MODELADO DE SISTEMAS SOFTWARE

Informe: Práctica 7

Héctor Luís
Mariño Fernández
alu0100595604

ÍNDICE

Introducción

Esta práctica tiene como objetivo modelar mediante diagramas de secuencia y diagramas de comunicación el funcionamiento extendido de un sistema software para clasificación de instancias, basado en el método k-vecinos más cercanos (k-NN).

Se implementa en Java un clasificador k-NN que permite identificar los *k* vecinos más cercanos de una instancia y realizar la predicción de su clase. Además, se realiza la evaluación mediante la matriz de confusión y cálculo de la precisión predictiva.

Descripción de la clase KNNClassifier

La clase `KNNClassifier` es el núcleo funcional del sistema y está diseñada para realizar el proceso completo de clasificación mediante el método k-NN. Sus principales responsabilidades son:

- **Gestión del dataset**:
    - Carga y almacenamiento de las instancias del conjunto original.
    - División aleatoria del dataset en subconjuntos de entrenamiento y prueba, mediante el método `splitDatasetRatio(ratio, random, seed)`, que baraja las instancias para asegurar un reparto equilibrado.
- **Clasificación de instancias**:
    - Implementa el método `classify(instance)` que recibe una instancia de prueba y realiza el cálculo de distancias con respecto a las instancias de entrenamiento.
    - Utiliza la métrica de distancia configurada (por defecto, distancia Euclidiana).
    - Ordena las distancias para seleccionar los *k* vecinos más cercanos.
    - Emplea un sistema de votación ponderada basado en las distancias para predecir la etiqueta más probable con el método `predictClass(neighbors)`.
- **Evaluación del modelo**:
    - Durante un experimento, actualiza una matriz de confusión con las predicciones realizadas mediante `updateMatrix(actualClass, predictedClass)`.
    - Calcula la precisión predictiva global con `computeAccuracy()` para evaluar la efectividad del clasificador.

- **Parámetros configurables**:
    - Número *k* de vecinos a considerar.
    - Ratio de división para entrenamiento/prueba.
    - Semilla para la generación aleatoria y reproducibilidad.

La clase encapsula toda la lógica necesaria para preparar los datos, ejecutar el experimento de clasificación y evaluar resultados, sirviendo como la pieza central del sistema modelado.

Diagramas solicitados

1. Diagramas de Secuencia

- **Clasificación de una instancia**
  Se modela el proceso mediante el cual el sistema recibe una instancia de prueba, calcula las distancias a las instancias del conjunto de entrenamiento, determina los *k* vecinos más cercanos, realiza la votación ponderada y devuelve la clase predicha.

- **Generación aleatoria de los conjuntos de entrenamiento y prueba a partir del dataset original**
  Se representa cómo el sistema carga todas las instancias del dataset, realiza una división aleatoria basada en un ratio configurable y genera los conjuntos de entrenamiento y prueba para su posterior uso en el experimento.

---

2. Diagramas de Comunicación

- **Obtención de la matriz de confusión y cálculo de la precisión predictiva**
  Este diagrama muestra las interacciones necesarias para actualizar la matriz de confusión con las predicciones realizadas durante el experimento, así como el cálculo final de la precisión predictiva del clasificador.

Implementación Java

Se ha desarrollado una clase principal denominada `KNNClassifier` que incorpora las funcionalidades descritas, permitiendo:

- Cargar y dividir el dataset.
- Calcular distancias y seleccionar vecinos.
- Clasificar instancias.
- Evaluar el rendimiento mediante la matriz de confusión.

Referencias a diagramas

Para visualizar los diagramas detallados, consulta el directorio `/docs` en el repositorio

Estos diagramas ilustran claramente las interacciones y el flujo de información en el sistema para cada caso descrito.

Enlace al repositorio

Todo el código fuente, diagramas y documentación están disponibles en:
https://github.com/HectorLMF/Modelado-PRC-7