LAB 01 – EDA

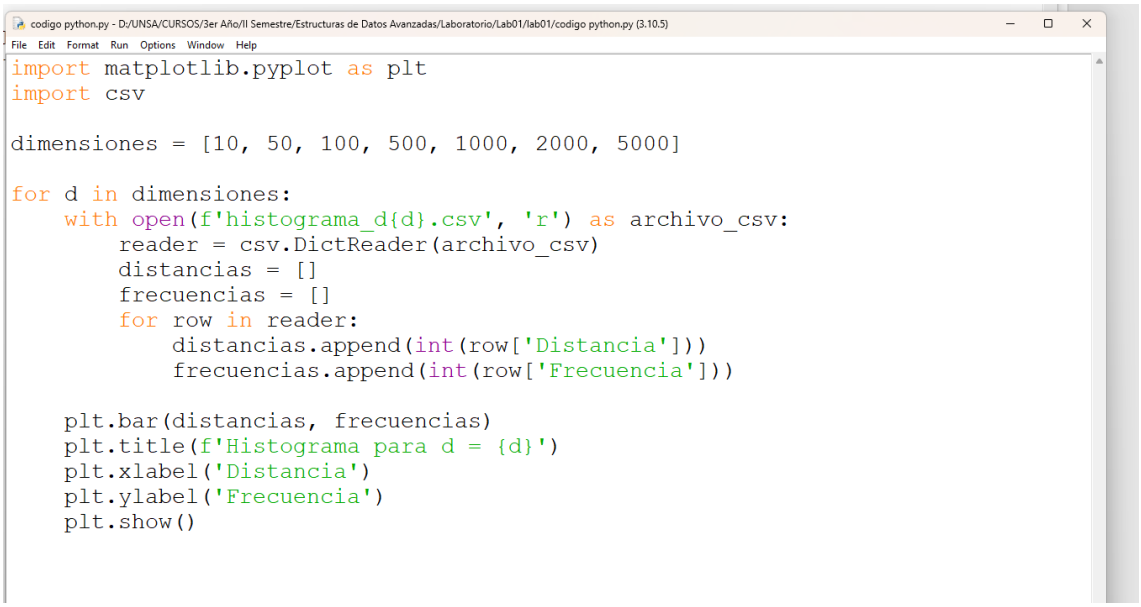
Carazas Quispe, Alessander Jesus

Codigo c++ :

```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <random>
#include <cmath>
#include <map>

// Función para generar puntos aleatorios en una dimensión d
std::vector<std::vector<double>> generarPuntosAleatorios(int d, int numPuntos) {
    std::vector<std::vector<double>> puntos;
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_real_distribution<double> dis(0.0, 1.0);

    for (int i = 0; i < numPuntos; i++) {
        std::vector<double> punto;
        for (int j = 0; j < d; j++) {
            punto.push_back(dis(gen));
        }
        puntos.push_back(punto);
    }

    return puntos;
}

// Función para calcular la distancia euclidiana entre dos puntos
double distanciaEuclidiana(const std::vector<double>& punto1, const std::vector<double>& punto2) {
    double distancia = 0.0;
    for (size_t i = 0; i < punto1.size(); i++) {
        distancia += pow(punto1[i] - punto2[i], 2);
```

```cpp
    }
    return sqrt(distancia);
}

// Función para generar un histograma de distancias y guardarlo en un archivo
void guardarHistograma(int d, const std::vector<std::vector<double>>& puntos) {
    std::map<int, int> histograma;

    for (size_t i = 0; i < puntos.size(); i++) {
        for (size_t j = i + 1; j < puntos.size(); j++) {
            int distancia = static_cast<int>(distanciaEuclidiana(puntos[i], puntos[j]));
            histograma[distancia]++;
        }
    }

    std::ofstream archivo("histograma_d" + std::to_string(d) + ".csv");
    if (!archivo.is_open()) {
        std::cerr << "Error al abrir el archivo.\n";
        return;
    }

    archivo << "Distancia,Frecuencia\n";
    for (const auto& entry : histograma) {
        archivo << entry.first << "," << entry.second << "\n";
    }

    archivo.close();
}

int main() {
    std::vector<int> dimensiones = {10, 50, 100, 500, 1000, 2000, 5000};

    for (int d : dimensiones) {
        int numPuntos = 100;
        std::vector<std::vector<double>> puntos = generarPuntosAleatorios(d, numPuntos);
        guardarHistograma(d, puntos);

        // Imprime el mensaje de finalización para cada dimensión
        std::cout << "Histograma guardado para d = " << d << ".\n";
    }

    return 0;
}
```

Codigo python :



RESULTADO HISTOGRAMAS :

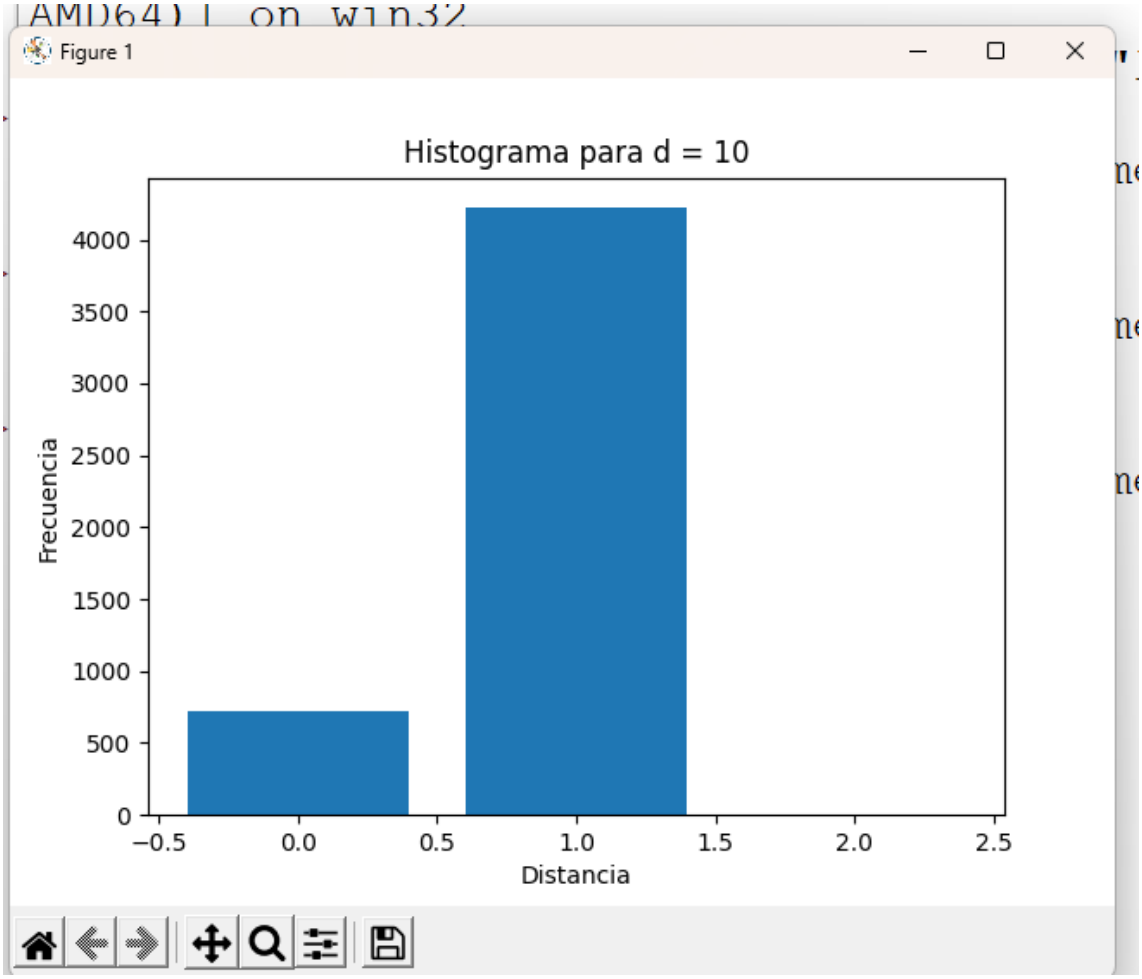



Figure 1
Histograma para d = 50
Frecuencia
3000
2500
2000
1500
1000
500
0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Distancia


Fuente
Párrafo
Figure 1
Histograma para d = 100
Frecuencia
3000
2500
2000
1500
1000
500
0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
Distancia

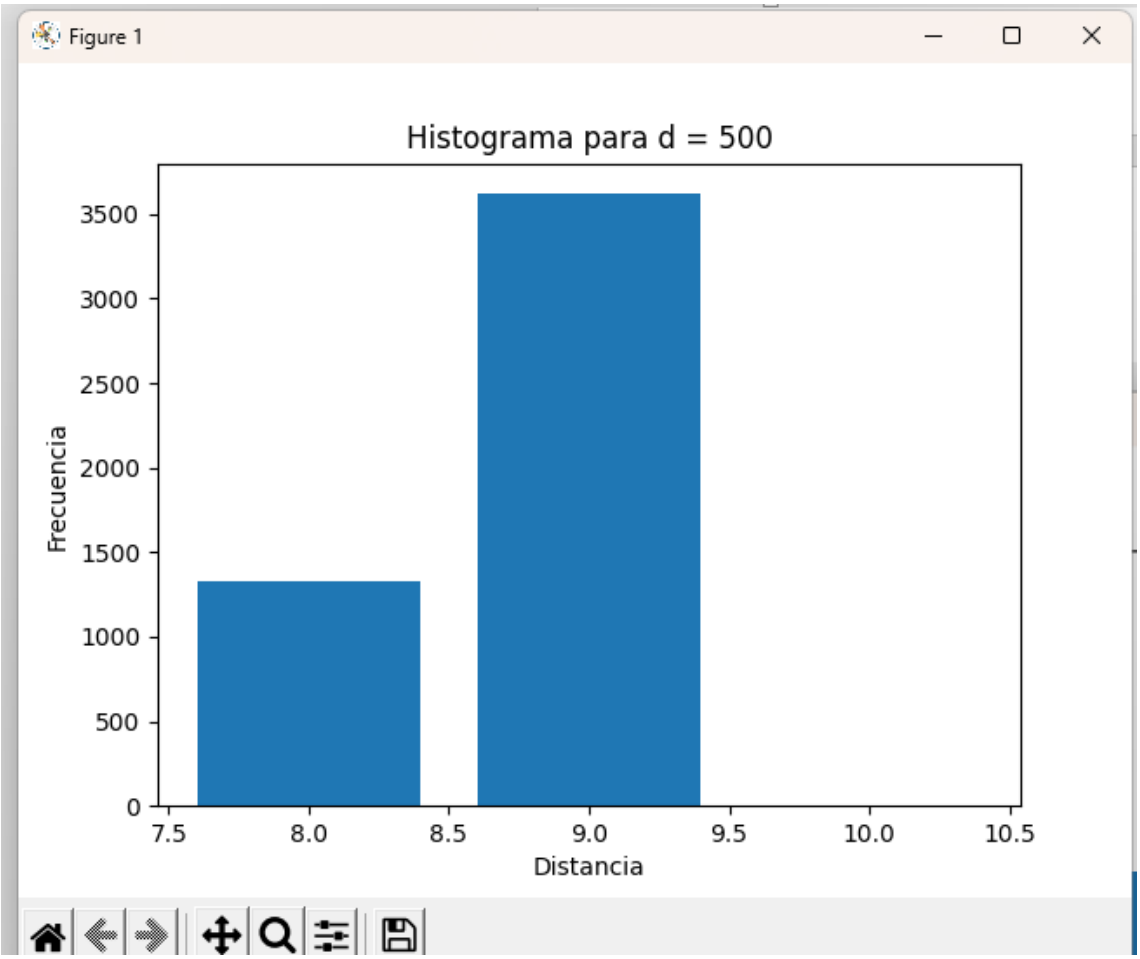


Figure 1
Histograma para d = 500
Frecuencia
Distancia
0
500
1000
1500
2000
2500
3000
3500
7.5
8.0
8.5
9.0
9.5
10.0
10.5

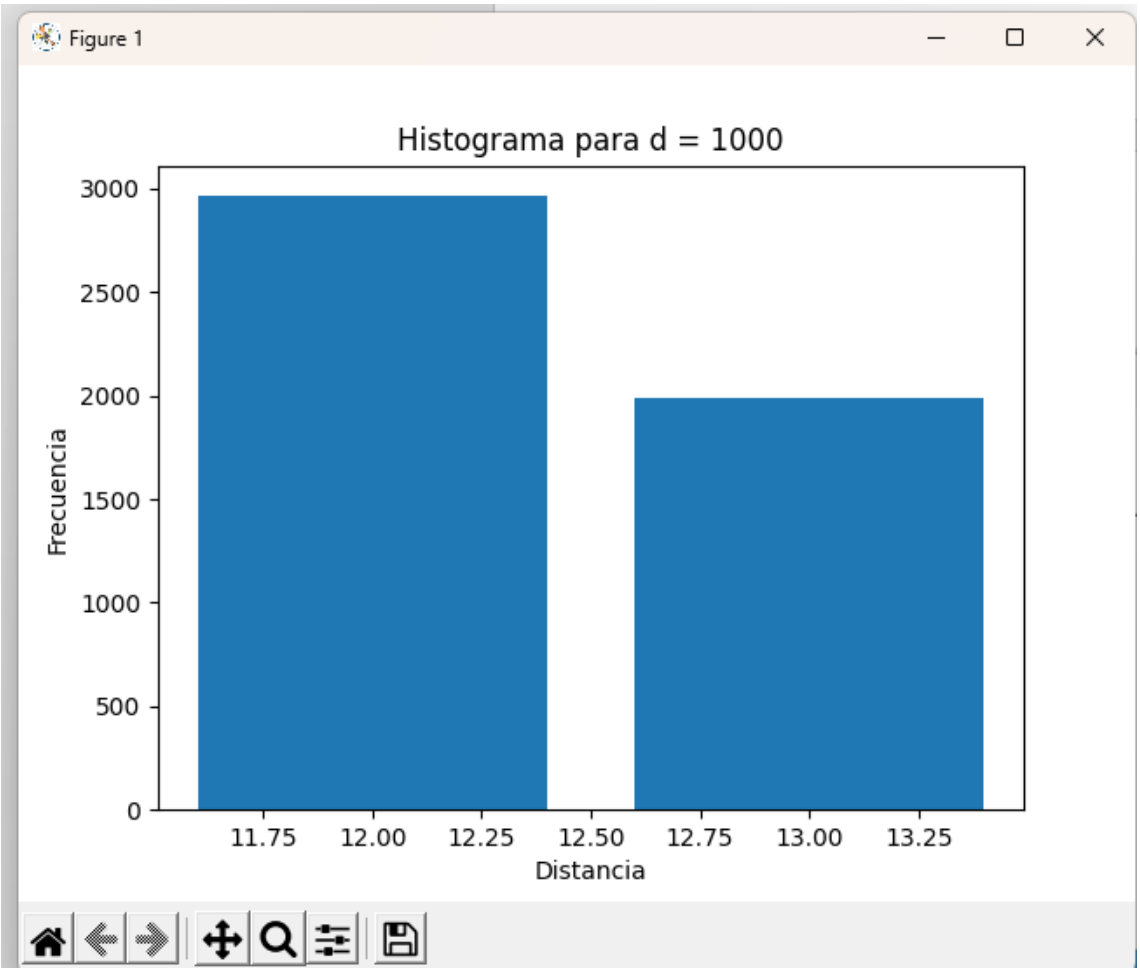


Figure 1
Histograma para d = 1000
3000
2500
2000
1500
1000
500
0
Frecuencia
11.75
12.00
12.25
12.50
12.75
13.00
13.25
Distancia

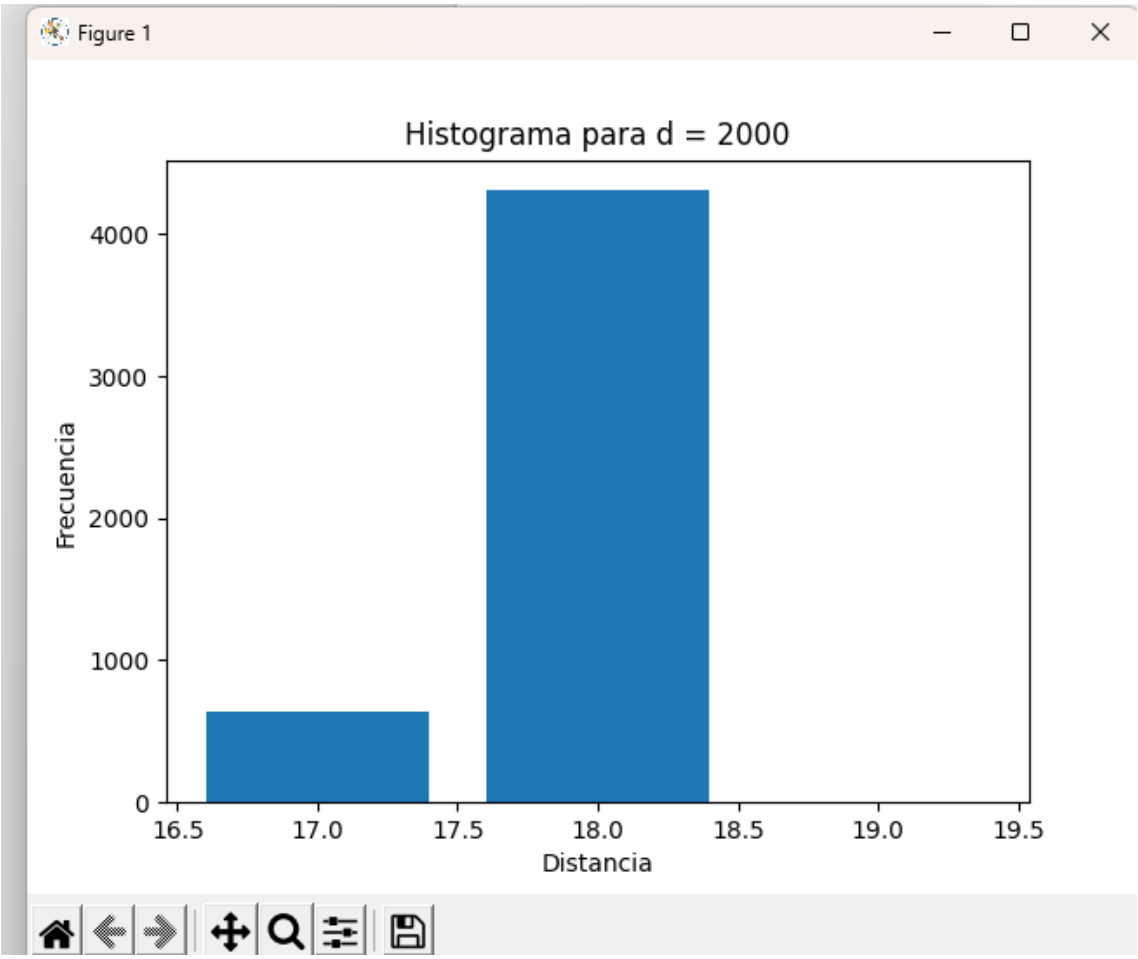
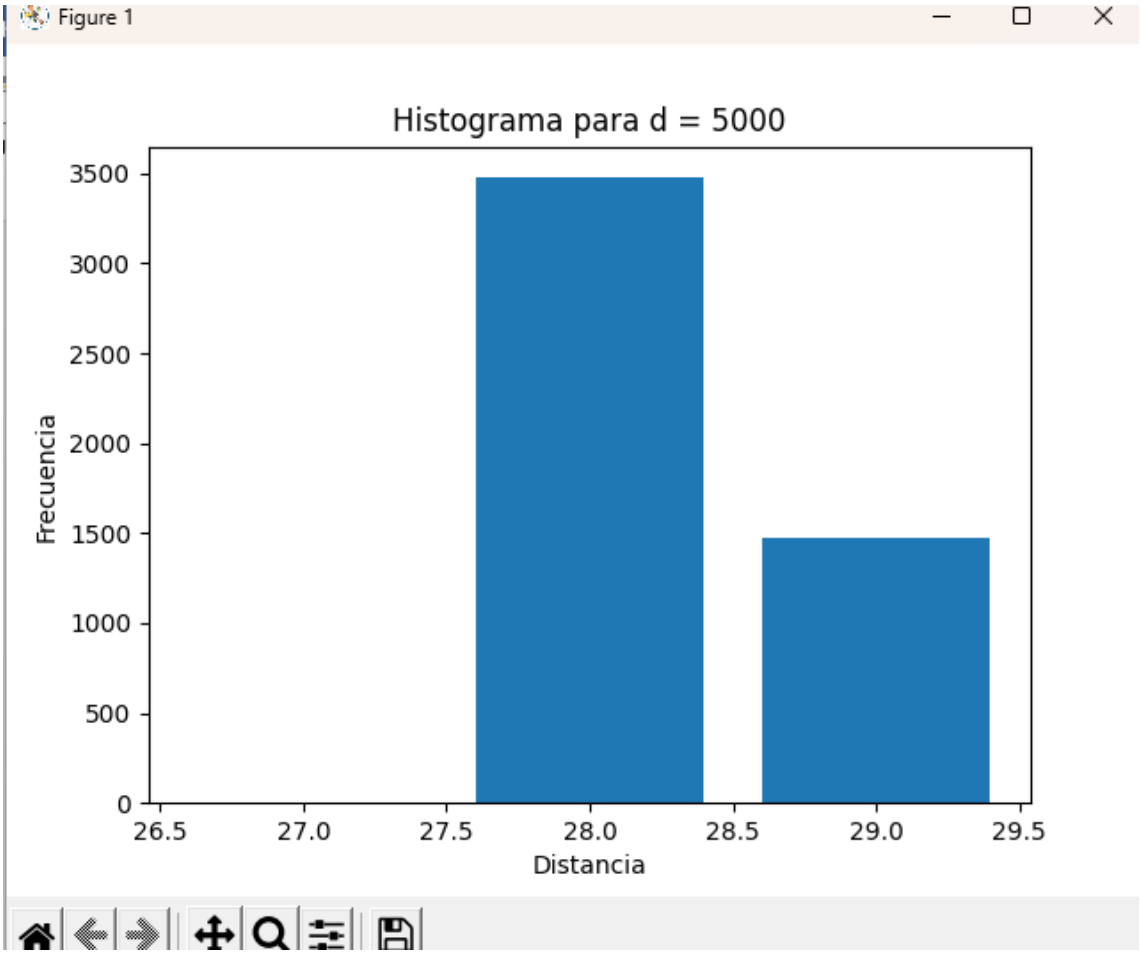


Figure 1
Histograma para d = 2000
Frecuencia
Distancia
0
1000
2000
3000
4000
16.5
17.0
17.5
18.0
18.5
19.0
19.5



Código en C++

El código en C++ se encarga de generar puntos aleatorios en un espacio multidimensional y calcular las distancias euclidianas entre todos los pares de puntos. A continuación, se describe el funcionamiento del código:

Generación de Puntos Aleatorios: La función generarPuntosAleatorios crea un conjunto de puntos aleatorios en una dimensión d, donde d es la dimensión especificada y numPuntos es el número de puntos generados. Para lograrlo, se utiliza un generador de números aleatorios y se distribuyen los valores entre 0 y 1.

Cálculo de Distancias Euclidianas: La función distanciaEuclidiana calcula la distancia euclidiana entre dos puntos dados. Esto se realiza mediante la fórmula de distancia euclidiana estándar.

Generación de Histograma: La función guardarHistograma calcula todas las distancias euclidianas entre pares de puntos y crea un histograma de frecuencias de esas distancias. Luego, guarda este histograma en un archivo CSV, que contiene las distancias y sus frecuencias.

Bucle Principal: El bucle principal del programa itera a través de diferentes dimensiones (10, 50, 100, 500, 1000, 2000, 5000). Para cada dimensión, se generan puntos aleatorios, se calculan las distancias euclidianas y se guarda el histograma en un archivo CSV. Además, se imprime la distancia euclidiana entre todos los pares de puntos para propósitos de análisis.

Código en Python

El código en Python se encarga de graficar los histogramas generados por el código en C++. A continuación, se describe el funcionamiento del código Python:

Importación de Bibliotecas: El código comienza importando la biblioteca matplotlib.pyplot para crear gráficos de barras y la biblioteca csv para leer los archivos CSV generados por el código en C++.

Iteración por Dimensiones: Se itera a través de las mismas dimensiones que se usaron en el código en C++.

Lectura de Datos: El código abre el archivo CSV correspondiente a cada dimensión y lee los datos de distancias y frecuencias.

Graficación de Histograma: Utilizando Matplotlib, se crea un histograma de barras con las distancias en el eje x y las frecuencias en el eje y. Cada histograma representa la distribución de distancias euclidianas para una dimensión dada.

Visualización de Histogramas: Los histogramas se visualizan uno por uno para cada dimensión. Cada gráfico muestra cómo se distribuyen las distancias euclidianas en esa dimensión particular.

HISTOGRAMA:

Valores en el Eje X (Distancias): En el eje x del histograma, podemos observar un rango de valores que representa las distancias euclidianas. Dado que los puntos se generaron en un espacio de 1000 dimensiones y sus coordenadas están distribuidas entre 0 y 1, es probable que las distancias estén en un rango relativamente pequeño, posiblemente entre 0 y algún valor máximo.

Valores en el Eje Y (Frecuencias): En el eje y, la frecuencia indica cuántas veces aparece cada distancia en la muestra. Es probable que algunas distancias sean más comunes que otras. Esto podría relacionarse con la densidad de puntos en el espacio de alta dimensión y cómo se agrupan o se dispersan.

Forma de la Distribución: La forma general de la distribución de distancias podría variar. Podría ser una distribución uniforme si los puntos están uniformemente distribuidos en el espacio, o podría mostrar un pico si los puntos tienden a agruparse en ciertas regiones.

Impacto de la Dimensión: En dimensiones más altas, como 1000, es posible que las distancias euclidianas sean menos informativas debido a la maldición de la dimensionalidad. Las distancias entre puntos tienden a ser más similares en dimensiones altas, lo que podría resultar en un histograma con una distribución más uniforme.

Conclusiones

Este conjunto de programas permite analizar cómo se comportan las distancias euclidianas en espacios multidimensionales. Los histogramas generados y graficados en Python proporcionan una visualización clara de estas distribuciones en función de la dimensión. A través de este análisis, se pueden identificar patrones y tendencias en las distancias euclidianas en diferentes dimensiones, lo que puede tener aplicaciones en diversas áreas, como la estadística, la investigación científica y el aprendizaje automático.

LINK GITHUB:

https://github.com/AlessanderLCarazas/ESTRUCTURAS-DE-DATOS-AVANZADAS