CS2023 - Programación III

Práctica Calificada #1A

Rubén Rivas Medina

Setiembre 2025

Indicaciones Específicas

- **Tiempo límite:** 100 minutos
- **Formato de entrega:**
  - La solución debe implementarse en C++ y entregarse en archivos:
    - **transform.h**, **transform.cpp**
  - Los archivos deben nombrarse **EXACTAMENTE** como se indica (sensibles a mayúsculas)
- **Plataforma de entrega:**
  - Subir directamente los archivos a **https://www.gradescope.com** ó

Recomendaciones a tomar en cuenta que podrian afectar su calificación

- **Asegurese de seguir las convenciones que se le solicitan**, nombres de clases, namespace, métodos exactamente igual a los solicitados en las pruebas, un uso de estructuras o métodos extraños generara sospecha del uso de un Chatbox y penalización en la nota hasta CERO.
- **Si las funciones estan totalmente inoperativas** (requiere cambios estructurales mayores para funcionar) **o contiene errores graves** (crashes, implementación incorrecta de funcionalidad básica), La calificación podria ser reducida hasta un **60%** del valor total de lo calificado.

1 Transform: Normalize, MinMax

Se desea implementar un sistema de preprocesamiento de datos tabulares que permita aplicar transformaciones polimórficas sobre un dataset. Cada fila del dataset está formada por valores numéricos (features) . Se requiere definir una jerarquía de clases que incluya:

- Una estructura **Sample**, que representa una fila del dataset.
- Una clase abstracta **Transform**, que define la interfaz polimórfica:
  - **virtual Sample apply(const Sample& s) const = 0;**
  - **virtual std::string name() const = 0;**
- La clase derivada **Normalize**, que reescala cada columna para que tenga media 0 y desviación estándar 1.
  - Almacena internamente: means, stds, epsilon.
  - Puede construirse de dos maneras:
    - Con parámetros explícitos (ya calculados).
    - Mediante el método estático **fit(const Dataset&, const double epsilon=1e-9)** que obtiene medias y desviaciones de cada columna.
  - Fórmulas aplicadas por columna j con N filas:

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij}$$

$$\sigma_j = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_{ij}-\mu_j)^2}$$

$$x'_{ij} = \frac{x_{ij}-\mu_j}{\sigma_j+\varepsilon}$$

  - Ejemplo: para la columna $[1,3,5]$ se obtiene

$$\mu = \frac{1+3+5}{3} = 3, \quad \sigma = \sqrt{\frac{(1-3)^2+(3-3)^2+(5-3)^2}{3-1}} = 2$$

$$x' = \left[\tfrac{1-3}{2},\ \tfrac{3-3}{2},\ \tfrac{5-3}{2}\right] = [-1,\ 0,\ 1]$$

- La clase derivada **MinMax**, que reescala cada columna al rango $[0,1]$.
  - Almacena internamente: mins, maxs, epsilon.
  - Puede construirse de dos maneras:
    - Con parámetros explícitos (ya calculados).
    - Mediante el método estático **fit(const Dataset&, const double epsilon=1e-9)** que obtiene mínimos y máximos de cada columna.
  - Fórmula aplicada:

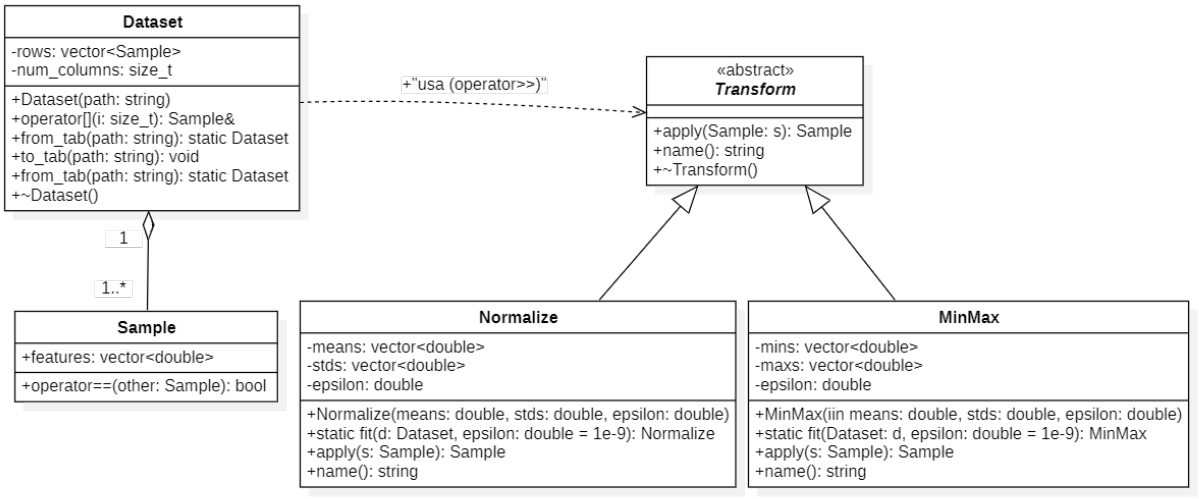
$$x'_{ij} = \frac{x_{ij} - \text{mín}_j}{\text{máx}_j - \text{mín}_j + \varepsilon}$$

- Una clase **Dataset**, que almacena dinámicamente un conjunto de Sample, y permite aplicar transformaciones usando sobrecarga de operadores.

Estructura de Sample:

```
struct Sample {
  std::vector<double> features;
  bool operator==(const Sample& other) const {
    return features == other.features && label == other.label;
  }
};
```

Diagrama de clases:



El sistema debe soportar:

- Lectura y escritura de archivos tabulares **.tab**.
- Operadores sobrecargados:
  - **Dataset::operator»(const Transform&)** para aplicar transformaciones.
  - **Dataset::operator[](size_t)** para acceder a filas.
  - **operator«(ostream&, const Dataset&)** para imprimir un resumen del dataset.

Ejemplos paso a paso

Ejemplo A: Normalize (fit automático)

Entrada (in1.tab):

```
1.0 2.0
3.0 6.0
5.0 10.0
```

Cálculos:

$$\mu_1 = 3.0,\ \ \sigma_1 = 2.0 \qquad \mu_2 = 6.0,\ \ \sigma_2 = 4.0$$

Salida esperada (Normalize):

```
-1.0 -1.0
0.0 0.0
1.0 1.0
```

Ejemplo B: MinMax (fit automático)

Entrada (in2.tab):

```
1.0 2.0
3.0 6.0
5.0 10.0
```

Cálculos:

$$\min_1 = 1.0,\ \ \max_1 = 5.0 \qquad \min_2 = 2.0,\ \ \max_2 = 10.0$$

Salida esperada (MinMax):

```
0.0 0.0
0.5 0.5
1.0 1.0
```

Ejemplo C: Columna constante (uso de ε)

Entrada (in3.tab):

```
2.0 5.0
2.0 5.0
2.0 5.0
```

Normalize:

$$\mu = 2.0,\ \ \sigma = 0 \quad \Rightarrow \quad x'_{ij} = \frac{x_{ij} - 2.0}{0 + \varepsilon} = 0$$

MinMax:

$$\min = \max \quad \Rightarrow \quad x'_{ij} = \frac{x_{ij} - \min}{0 + \varepsilon} = 0$$

Salida esperada (ambas transformaciones):

```
0.0 0.0
0.0 0.0
0.0 0.0
```

Casos de prueba

Use Case #1: Normalize con fit automático

```
Dataset d("in1.tab");

Normalize norm = Normalize::fit(d, true);
Dataset out = d >> norm;

std::cout << out;
// -1.0 -1.0
// 0.0 0.0
// 1.0 1.0
```

Use Case #2: MinMax con fit automático

```
Dataset d("in1.tab");

MinMax mm = MinMax::fit(d);
Dataset out = d >> mm;

std::cout << out;
// 0.0 0.0
// 0.5 0.5
// 1.0 1.0
```

Use Case #3: Constructores con parámetros explícitos

```
Dataset d("in1.tab");

// Normalize con medias y desv conocidas
Normalize norm({3.0, 6.0}, {2.0, 4.0});
Dataset out1 = d >> norm;

// MinMax con min y max conocidos
MinMax mm({1.0, 2.0}, {5.0, 10.0});
Dataset out2 = d >> mm;

// Esperado: mismos resultados que con fit()
```

Use Case #4: Error de archivo vacío

```
try {
    Dataset d("empty.tab");
    Normalize n = Normalize::fit(d);
} catch(const std::exception& e) {
    std::cout << e.what();
}
// Esperado: excepción clara indicando dataset vacío
```

Use Case #5: Operador []

```
Dataset d("in1.tab");
Sample s = d[0];
std::cout << s.features.size();
// Esperado: número de columnas del dataset (ej. 2)
```

Use Case #6: Columna constante (uso de epsilon)

```
Dataset d("in3.tab"); // todas las filas iguales
Normalize n = Normalize::fit(d);
Dataset out1 = d >> n;

MinMax m = MinMax::fit(d);
Dataset out2 = d >> m;

// Esperado: todas las salidas son ceros
```

Rúbricas de evaluación

Problema

Criterio	Logrado	Parcial	Mínimo	No logrado	No realizado
Definición correcta de la jerarquía Transform y derivadas	4.0	2.0	1	0.5	0.0
Implementación de Normalize con medias y desviaciones	4.0	2.0	1	0.5	0.0
Implementación de MinMax con rangos [0,1]	4.0	2.0	1	0.5	0.0
Sobrecarga de operadores (>>, <<, []) funcionales	4.0	2.0	1	0.5	0.0
Entrada/salida de archivos y manejo de errores	4.0	2.0	1	0.5	0.0
Total	**20.0**				