IIIT Bangalore

26/C, Opposite of Infosys gate 1 Electronics City Phase 1, Hosur Road Bengaluru - 560100

Research Internship Report

Project Title:

Design an Algorithm for Dynamic Data Clustering using TDA and ML

Submitted by:

Pranshu Kumar Mishra

Submitted to:

Dr. Amit Chattopadhyay

Dr. Suman Saha

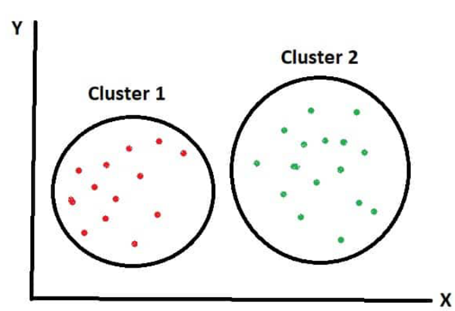
1st June – 3rd August 2023

ABSTRACT

The integration of topological data analysis (TDA) with machine learning (ML) for the dynamic clustering of complicated datasets is summarized in this internship report. The study shows how cutting-edge ML algorithms are combined with TDA's mathematical framework to address the issues presented by dynamic data. The study demonstrates the effectiveness of this hybrid technique in capturing subtle patterns and temporal transitions through methodical research on a variety of datasets. The study identifies possible domain-specific applications that could improve the precision and interpretability of cluster analysis. Overall, the paper highlights how TDA and ML work together to create effective dynamic data clustering solutions.

INTRODUCTION

What is Clustering?
Clustering is the process of dividing datasets into clusters(groups) with similar data points(features) and each cluster is represented by its unique cluster id.



Applications of Clustering:

- Market monitoring
- Social media analysis
- E-Commerce/OTT App Recommendation Algorithm
- Image Recognition in Traffic Control System

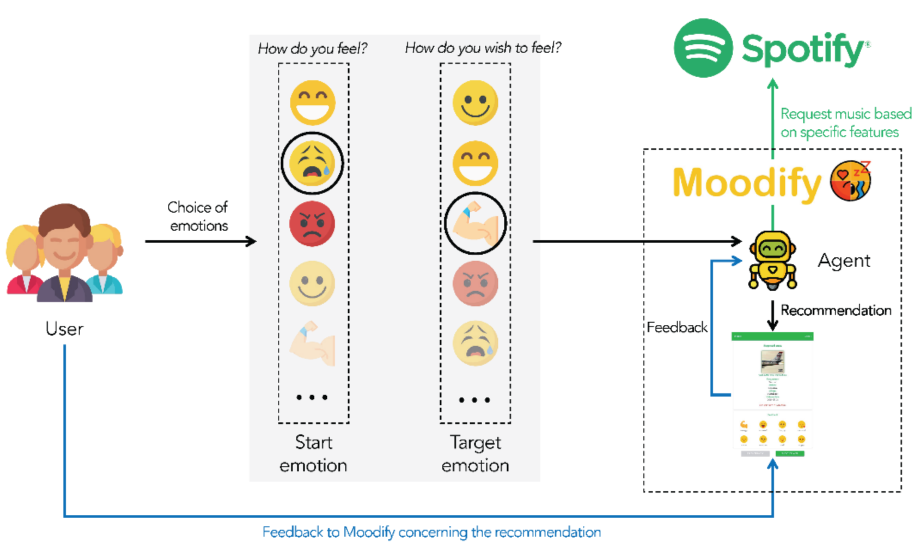
For a better clustering analysis, **K-Means Algorithm** has been used.

Difference between K-Means and Dynamic K-Means Algorithms:

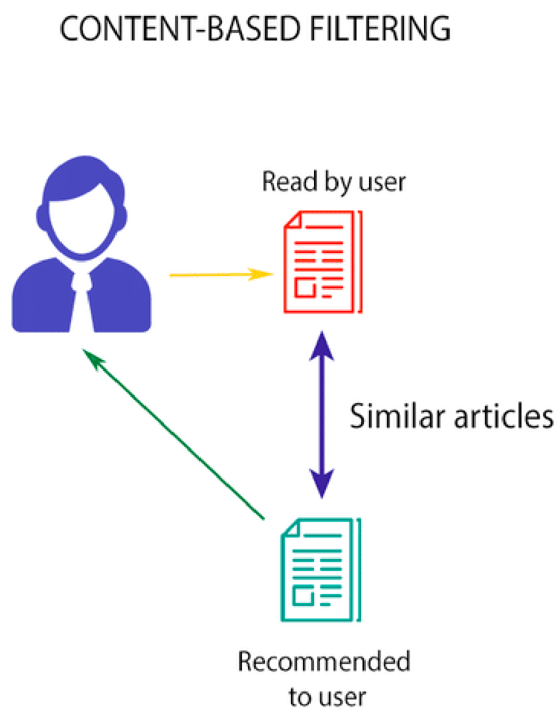
- In K-Means, K (no. of clusters) is defined by the user whereas Dynamic K-Means automatically determines the number of clusters.
- Also, K-Means operates over a fixed dataset while the Dynamic K-Means operates over a changing dataset.
- K-Means has no convergence criteria whereas Dynamic K-Means operates over a convergence criterion to ensure the stability of dataset.

Real Life Applications of Dynamic K-Means Algorithms

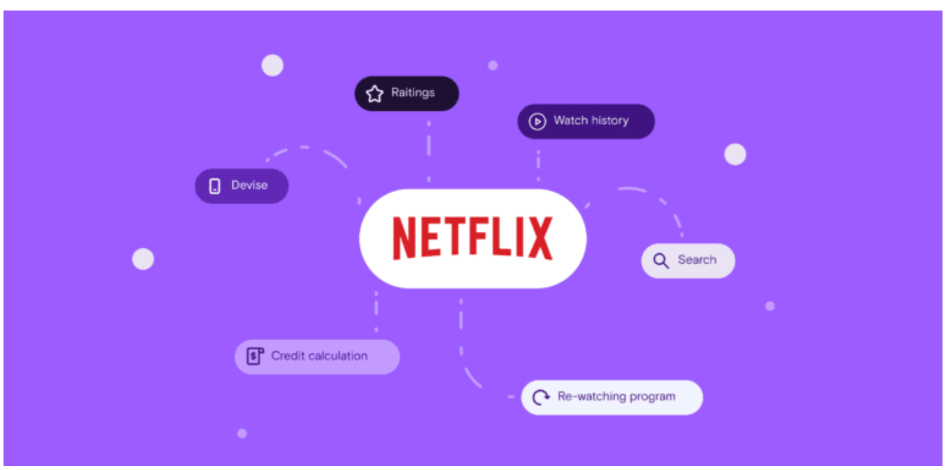
- Spotify's Song based Playlist Recommendation System



- Amazon Product Recommendation Algo



- Netflix Movie/TV Show Recommendation Algo



Dynamic K-Means Clustering Analysis

Three major steps involved in the process are as follows –

- Initialization: Initialize k centroids randomly.
- Update Phase: For each incoming data point, assign it to the nearest centroid(cluster) based on the Euclidean distance.

$$d((x,y),(a,b)) = \sqrt{(x-a)^2 + (y-b)^2}$$

Euclidean distance formula

- Centroid Update: After each data point assignment, update the centroid of the corresponding cluster by considering the new data point. The update can be incremental, reducing the computational cost of recalculating centroids manually.

Q. How is **Machine Learning** being applied in this project?

A. Clutsering is an unsupervised learning model. An Unsupervised learning model is a subset of machine learning in which the algorithm's purpose is to discover structures, relationships, or patterns in a dataset without the presence of labeled datasets or human intervention. Clearly, it means that the algorithm investigates the data's intrinsic structure to group related data points or find underlying patterns.

Topological Data Analysis (TDA)

What is TDA?

Topological Data Analysis (TDA) is a branch of data analysis that aims to extract topological and geometric features from complex data sets. It provides a way to study the shape, structure, and connectivity of data points and uncover hidden patterns and relationships that might not be evident through traditional statistical methods.

Applications of TDA:

- Image and Signal Processing
- Machine Learning and Data Mining
- IoT
- Robotics
- Social Network Analysis
- Natural Language Processing
- And many more….

To perform TDA in the project, following python library has been considered:

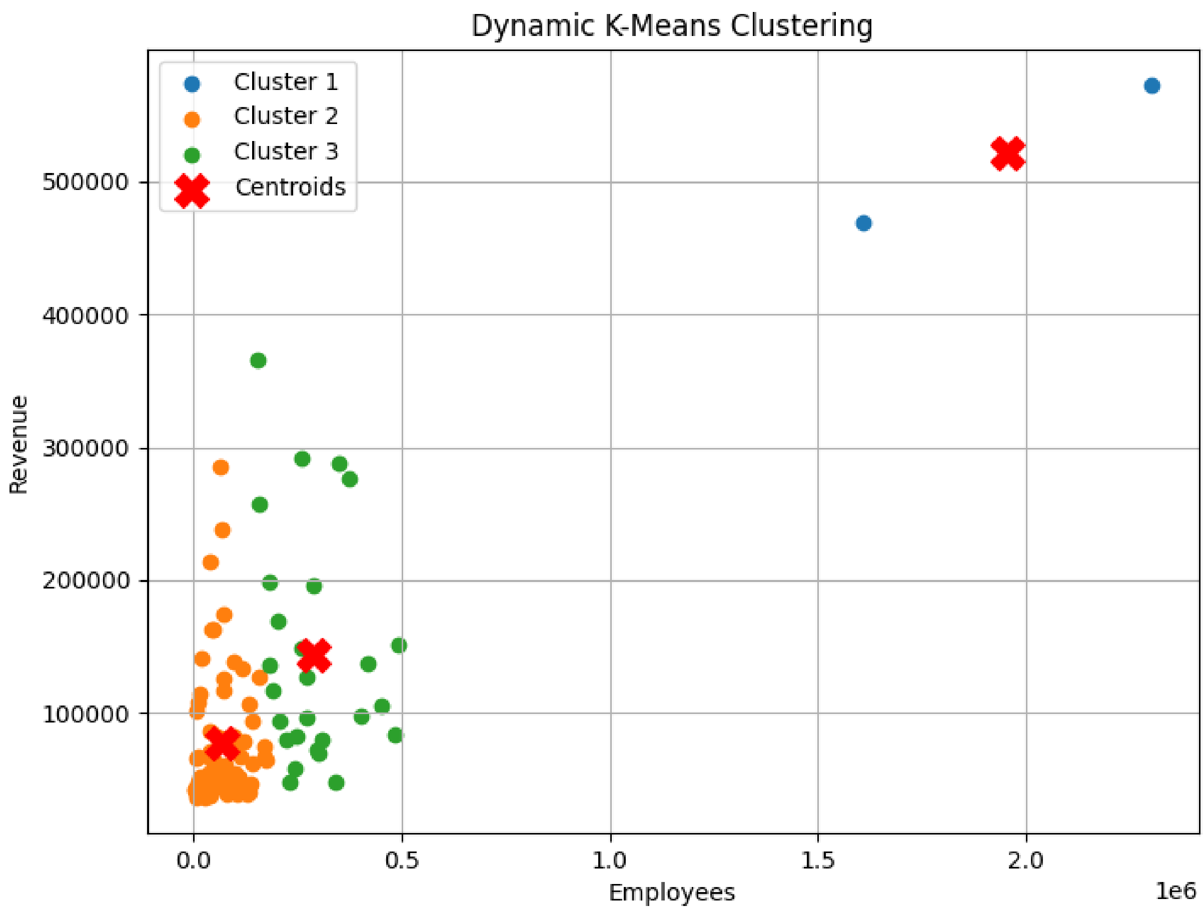
Ripser, which works on the principle of Vietoris Rips Complex that is used to study the relationships between the set of points in a metric space.

CODE v1.0 (Before TDA)

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

def euclidean_distance(point1, point2):                                    #Calculates the Euclidean distance between two data points
    return np.sqrt(np.sum((point1 - point2) ** 2))
```

```python
def assign_to_clusters(data, centroids):                                   #Assigns data points to the nearest cluster based on current centroids
    clusters = [[] for _ in range(len(centroids))]
    for point in data:
        distances = [euclidean_distance(point, centroid) for centroid in centroids]
        cluster_idx = np.argmin(distances)
        clusters[cluster_idx].append(point)
    return clusters

def update_centroids(clusters):                                            #Updates the centroids based on the mean of the data points in each cluster
    return [np.mean(cluster, axis=0) for cluster in clusters]

def dynamic_kmeans(data, k, tolerance=1e-4):                              #Performs dynamic k-means clustering
    np.random.seed(42)                                                     #Randomly initializes centroids
    centroids = np.array([data[i] for i in np.random.choice(len(data), k)])
    while True:
        old_centroids = np.copy(centroids)
        clusters = assign_to_clusters(data, centroids)                    #Assigns data points to clusters using the current centroids
        centroids = update_centroids(clusters)                            #Updates the centroids based on the current cluster assignments
        centroid_shift = np.sum(np.sqrt(np.sum((centroids - old_centroids) ** 2, axis=1)))  #Calculates the shift in centroids to check for convergence
        print(f"Centroid Shift = {centroid_shift}")
        if centroid_shift < tolerance:                                     #Check for convergence
            print("Converged!")
            break
    return centroids, clusters

file_path = "LargestCompaniesInUSAbyReveneue.csv"                         #Reading data from CSV file
data_df = pd.read_csv(file_path)
data_df['Revenue'] = data_df['Revenue'].str.replace(',', '').astype(float)     #Converts comma separated values to float data type
data_df['Employees'] = data_df['Employees'].str.replace(',', '').astype(float)
data = data_df[['Employees','Revenue']].values.tolist()                   #Selecting the 'Revenue' and 'Employees' column for clustering
data_df['Employees'] = pd.to_numeric(data_df['Employees'], errors='coerce')   #Converts non-numeric values to numeric values
data_df['Revenue'] = pd.to_numeric(data_df['Revenue'], errors='coerce')
data_df.dropna(subset=['Employees', 'Revenue'], inplace=True)             #Removes rows with NaN values
k=3                                                                         #Number of clusters
centroids, clusters = dynamic_kmeans(data, k)
plt.figure(figsize=(8, 6))                                                 #Plotting of data and cluster centroids
for i, cluster in enumerate(clusters):                                     #Plotting of data points in each cluster
    cluster_data = np.array(cluster)
    plt.scatter(cluster_data[:, 0], cluster_data[:, 1], label=f"Cluster {i+1}")
centroids_data = np.array(centroids)                                       #Plotting of cluster centroids
plt.scatter(centroids_data[:, 0], centroids_data[:, 1], color='red', marker='X', s=200, label='Centroids')
plt.ylabel('Revenue')
plt.xlabel('Employees')
plt.title('Dynamic K-Means Clustering')
plt.legend()
plt.grid(True)
plt.show()
```

OUTPUT

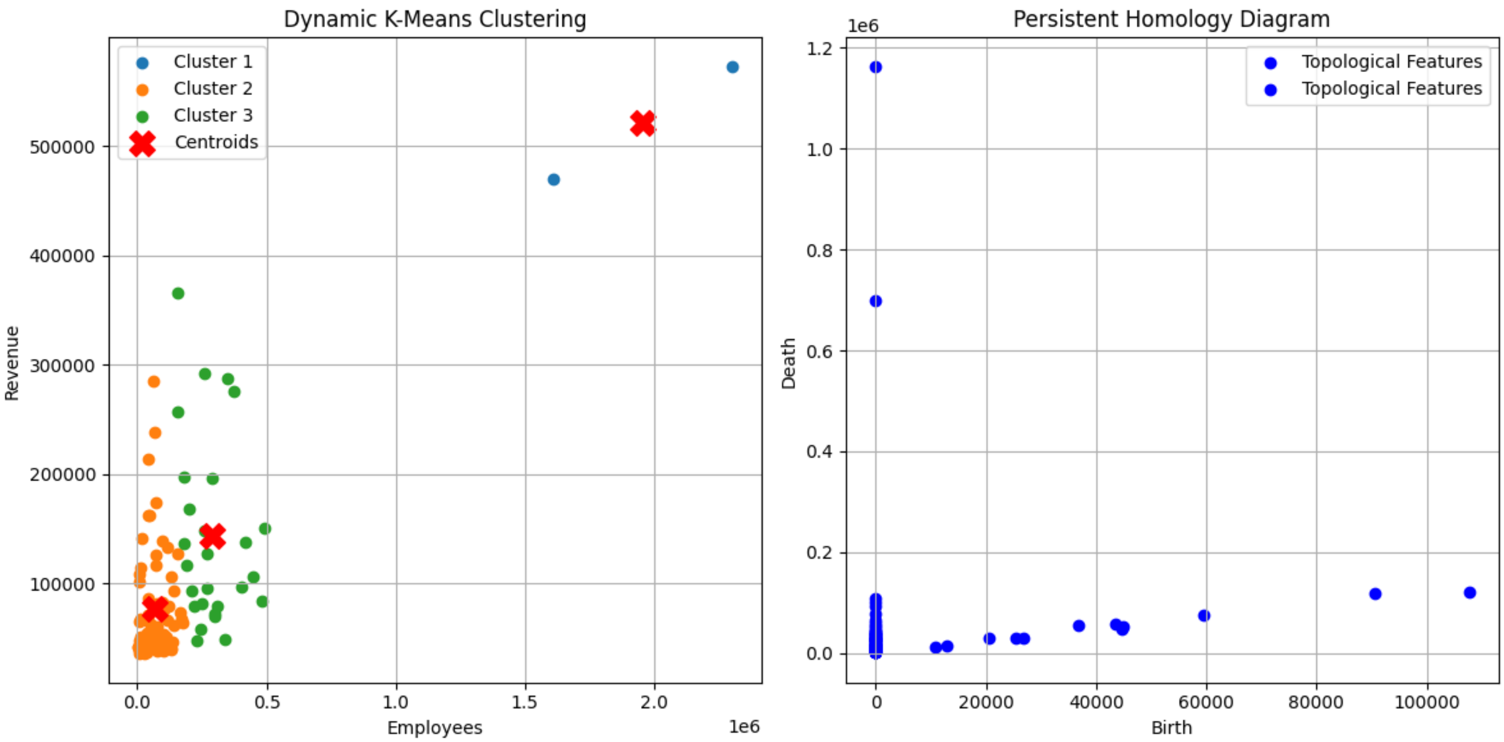


Dynamic K-Means Clustering
Cluster 1
Cluster 2
Cluster 3
Centroids
500000
400000
300000
200000
100000
Revenue
0.0
0.5
1.0
1.5
2.0
Employees
1e6

CODE v2.0 (After TDA)

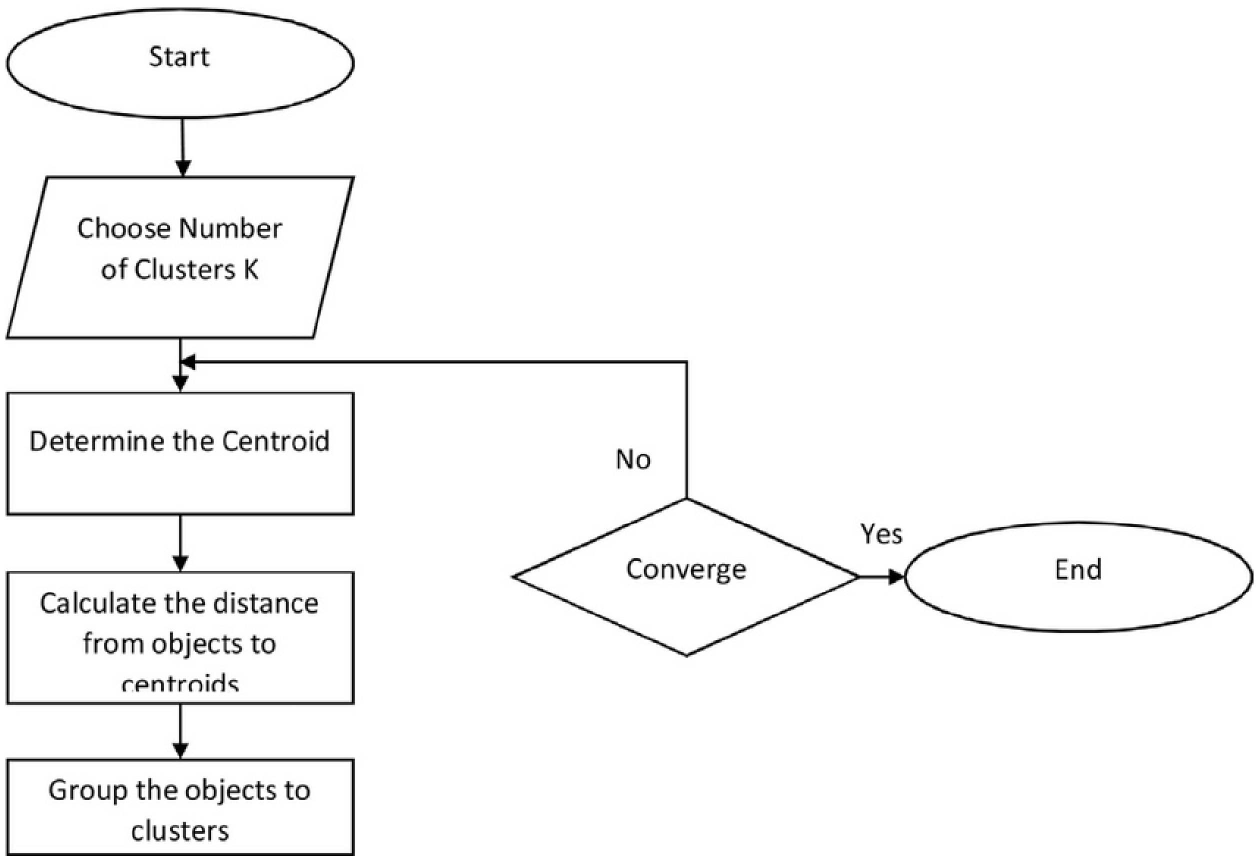
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from ripser import Rips

def euclidean_distance(point1, point2):                    #Calculates the Euclidean distance between two data points
    return np.sqrt(np.sum((point1 - point2) ** 2))

def assign_to_clusters(data, centroids):                   #Assigns data points to the nearest cluster based on current centroids
    clusters = [[] for _ in range(len(centroids))]
    for point in data:
        distances = [euclidean_distance(point, centroid) for centroid in centroids]
        cluster_idx = np.argmin(distances)
        clusters[cluster_idx].append(point)
    return clusters

def update_centroids(clusters):                            #Updates the centroids based on the mean of the data points in each cluster
    return [np.mean(cluster, axis=0) for cluster in clusters]

def dynamic_kmeans(data, k, tolerance=1e-4):               #Performs dynamic k-means clustering
    np.random.seed(42)                                     #Randomly initializes centroids
    centroids = np.array([data[i] for i in np.random.choice(len(data), k)])
    while True:
        old_centroids = np.copy(centroids)
        clusters = assign_to_clusters(data, centroids)     #Assigns data points to clusters using the current centroids
        centroids = update_centroids(clusters)             #Updates the centroids based on the current cluster assignments
        centroid_shift = np.sum(np.sqrt(np.sum((centroids - old_centroids) ** 2, axis=1))) #Calculates the shift in centroids to check for convergence
        print(f"Centroid Shift = {centroid_shift}")
        if centroid_shift < tolerance:                     #Check for convergence
            print("Converged!")
            break
    return centroids, clusters

def topological_data_analysis(data, k, maxdim=1):
    centroids, clusters = dynamic_kmeans(data, k)          #Performs dynamic k-means clustering
    rips = Rips(maxdim=maxdim)                             #Computes persistent homology using ripser
    data_np = np.array(data)
    diagrams = rips.fit_transform(data_np)
    plt.figure(figsize=(12, 6))                            #Visualizes the clustered data and topological features
    plt.subplot(1, 2, 1)
    for i, cluster in enumerate(clusters):                 #Plots data points in each cluster
        cluster_data = np.array(cluster)
        plt.scatter(cluster_data[:, 0], cluster_data[:, 1], label=f"Cluster {i+1}")
    centroids_data = np.array(centroids)                   #Plots cluster centroids
    plt.scatter(centroids_data[:, 0], centroids_data[:, 1], color='red', marker='X', s=200, label='Centroids')
    plt.ylabel('Revenue')
    plt.xlabel('Employees')
    plt.title('Dynamic K-Means Clustering')
    plt.legend()
    plt.grid(True)
    plt.subplot(1, 2, 2)
    for diagram in diagrams:
        if len(diagram) > 0:
            birth, death = diagram[:, :2].T
            plt.plot([0, maxdim], [0, maxdim], 'k--', alpha=0.5)
            plt.scatter(birth, death, c='b', marker='o', label='Topological Features')
            plt.xlabel('Birth')
            plt.ylabel('Death')
            plt.title('Persistent Homology Diagram')
            plt.legend()
            plt.grid(True)
    plt.tight_layout()
    plt.show()

file_path = "LargestCompaniesInUSAbyReveneue.csv"          #Reading data from CSV file
data_df = pd.read_csv(file_path)
data_df['Revenue'] = data_df['Revenue'].str.replace(',', '').astype(float)      #Converts comma separated values to float data type
data_df['Employees'] = data_df['Employees'].str.replace(',', '').astype(float)
data = data_df[['Employees','Revenue']].values.tolist()    #Selects the 'Revenue' and 'Employees' column for clustering
data_df['Employees'] = pd.to_numeric(data_df['Employees'], errors='coerce')     #Converts non-numeric values to numeric values
data_df['Revenue'] = pd.to_numeric(data_df['Revenue'], errors='coerce')
data_df.dropna(subset=['Employees', 'Revenue'], inplace=True)                   #Removes rows with NaN values
k=3                                                        #Number of clusters

topological_data_analysis(data, k)                         #Performs topological data analysis
```

OUTPUT



Graphical view of the dataset using Dynamic K-Means (left) and Persistent Homology portraying TDA

Workflow/State Diagram



Tools Used in the Project

- Visual Studio Code

  Microsoft created Visual Studio Code, sometimes known as VS Code, which is a popular and open-source source-code editor. It is made to help programmers and developers write, modify, and manage code in a variety of programming languages.

- Matplotlib

  Matplotlib is used for data analysts, scientists, engineers, and researchers who need to visualize their data in a useful and educational way. It offers a wide range of tools for producing various types of plots and charts.

- Numpy

  NumPy (Numerical Python) is an open-source tool for data processing, analysis, and scientific research and serves as the foundation for many other scientific computing libraries in Python.

- Pandas

  Pandas is a free and open-source Python toolkit that offers tools and data structures for handling and analyzing structured data effectively. It is based on the NumPy library and is frequently used in a variety of sectors, including data science, finance, economics, and other areas, for data cleaning, transformation, exploration, and analysis.

- Ripser

  Ripser, an open-source python tool uses the idea of persistent homology and computes topological characteristics of data sets. A mathematical method in algebraic topology called persistent homology enables the description of the shape or structure of data at many spatial scales.

CONCLUSION

Therefore, from the above graph it is clear that –

- Cluster 1 having more employees is having a higher revenue as compared to the rest of the companies.
- Cluster 2 having lesser employees is having a lower revenue due to lack of productivity in those companies.
- Similarly, Cluster 3 has lesser employees than Cluster 1 but more than Cluster 2 is having a better revenue growth over time.

Hence, the following operation implies that having more employees leads to a higher revenue over a certain period.

REFERENCES

- https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/
- https://www.kaggle.com/code/ryanholbrook/clustering-with-k-means
- https://developers.google.com/machine-learning/clustering/overview

Dataset of “Largest Companies in USA” is downloaded from –

- https://www.kaggle.com/datasets