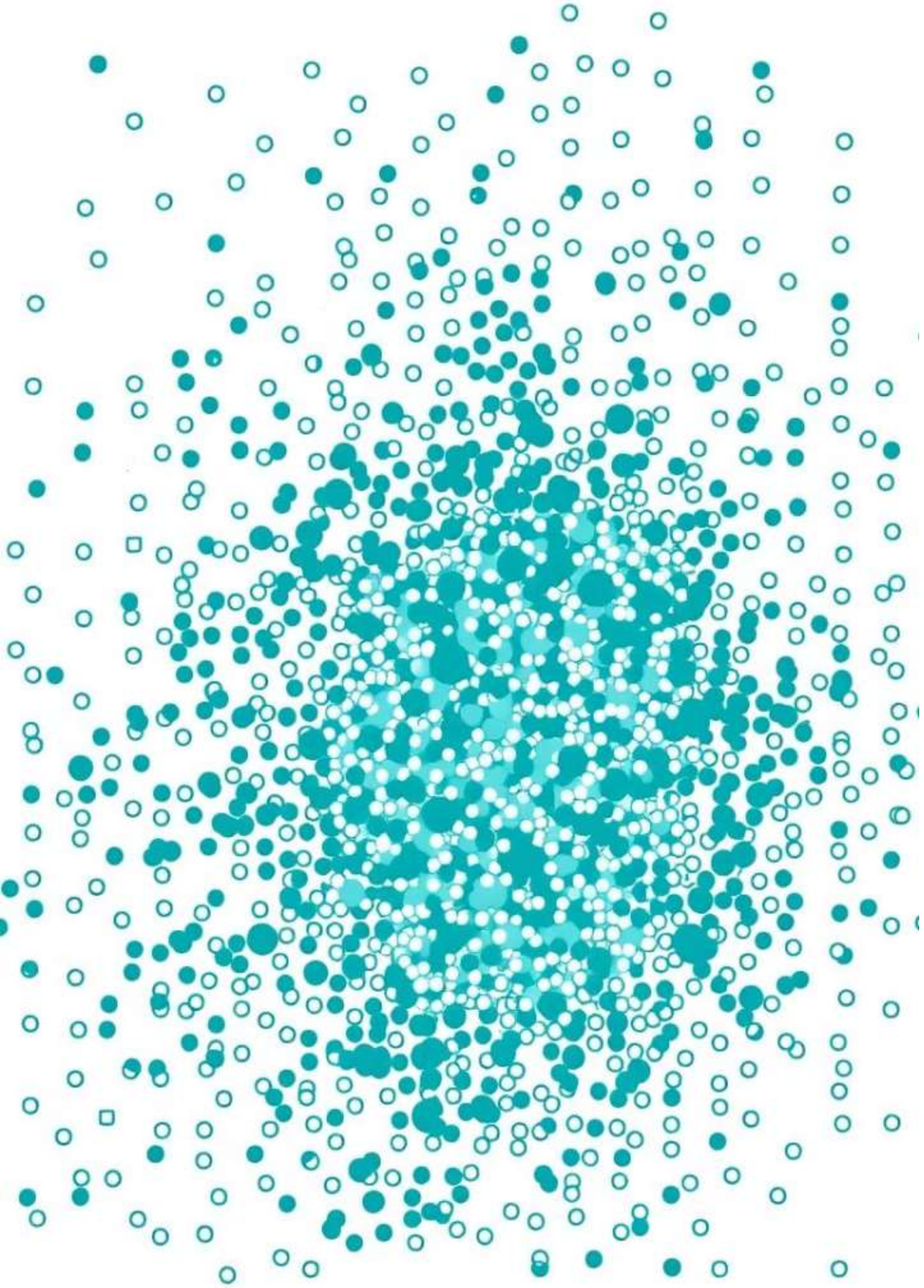
El Método K-Nearest Neighbors (KNN)

El método K-Nearest Neighbors (KNN) es un algoritmo simple y eficaz de aprendizaje supervisado utilizado para clasificación y regresión. Se basa en la premisa de que los elementos similares se encuentran cerca en el espacio de características.



Cómo Funciona KNN

1

1 Definir K

El parámetro K define cuántos vecinos más cercanos se utilizarán para clasificar un nuevo punto de datos.

2

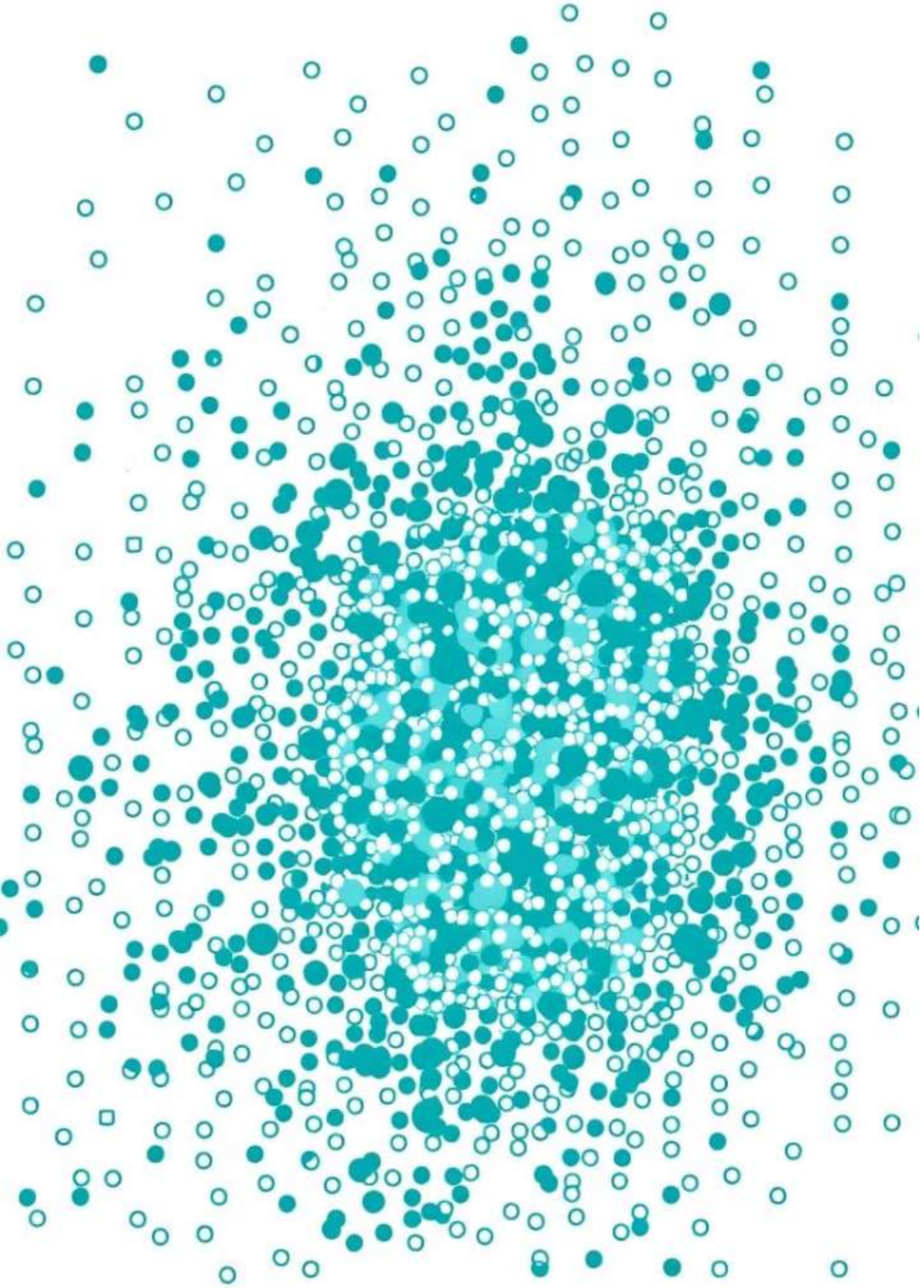
2. Calcular Distancias

Se calcula la distancia entre el nuevo punto y todos los puntos de datos existentes.

3

3. Seleccionar Vecinos

Se seleccionan los K puntos más cercanos al nuevo punto.



Cómo Funciona KNN

1. **Entrenamiento**:
KNN no tiene una fase de entrenamiento tradicional como otros algoritmos. Simplemente almacena los datos de entrenamiento.

2. **Predicción**:
Cuando se presenta un nuevo punto, el algoritmo calcula la distancia entre este y todos los puntos en el conjunto de entrenamiento.
Selecciona los **K vecinos más cercanos** (por lo general usando una métrica como la distancia euclidiana).
Clasificación: Asigna al punto la clase más común entre sus K vecinos.
Regresión: Asigna al punto un promedio (o una función agregada) de los valores objetivo de sus K vecinos.

3. **Parámetros importantes**:
K: Número de vecinos a considerar. Valores pequeños pueden llevar a un modelo ruidoso (*overfitting*), y valores grandes pueden suavizar demasiado (*underfitting*).
Métrica de distancia: La distancia euclidiana es común, pero se pueden usar otras métricas (Manhattan, Minkowski, etc.).

Métricas de Distancia

Distancia Euclidiana

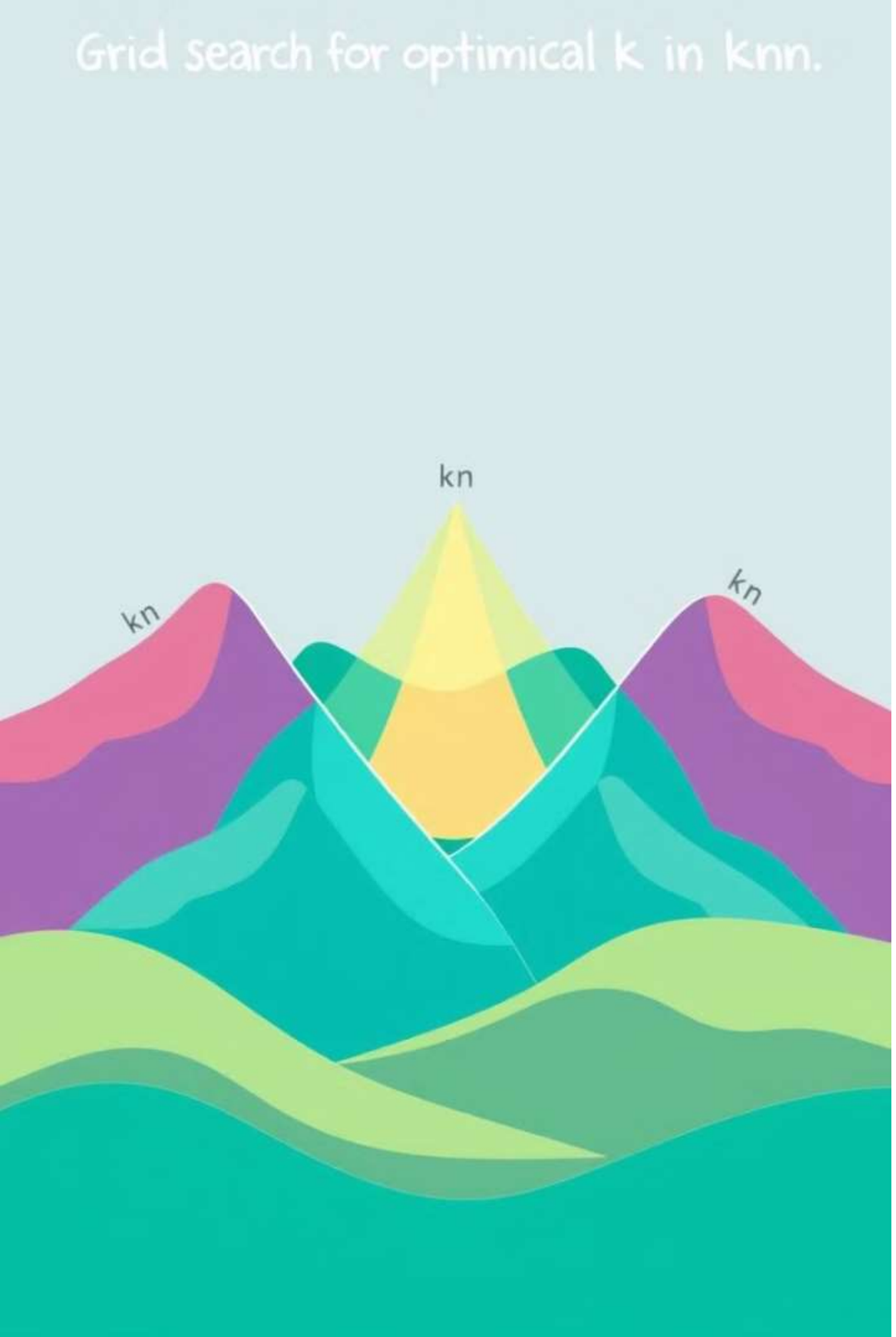
La métrica más común, que mide la distancia recta entre dos puntos.

Distancia Manhattan

Suma las diferencias absolutas entre las coordenadas de los puntos.

Distancia Coseno

Mide el coseno del ángulo entre dos vectores.



Seleccionando el Valor de K

Regla general

Un valor impar de K evita empates y es más robusto al ruido.

Validación Cruzada

Probar diferentes valores de K y elegir el que mejores resultados dé.

Complejidad

Un valor de K mayor aumenta la complejidad computacional.

Sobreajuste

Un valor de K muy pequeño puede provocar sobreajuste a los datos de entrenamiento.

Ventajas y Desventajas de KNN

1 **Ventajas**

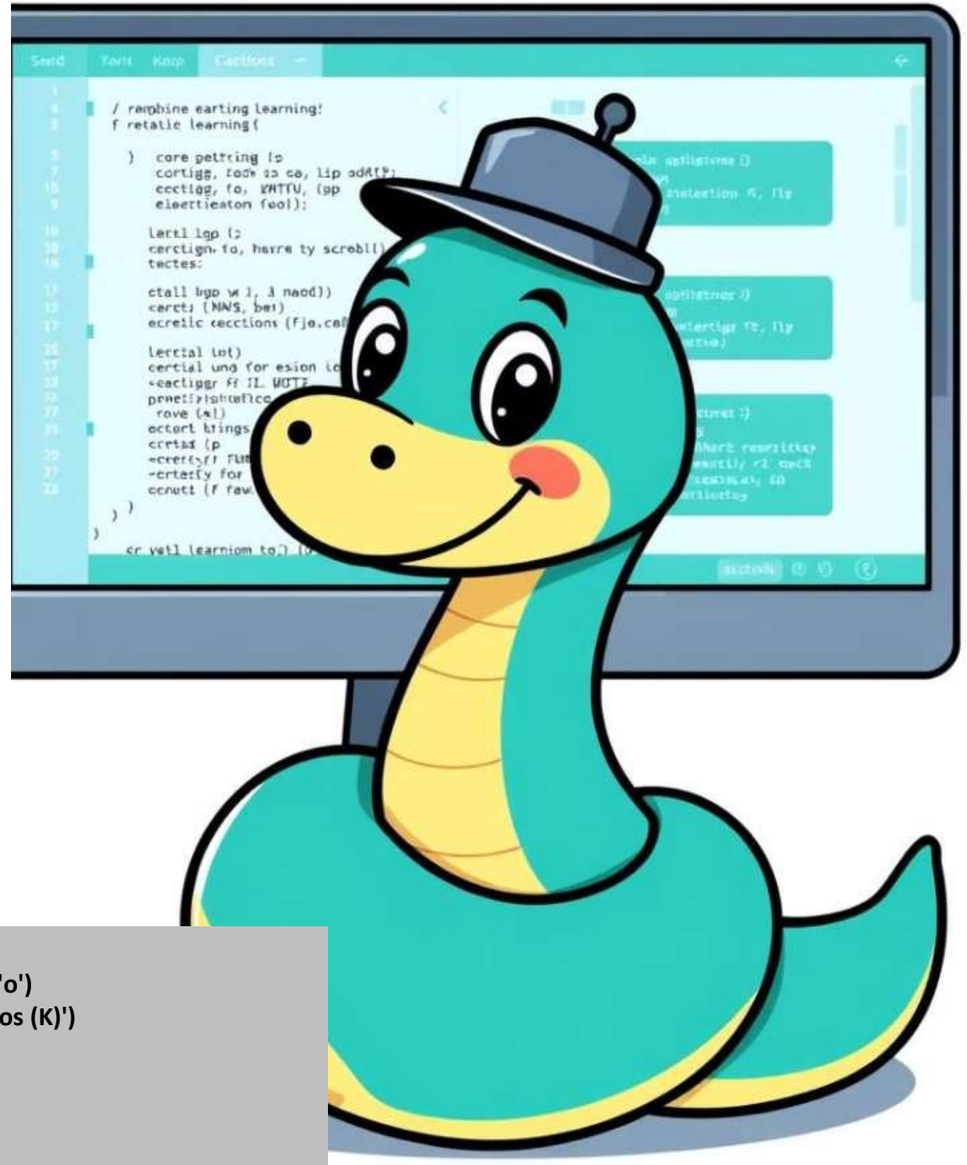
Fácil de implementar, no requiere entrenamiento, maneja datos no lineales.
No requiere suposiciones sobre la distribución de los datos.
Se adapta bien a problemas multiclase.

2 **Desventajas**

Sensible a la escala de las características, requiere mucha memoria, lento en tiempo de ejecución.

Ejemplo Práctico en Python

```
# Importar las librerías necesarias
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score
import matplotlib.pyplot as plt

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

# Dividir en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Escalar las características (importante para KNN)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Crear el modelo KNN con K=3
knn = KNeighborsClassifier(n_neighbors=3)

# Entrenar el modelo
knn.fit(X_train, y_train)

# Predecir las etiquetas del conjunto de prueba
y_pred = knn.predict(X_test)

# Evaluar el modelo
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nReporte de clasificación:\n", classification_report(y_test, y_pred))

# Visualización de cómo cambia el accuracy para diferentes valores de K
k_values = range(1, 20)
accuracies = []

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    accuracies.append(knn.score(X_test, y_test))
```

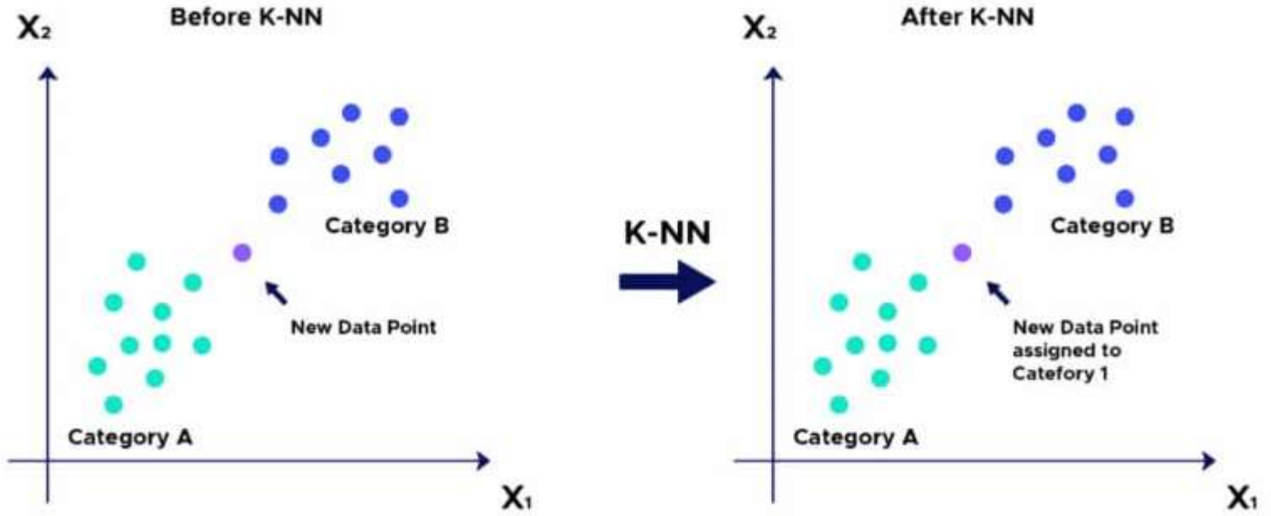


```
plt.plot(k_values, accuracies, marker='o')
plt.title('Accuracy vs Número de Vecinos (K)')
plt.xlabel('Número de Vecinos (K)')
plt.ylabel('Accuracy')
plt.show()
```

Optimización de parámetros

Puedes usar herramientas como **GridSearchCV** para buscar automáticamente los mejores valores de parámetros para tu problema específico:

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_neighbors': [3, 5, 7, 9],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan']
}

grid_search = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5)
grid_search.fit(X_train, y_train)

print("Mejores parámetros:", grid_search.best_params_)
```



Conclusiones

Simplicidad	KNN es un método simple y fácil de implementar.
Flexibilidad	Puede manejar problemas de clasificación y regresión.
Sensibilidad	El rendimiento depende de la elección adecuada de K y las métricas de distancia.
Escalabilidad	Puede ser computacionalmente costoso para conjuntos de datos grandes.