MODELADO DE SISTEMAS SOFTWARE

Informe: Práctica 3 - Clasificador KNN extendido.

Héctor Luís
Mariño Fernández
alu0100595604

ÍNDICE

1. Introducción al Proyecto....3
2. Enlaces al Proyecto en GitHub....3
3. Diagrama de Clases....3
4. Explicación de Clases, Atributos y Métodos....3
- Paquete model (Gestión de Datos)....3
  - Attribute....3
  - Instance....4
  - Dataset....4
- **Paquete classification (Reglas de Clasificación)....4**
  - ClassificationRule....4
  - MajorityVote....4
  - ThresholdVote....4
- Paquete distance (Métricas de Distancia)....5
  - DistanceMetric....5
  - EuclideanDistance....5
  - ManhattanDistance....5
  - ChebyshevDistance....5
- Paquete weighting (Pesado de Vecinos)....5
  - CaseWeightingStrategy....5
  - EqualWeighting....5
  - DistanceInverseWeighting....5
  - RankBasedWeighting....5
- Paquete knn (Clasificador k-NN)....6
  - KNNClassifier....6
- Paquete experiment (Gestión de Experimentos)....6
  - ExperimentManager....6

1. Introducción al Proyecto

El presente documento describe el desarrollo de un **clasificador k-NN** con configuración avanzada, permitiendo a los usuarios seleccionar diversos parámetros como la **métrica de distancia**, el **pesado de vecinos**, la **regla de clasificación**, y la **normalización de datos**. Además, el sistema es capaz de **almacenar y replicar experimentos** mediante archivos generados dinámicamente.

Este clasificador permite analizar **datasets tabulares** en formato `.csv`, procesando sus atributos y ejecutando comparaciones con instancias existentes. Como resultado, se obtiene una **predicción de clase** basada en el comportamiento de los vecinos más cercanos según la configuración definida por el usuario.

2. Enlaces al Proyecto en GitHub

El código fuente del proyecto y documentación detallada se encuentran disponibles en GitHub:

- **Repositorio principal:** https://github.com/HectorLMF/Modelado--PRC3/

3. Diagrama de Clases

El proyecto se estructura en **diferentes paquetes**, organizados según la funcionalidad de cada módulo. A continuación, se presenta el **diagrama de clases en varios formatos**, que ilustra la relación entre los componentes principales del sistema.

Ubicación: (en el repositorio) `docs/diagrams`

4. Explicación de Clases, Atributos y Métodos

🔹 Paquete model (Gestión de Datos)

Define la estructura de los datos utilizados en el clasificador.

Attribute

Clase que representa un **atributo** del dataset, incluyendo su tipo y si es la etiqueta de clase.

- **Atributos:**
  - `name` (`String`): Nombre del atributo.
  - `quantitative` (`boolean`): Indica si es numérico (`true`) o categórico (`false`).
  - `isClass` (`boolean`): Indica si es el atributo de clase (`true`).
- **Métodos:**
  - `getName()`: Devuelve el nombre del atributo.
  - `isQuantitative()`: Devuelve `true` si el atributo es numérico.
  - `isClass()`: Indica si el atributo es la etiqueta de clase.

`Instance`

Clase que representa una **instancia de datos** con sus valores.

- **Atributos:**
  - `values` (`List<Object>`): Lista de valores asociados a la instancia.
- **Métodos:**
  - `getValue(int index)`: Obtiene el valor en el índice especificado.
  - `setValue(int index, Object value)`: Asigna un nuevo valor al índice.
  - `getValues()`: Devuelve todos los valores de la instancia.

`Dataset`

Clase que almacena y administra el **conjunto de datos**.

- **Atributos:**
  - `data` (`List<Instance>`): Lista de instancias.
  - `attributes` (`List<Attribute>`): Lista de atributos.
  - `classAttributeIndex` (`int`): Índice del atributo de clase.
  - `normalizationMode` (`NormalizationMode`): Modo de normalización.
- **Métodos:**
  - `addInstance(Instance instance)`: Añade una instancia.
  - `normalize(NormalizationMode mode)`: Normaliza los atributos predictivos.
  - `getAttributeStats()`: Retorna estadísticas de atributos.

🔹 Paquete classification (Reglas de Clasificación)

Define cómo se votan las clases de los vecinos.

`ClassificationRule`

Interfaz que define una regla de clasificación.

`MajorityVote`

Clasifica con **mayoría simple**: la clase con más votos gana.

`ThresholdVote`

Clasifica según un **umbral mínimo** de votos.

- **Atributos:**
  - `threshold` (`float`): Porcentaje mínimo requerido para aceptar una clase.
- **Métodos:**
  - `predictClass(Map<String, Float> votes)`: Predice la clase basándose en el umbral.

🔹 Paquete `distance` (Métricas de Distancia)

`DistanceMetric`

Interfaz para definir una **métrica de distancia**.

`EuclideanDistance`

Implementa la distancia **Euclidiana**.

`ManhattanDistance`

Implementa la distancia **Manhattan**.

`ChebyshevDistance`

Implementa la distancia **Chebyshev**.

🔹 Paquete `weighting` (Pesado de Vecinos)

`CaseWeightingStrategy`

Interfaz para definir estrategias de **pesado de vecinos**.

`EqualWeighting`

Todos los vecinos tienen el mismo peso.

`DistanceInverseWeighting`

El peso es **inversamente proporcional a la distancia**.

`RankBasedWeighting`

Asigna pesos decrecientes según el orden del vecino.

🔹 Paquete `knn` (Clasificador k-NN)

`KNNClassifier`

Implementa el algoritmo **k-NN** con configuración avanzada.

- **Atributos:**
  - `k`: Número de vecinos considerados.
  - `distanceMetric`: Métrica de distancia usada.
  - `classificationRule`: Algoritmo de votación.
  - `caseWeightingStrategy`: Estrategia de pesado de vecinos.
- **Métodos:**
  - `classify(Instance instance)`: Predice la clase de una instancia.
  - `setK(int k)`: Ajusta el valor de `k`.

🔹 Paquete `experiment` (Gestión de Experimentos)

`ExperimentManager`

Clase que gestiona la ejecución y almacenamiento de **experimentos**.

- **Métodos:**
  - `splitDatasetRatio(float testRatio, boolean random, int seed)`: Divide el dataset.
  - `saveSplit()`: Guarda los conjuntos en `datasets/`.
  - `runExperiment(KNNClassifier classifier, String datasetPath)`: Ejecuta el experimento.
  - `generateExperimentReport()`: Genera un informe en `experiments_output/`.