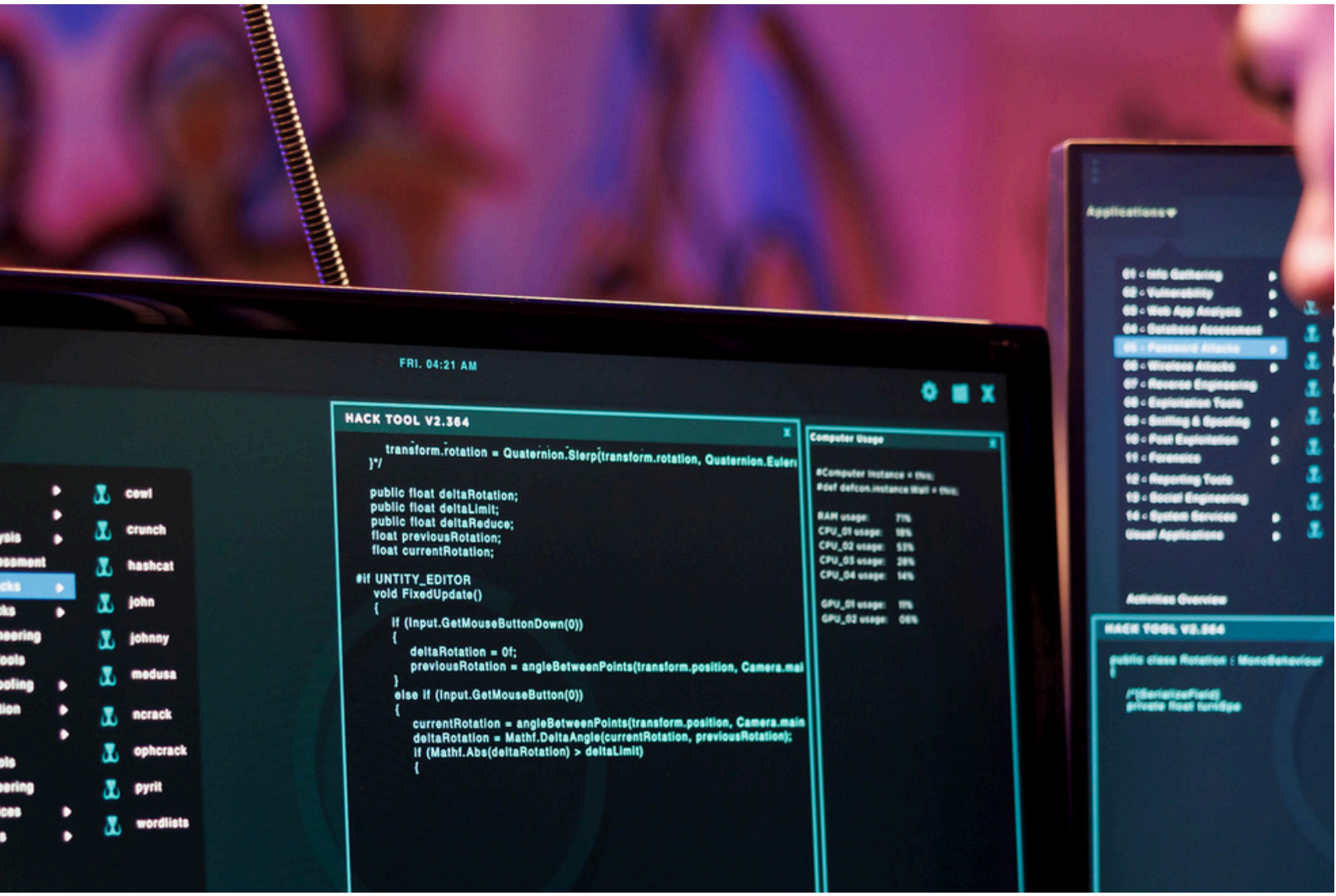
TIC



Introducción a los algoritmos de agrupamiento

TIC

Introducción a los algoritmos de agrupamiento

El agrupamiento es una técnica de aprendizaje no supervisado que consiste en dividir un conjunto de datos en grupos o clusters, donde los elementos dentro de un mismo grupo son más similares entre sí que con los elementos de otros grupos. Esta técnica es fundamental en el aprendizaje no supervisado porque permite descubrir patrones y estructuras ocultas en los datos sin la necesidad de etiquetas previas. Por ejemplo:

En un conjunto de datos de clientes de un supermercado, el agrupamiento podría ayudar a identificar diferentes segmentos de clientes con características similares de compra.

Conceptos Básicos como Centroides, Distancia Euclidiana y Función de Similitud

Centroides

En el contexto del agrupamiento, un centroide es el punto representativo de un cluster, calculado como el promedio de todos los puntos en el cluster. Los centroides son utilizados por algoritmos como K-Means para definir la ubicación de los clusters.

Distancia Euclidiana

Es una medida de la distancia entre dos puntos en un espacio euclidiano. Se calcula como la longitud del segmento de línea recta que une los dos puntos. La distancia euclidiana es comúnmente utilizada en algoritmos de agrupamiento para medir la similitud entre puntos.

TIC

Función de Similitud	Es una función que cuantifica la similitud entre dos elementos en un conjunto de datos. Puede ser una medida de distancia, como la distancia euclidiana, o una medida de similitud, como la correlación de Pearson. La función de similitud es esencial para determinar qué tan cerca están dos puntos y, por lo tanto, para agruparlos adecuadamente

Aplicaciones de los Algoritmos de Agrupamiento en Diferentes Campos

Los algoritmos de agrupamiento tienen una amplia gama de aplicaciones en diversos campos, incluyendo:

- **Marketing:** Segmentación de clientes para campañas de marketing personalizadas.
- **Medicina:** Agrupamiento de pacientes basado en datos de historias clínicas para la detección temprana de enfermedades.
- Análisis de Imágenes: Agrupamiento de píxeles para la segmentación de imágenes médicas o la identificación de objetos en fotografías.
- **Análisis de Redes Sociales:** Identificación de comunidades o grupos de usuarios con intereses similares en redes sociales.
- **Procesamiento de Lenguaje Natural:** Agrupamiento de documentos de texto para la organización y clasificación de información.

Generación de Datos Aleatorios

```
import numpy as np

# Generar datos aleatorios con dos características para la
demostración

np.random.seed(0)

X = np.random.rand(100, 2)


print("Datos generados aleatoriamente:")

print(X[:5])
```

Cálculo de la Distancia Euclidiana

```
from scipy.spatial.distance import euclidean

# Calcular la distancia euclidiana entre dos puntos

point1 = [1, 2]

point2 = [4, 6]

distance = euclidean(point1, point2)

print("Distancia Euclidiana entre los puntos:", distance)
```

Implementación de K-Means

```
from sklearn.cluster import KMeans

import matplotlib.pyplot as plt


# Generar datos aleatorios para clustering

X = np.random.rand(100, 2)
```

```
# Inicializar y ajustar el modelo K-Means
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)

# Visualizar los clusters
plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_, cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,
1], marker='x', color='red', s=200)
plt.title("Clustering con K-Means")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

Aplicación de Clustering en un Conjunto de Datos Real

```
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Cargar conjunto de datos Iris
iris = load_iris()
X = iris.data
y = iris.target

# Normalizar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# Reducir dimensionalidad con PCA para visualización
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Inicializar y ajustar el modelo K-Means
kmeans = KMeans(n_clusters=3)
kmeans.fit(X_scaled)

# Visualizar los clusters
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=kmeans.labels_,
cmap='viridis')
plt.title("Clustering con K-Means en Conjunto de Datos Iris")
plt.xlabel("Componente Principal 1")
plt.ylabel("Componente Principal 2")
plt.show()
```