Bénemerita Universidad Autónoma de Puebla

Facultad DE Ciencias de la Computación

Ingenieria en Tecnologías de la Información

Materia

Mineria de Datos

Reporte
Práctica 1-K

Profesor:

Pedro T. Vera

Integrantes:

Blanca Flor Visca Cocotzin
Christopher Aguilar Mendez
Angel Fernando Ruiz Vasquez

03 de febrero del 2026

Índice

Mineria de Datos
Práctica 1-KN

Blanca Flor Visca Cocotzin, Christopher Aguilar Mendez, Angel Fernando Ruiz Vasquez

03 de febrero, 2026

Resumen

En este reporte se presenta la implementación del algoritmo de clasificación supervisada K-Nearest Neighbors (K-NN) utilizando la base de datos Iris del repositorio UCI. El modelo se desarrolló empleando la distancia euclidiana como medida de similitud y se evaluó mediante la métrica de exactitud (accuracy) para los valores de K=3, K=5 y K=7, con el fin de seleccionar el parámetro que ofrece mejor desempeño. Posteriormente, el clasificador se aplicó a un conjunto de datos desconocidos para asignar su clase correspondiente. Finalmente, se integró una interfaz gráfica para facilitar la carga de archivos y la ejecución del programa.

1. Introducción

La clasificación es una tarea fundamental en minería de datos, ya que permite asignar una categoría a nuevos objetos a partir de ejemplos conocidos. En esta práctica se implementó el algoritmo K-Nearest Neighbors (K-NN) utilizando la base de datos Iris, aplicando distancia euclidiana para medir similitud entre registros. Además, se evaluó el rendimiento del modelo para distintos valores de K con el fin de seleccionar el más adecuado.

2. Descripción del algoritmo K-NN

El algoritmo *K-Nearest Neighbors* (K-NN) es un método de clasificación supervisada que asigna una clase a un objeto desconocido a partir de los K ejemplos más cercanos del conjunto de entrenamiento. Para ello se calcula la distancia (en esta práctica, distancia euclidiana), se seleccionan los K vecinos con menor distancia y se asigna la clase por votación mayoritaria.

2.1. Fórmulas utilizadas

1) Distancia euclidiana: para medir la cercanía entre dos objetos p y q con n atributos, se utilizó:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{1}$$

En el conjunto Iris se tienen $n = 4$ atributos por objeto.

2) Votación mayoritaria: una vez obtenidos los K vecinos más cercanos, la clase asignada se definió como la clase con mayor frecuencia entre dichos vecinos:

$$\hat{y} = \arg\max_{c} \quad \text{count}(c) \tag{2}$$

donde count(c) representa el número de vecinos cuya clase es c.

3) Exactitud del modelo (Accuracy): para evaluar el desempeño del clasificador, se utilizó la métrica de exactitud:

$$\text{Exactitud} = \frac{A}{B} \tag{3}$$

donde:

- A = número de casos correctamente clasificados del conjunto de prueba.
- B = total de casos del conjunto de prueba.

3. Metodología

Para resolver el problema se siguió una metodología por etapas:

1. **Lectura de datos:** se cargaron los archivos `DataTrained-iris.data`, `TestData-iris.data` y `NewData-iris.data`.
2. **Preparación de datos:** se separaron los atributos numéricos (4 características) y las etiquetas de clase cuando estaban disponibles.

3. **Cálculo de distancias:** se implementó la distancia euclidiana para medir la cercanía entre objetos.

4. **Clasificación K-NN:** para cada objeto se calcularon distancias respecto al conjunto de entrenamiento, se ordenaron y se eligieron los K vecinos más cercanos.

5. **Evaluación:** se calculó la exactitud del modelo usando el conjunto de prueba.

6. **Selección del mejor K:** se compararon los resultados para $K = 3, 5, 7$ y se eligió el que obtuvo mayor exactitud.

7. **Clasificación de nuevos datos:** se asignó clase a los objetos del archivo `NewData-iris.data` con el mejor K.

8. **Interfaz gráfica (GUI):** se implementó una GUI para cargar archivos, ejecutar el modelo y generar automáticamente los archivos de salida.

4. Entrenamiento del modelo

Para el entrenamiento del clasificador se utilizó el conjunto `DataTrained-iris.data`, el cual contiene objetos previamente etiquetados con su clase correspondiente. A diferencia de otros algoritmos de aprendizaje supervisado, el método K-NN no genera un modelo matemático explícito durante la fase de entrenamiento, sino que almacena los datos de entrenamiento y realiza el cálculo de distancias en el momento de la clasificación.

Con el objetivo de analizar el comportamiento del algoritmo ante distintos parámetros, se realizaron pruebas con $K = 3$, $K = 5$ y $K = 7$. Para cada valor de K, el clasificador fue evaluado utilizando el conjunto `TestData-iris.data`, comparando la clase predicha con la clase real y calculando la métrica de exactitud.

Los resultados obtenidos se resumen en la Tabla 1. A partir de esta comparación se seleccionó como mejor valor de K aquel que presentó la mayor exactitud, ya que representa la configuración más adecuada para el problema de clasificación planteado.

Valor de K	**Exactitud**
3	96.67 %
5	96.67 %
7	96.67 %

Tabla 1: Comparación de exactitud del modelo para distintos valores de K.

De acuerdo con la Tabla 1, los valores de $K = 3$, $K = 5$ y $K = 7$ alcanzaron la misma exactitud, correspondiente a 96.67 %. Este resultado indica que, para el conjunto de datos Iris utilizado

en esta práctica, el algoritmo K-NN mantiene un desempeño estable ante variaciones moderadas del parámetro K, sin verse afectado significativamente por la inclusión de un mayor número de vecinos.

Aunque los tres valores produjeron la misma exactitud, se seleccionó $K = 3$ **como el mejor modelo**, ya que representa la configuración más simple que logra el máximo desempeño, reduciendo el costo computacional y minimizando la influencia de vecinos más lejanos que podrían introducir ambigüedad en la clasificación.

5. Clasificación de datos desconocidos

Una vez determinado el mejor valor de K, se procedió a clasificar los registros del archivo `NewData-iris.data`, el cual contiene objetos sin etiqueta de clase. Para cada objeto se calcularon las distancias euclidianas respecto al conjunto de entrenamiento, se seleccionaron los K vecinos más cercanos y se asignó la clase mediante el criterio de votación mayoritaria.

Como resultado, se obtuvo la clase predicha para cada uno de los objetos desconocidos. Dichos resultados fueron almacenados en un archivo de salida en formato Excel, donde se incluyen los atributos de cada objeto y la clase asignada por el modelo. Esta etapa permite comprobar la utilidad del algoritmo K-NN para predecir la clase de nuevos casos a partir de ejemplos previamente conocidos.

6. Uso de WEKA como herramienta

Para complementar la práctica, se utilizó la herramienta de minería de datos WEKA con el objetivo de aplicar el algoritmo K-NN sobre el mismo conjunto de datos Iris y comparar los resultados con la implementación desarrollada en Python.

El procedimiento consistió en cargar el conjunto de entrenamiento en WEKA, configurar el clasificador `IBk`, establecer los valores de K propuestos (3, 5 y 7) y ejecutar la evaluación utilizando un conjunto de prueba proporcionado por el usuario. Posteriormente, se registraron las métricas generadas por la herramienta, principalmente la exactitud del modelo.

El uso de una herramienta especializada permite validar el correcto funcionamiento del algoritmo implementado, además de facilitar el análisis mediante reportes automáticos, matrices de confusión y métricas adicionales de desempeño.

7. Comparación de resultados: Python vs herramienta

Finalmente, se compararon los resultados obtenidos por el programa implementado en Python con los resultados generados por la herramienta de minería de datos WEKA. La comparación se centró

principalmente en la exactitud obtenida para cada valor de K y en la clasificación asignada a los objetos de prueba.

En general, ambos enfoques produjeron resultados muy similares, ya que se utilizó el mismo algoritmo (K-NN) y la misma medida de distancia (euclidiana). Las pequeñas variaciones observadas pueden atribuirse al manejo interno de empates, el orden de los datos o detalles del proceso de evaluación.

En particular, para los valores $K = 3$, $K = 5$ y $K = 7$, ambos enfoques lograron clasificar correctamente 29 de los 30 objetos del conjunto de prueba, lo que equivale a una exactitud del 96.6667 %, redondeada a 96.67 %. Estas coincidencias confirman que la implementación en Python reproduce fielmente el comportamiento del clasificador K-NN de WEKA, utilizando la misma medida de distancia euclidiana y el mismo criterio de votación mayoritaria.

La Tabla 2 muestra la comparación directa entre ambas implementaciones.

Valor de K	**Exactitud (Python)**	**Exactitud (WEKA)**
3	96.67 %	96.67 %
5	96.67 %	96.67 %
7	96.67 %	96.67 %

Tabla 2: Comparación de exactitud obtenida entre Python y la herramienta de minería de datos.

8. Apéndices

A) Código fuente comentado del programa (Apendice A Codigo.py) y archivos generados.

B) Marco teórico (distancia euclidiana y K-NN).

C) Manual de usuario (uso de la GUI).

D) Manual técnico (software, librerías y WEKA).

Referencias

[1] GeeksforGeeks (2023). K-Nearest Neighbours (K-NN) Algorithm in Machine Learning. Recuperado de: `https://www.geeksforgeeks.org/machine-learning/k-nearest-neighbours/`. Fecha de consulta: 03/02/2026.

[2] IBM (2023). ¿Qué es K-Nearest Neighbors (K-NN) y cómo funciona? Recuperado de: `https://www.ibm.com/mx-es/think/topics/knn`. Fecha de consulta: 03/02/2026.

[3] WEKA (2024). WEKA Documentation. Recuperado de: `https://docs.weka.io/`. Fecha de consulta: 03/02/2026.