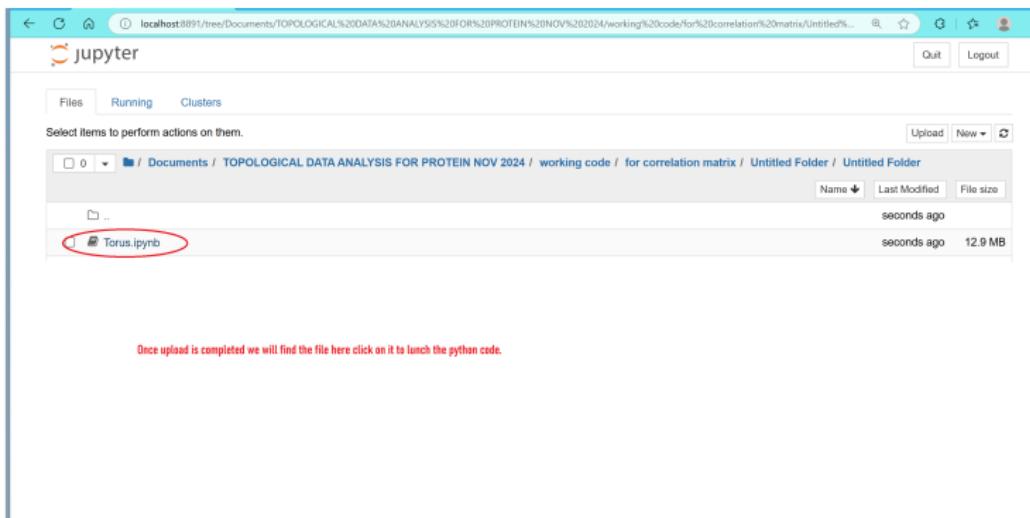


Figure 13: Click on the uploaded file to lunch the python code

Step 4 : Run the python code

The screenshot shows a Jupyter Notebook interface with the title "Torus - Jupyter Notebook". The URL in the address bar is "localhost:8851/notebooks/14/Documents/TOPLOGICAL%20DATA%20ANALYSIS%20FOR%20PROTEIN%20NOV%202024/working%20code/for%20correlation%20matrix.ipynb". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar below has icons for New, Open, Save, Run, Cell, Kernel, Help, and Code. A status bar indicates "Not Trusted" and "Python 3 (ipykernel)". The main area shows code in cell [1]:

```
In [1]:  
1 import pandas as pd  
2 import numpy as np  
3 import ripser  
4 import matplotlib.pyplot as plt  
5 import umap  
6 import math  
7 import networkx as nx  
8 import gdal as gd  
9 from gtda.mapper import CubicalCover, make_mapper_pipeline, Projection, plot_static_mapper_graph  
10 from sklearn.cluster import DBSCAN  
11 from scipy.spatial.distance import pdist, squareform  
12 from scipy.sparse.csgraph import connected_components  
13 from matplotlib.cm import get_cmap  
14 # Data viz  
15 from gtda.plotting import plot_point_cloud  
16 import matplotlib.patches as mpatches  
17 np.random.seed(42)  
18  
19 def calculate_diameter(xyz_points):  
20     """Calculate the diameter of a 3D dataset (the largest distance between any two points)."""  
21     distances = pdist(xyz_points)  
22     max_distance = np.max(distances)  
23     L = round(max_distance)  
24     return L  
25
```

Annotations in red:

1. New tab will open.
2. Click anywhere on the cell [1].
3. click on Run to load the preliminaries of the code, wait for the process to stop.

Figure 14: A new tab will open. Click on cell [1], then click Run to load the preliminaries of the code. Wait for the process to complete before proceeding.

Step 4 : Run the python code

```
In [2]:  
1 # Example usage  
2 import numpy as np  
3 import matplotlib.pyplot as plt  
4 from mpl_toolkits.mplot3d import Axes3D  
5 import gdudhi as gd  
6  
7 # Define the torus parametric equations  
8 def torus(u, v, R=3, r=1):  
9     x = (R + r * np.cos(v)) * np.cos(u)  
10    y = (R + r * np.cos(v)) * np.sin(u)  
11    z = r * np.sin(v)  
12    return np.column_stack((x, y, z))  
13  
14 # Parameters  
15 n_points = 10000 # Number of points  
16 R = 4 # Major radius  
17 r = 2 # Minor radius  
18  
19 # Generate a regular grid of u and v values  
20 u = np.linspace(0, 2 * np.pi, n_points)  
21 v = np.linspace(0, 2 * np.pi, n_points)  
22 u, v = np.meshgrid(u, v)  
23 u = u.flatten()  
24 v = v.flatten()  
25
```

Step 1: click anywhere in cell 2.

Step 2: Press on run to run the code.

Note : Here we are loading random value for our structural data so if required we can directly input the coordinate point as array here.

Figure 15: Follow the process to load the coordinate data.

Step 4 : Run the python code

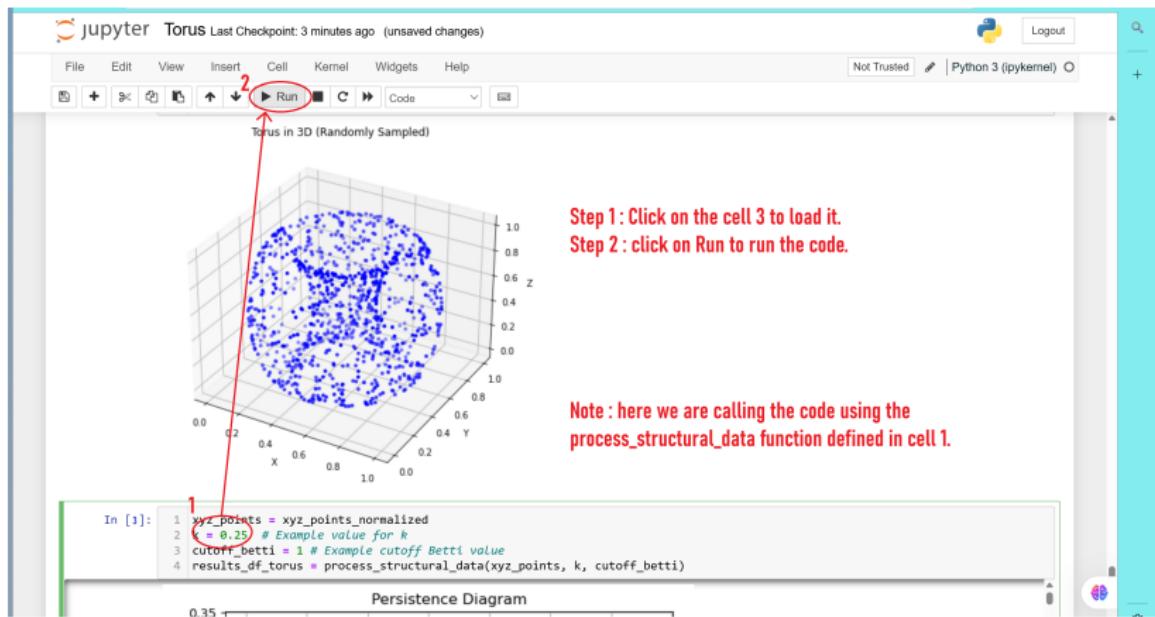


Figure 16: Follow the process to run the code for your data.

Step 5 : Print thr final result

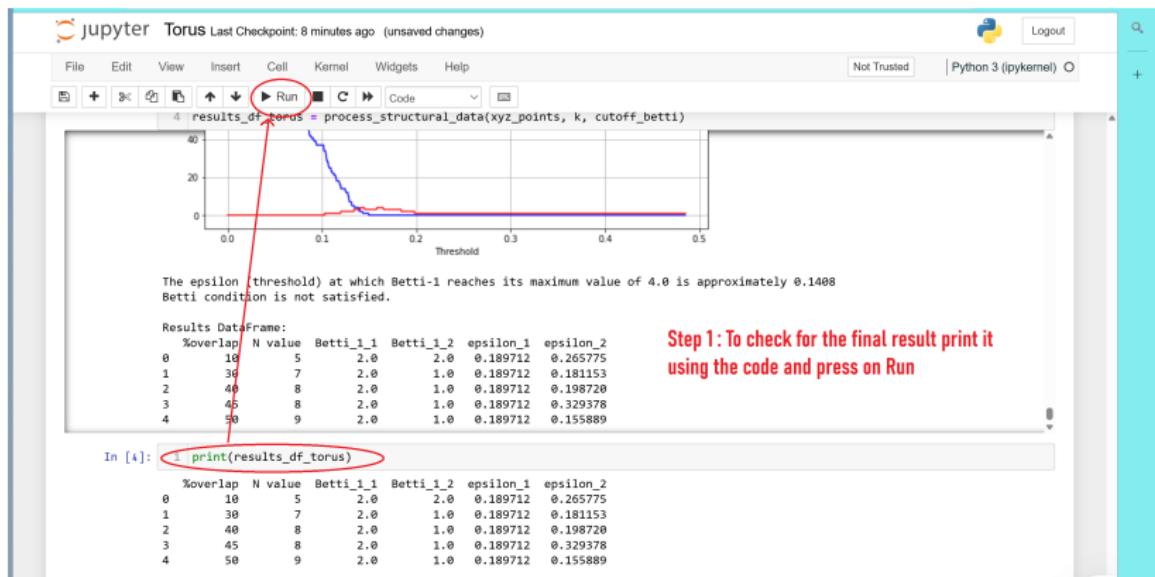


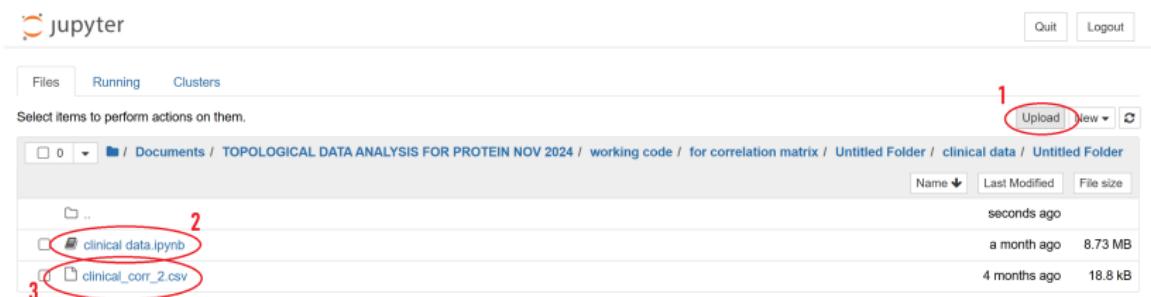
Figure 17: Follow the process to get the final result table.

① How to install Python in Anaconda Navigator

② Tutorial for Coordinate points

③ Tutorial for Correlation Matrix

Step 1 : Uploading the .ipynb and .csv file



1. Upload the clinical_data.ipynb file along with the correlation matrix .csv as done for coordinate points.
2. Click on the .ipynb file to run it and wait for new tab to open.

Figure 18: We follow the same steps of downloading the files and then similarly as coordinate files we upload the files here.

Step 2 : Run the code

The screenshot shows a Jupyter Notebook interface with the title "jupyter clinical data Last Checkpoint: 29/04/2025 (autosaved)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3 (ipykernel), and Logout. A red circle highlights the "Run" button in the toolbar. A red arrow points from the text "Step 1: Click on cell 1 and the press Run." to the "In [1]" cell. The code cell contains Python code for data visualization and clustering:

```
In [1]: 1 import pandas as pd
2 import numpy as np
3 import ripser
4 import matplotlib.pyplot as plt
5 import umap
6 import math
7 import networkx as nx
8 import gudhi as gd
9 from gtda.mapper import CubicalCover, make_mapper_pipeline, plot_static_mapper_graph
10 from sklearn.cluster import DBSCAN
11 from scipy.spatial.distance import pdist, squareform
12 from scipy.sparse.csgraph import connected_components
13 from matplotlib.cm import get_cmap
14 # Data viz
15 from gtda.plotting import plot_point_cloud
16 import matplotlib.patches as mpatches
17 np.random.seed(42)
18
19 def calculate_diameter(xyz_points):
20     """Calculate the diameter of a 3D dataset (the largest distance between any two points)."""
21     distances = pdist(xyz_points)
22     max_distance = np.max(distances)
23     L = round(max_distance)
24     return L
25
26 def calculate_partitions(L, epsilon, alpha):
27     """Calculate the number of partitions (N) for a given set length (L), partition length (epsilon), and overlap percentage
28     shift = epsilon * (1 - alpha / 100)
29     N = math.floor((L - epsilon) / shift) + 1
30     return N
31
```

Figure 19: We run the code step by step

Step 3 : Print the final result

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** jupyter clinical data Last Checkpoint: 29/04/2025 (unsaved changes)
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help. The "Run" button is highlighted with a red circle and labeled "1".
- Code Cell (In [3]):**

```
1 # Example usage
2 corr_matrix_path = 'D:\\correlation matrix sudarsan\\CorrMatrix\\ClinicalBreastData\\correlation_matrix.csv'
3 k = 0.84 # Example value for k
4 cutoff_betti = 0.5 # Example cutoff Betti value
5 results_df_clinical = process_correlation_matrix(corr_matrix_path, k, cutoff_betti)
```

A warning message is displayed in a red box:
C:\Users\nasee\anaconda3\lib\site-packages\ripsr\ripsr.py:247: UserWarning: The input matrix is square, but the distance_matrix flag is off. Did you mean to indicate that this was a distance matrix?
warnings.warn(
- Persistence Diagram Plot:** A scatter plot titled "Persistence Diagram" showing points colored by homology dimension (HD in blue, H1 in orange). The x-axis is "Dimension" (0.0 to 14) and the y-axis is "Depth" (0.9 to 1.4).
- Code Cell (In [4]):**

```
1 print(results_df_clinical)
```

	MOverlap	N	value	Betti_1_1	Betti_1_2	epsilon_1	epsilon_2
0	50	29	2.0	2.0	1.306469	1.399681	
1	55	32	2.0	2.0	1.306469	0.576101	Start

Figure 20: Follow the process to get the final result table. We need to double check for the .csv file and put the path in the section to be read automatically by the code.

Thanks For Your Attention!