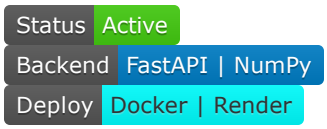
Implementación de K-Means para la Optimización de Ubicación Hospitalaria



KMEANS-hospitales-backend
Repositorio · API K-Means de ubicación hospitalaria

kmeans-hospitales-frontend
Repositorio · UI K-Means de ubicación hospitalaria

MODELO-KMEANS-HOSPITALES.ipynb
Notebook · Implementación Manual de K-Means

Este proyecto es una implementación interactiva del algoritmo de **Machine Learning no supervisado K-Means**, diseñado para resolver un problema de infraestructura urbana: encontrar la ubicación óptima de A hospitales dada una distribución geográfica de N residencias en un plano $M \times M$.

Introducción y Objetivo

El problema de la ubicación de instalaciones busca minimizar el costo de transporte o distancia entre los proveedores de servicios (hospitales) y los consumidores (casas).

En este simulador:

- **El Universo:** Una cuadrícula plana de $M \times M$ coordenadas.
- **Los Datos (Puntos):** N casas distribuidas aleatoriamente.
- **Los Centroides:** A hospitales cuya posición se optimiza iterativamente.

El objetivo del algoritmo es encontrar las coordenadas (x, y) para los hospitales tal que la distancia promedio que cualquier ciudadano debe recorrer para llegar a su hospital más cercano sea mínima.

Marco Teórico

1. Algoritmo K-Means

K-Means es un algoritmo de agrupamiento (clustering) que particiona un conjunto de datos en K grupos predefinidos (en nuestro caso, A hospitales). Funciona iterando dos pasos principales hasta la convergencia:

1. **Asignación (Expectation):** Cada punto de datos se asigna al centroide más cercano.
2. **Actualización (Maximization):** Se recalculan los centroides tomando el promedio de todos los puntos asignados a ese grupo.

2. Distancia Euclidiana

Para calcular la "cercanía", utilizamos la distancia euclidiana, que representa la distancia lineal física en un plano 2D. La fórmula entre una casa $P(x_1, y_1)$ y un hospital $Q(x_2, y_2)$ es:

$$d(P, Q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

3. Métricas de Rendimiento

El sistema calcula en tiempo real dos métricas clave:

- **Inercia (WCSS - Within-Cluster Sum of Squares):** Es la suma de las distancias al cuadrado de cada punto a su centroide asignado. Matemáticamente, buscamos minimizar este valor. Una inercia baja indica agrupamientos compactos.

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2)$$

- **Distancia Promedio:** Una métrica más interpretable para el usuario final. Representa, en promedio, cuántas unidades de distancia debe recorrer una persona para llegar al hospital.

Arquitectura del Sistema

El proyecto sigue una arquitectura desacoplada **Cliente-Servidor** contenerizada con Docker.

Backend (API & Modelo)

- **Lenguaje:** Python 3.10

- **Framework:** FastAPI
- **Cálculo Numérico:** NumPy
- **Persistencia:** Memoria volátil

Frontend (UI & Visualización)

- **Framework:** Next.js 14 (React).
- **Estilos:** Tailwind CSS para un diseño responsivo y moderno.
- **Visualización:** SVG nativo para renderizar la cuadrícula, casas y hospitales con alto rendimiento.
- **Estado:** React Hooks (`useState` , `useEffect`) manejando una máquina de estados finita.

Lógica del Negocio

La aplicación implementa una **Máquina de Estados Estricta** para guiar al usuario a través del proceso algorítmico correcto. No se permite saltar pasos ilógicos.

Paso (Step)	Estado del Sistema	Acción Permitida
0	Inicio	Configurar tamaño del Grid (M).
1	Grid Configurado	Generar población aleatoria (N).
2	Población Lista	Definir e inicializar Hospitales (A).
3	Listo para Iterar	**Asignar Clusters:** Calcular distancias y asignar casas.
4	Clusterizado	**Actualizar Centroides:** Mover hospitales a la media aritmética.
5	Optimizado	Volver al Paso 3 (Iterar nuevamente) hasta convergencia.

Despliegue en Producción

La aplicación está desplegada en **Render** utilizando contenedores Docker orquestados.

Estrategia de Conexión

Dado que el Frontend y el Backend están en repositorios y servicios separados:

1. **Backend:** Se despliega primero. Render le asigna una URL pública (ej. `https://kmeans-hospitales-backend.onrender.com`).
2. **Frontend:** Se despliega después. Durante la construcción (build), se le inyecta la URL del backend mediante una variable de entorno.

Configuración de Variables (Environment Variables)

En el servicio del Frontend en Render, se configuró:

- `NEXT_PUBLIC_API_URL`： `https://kmeans-hospitales-backend.onrender.com`

Esto permite que el cliente React sepa exactamente a dónde enviar las peticiones HTTP para ejecutar los cálculos matemáticos.