const data = msg.payload;

const k = 2; // number of clusters

const maxIterations = 10;

// Randomly initialize centroids (first k points)

let centroids = data.slice(0, k);

function distance(a, b) {

    return Math.sqrt(

        a.reduce((sum, val, i) => sum + Math.pow(val - b[i], 2), 0)

    );

}

let assignments = [];

for (let i = 0; i < maxIterations; i++) {

    let clusters = Array.from({ length: k }, () => []);

    assignments = [];

    // Assign each point to nearest centroid

    data.forEach(point => {

        let distances = centroids.map(c => distance(point, c));

        let clusterIndex = distances.indexOf(Math.min(...distances));

        clusters[clusterIndex].push(point);

        assignments.push({ point, cluster: clusterIndex });

    });

    // Recalculate centroids

    centroids = clusters.map(cluster => {

        let len = cluster.length;

        if (len === 0) return Array(data[0].length).fill(0); // Prevent NaN

        let sum = cluster.reduce((acc, point) => {

            return acc.map((val, idx) => val + point[idx]);

        }, Array(data[0].length).fill(0));

        return sum.map(val => val / len);

    });

}

// Output includes final centroids and point-cluster mapping

msg.payload = {

    finalCentroids: centroids,

    assignments: assignments

};

return msg;