INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

INTELIGENCIA ARTIFICIAL

PRÁCTICA 8

DATASETS

NOMBRE DEL ALUMNO: GARCÍA QUIROZ GUSTAVO IVAN

NOMBRE DEL PROFESOR: GARCÍA FLORIANO ANDRES

GRUPO: 6CV3

FECHA DE ENTREGA DEL REPORTE: 31/10/2024

1 Índice

2 Marco teórico

2.1 Dataset Iris Plant

l conjunto de datos flor Iris o conjunto de datos iris de Fisher es un conjunto de datos multivariante introducido por Ronald Fisher en su artículo de 1936, The use of multiple measurements in taxonomic problems (El uso de medidas múltiples en problemas taxonómicos) como un ejemplo de análisis discriminante lineal. A veces, se llama Iris conjunto de datos de Anderson porque Edgar Anderson coleccionó los datos para cuantificar la variación morfológica de la flor Iris de tres especies relacionadas.

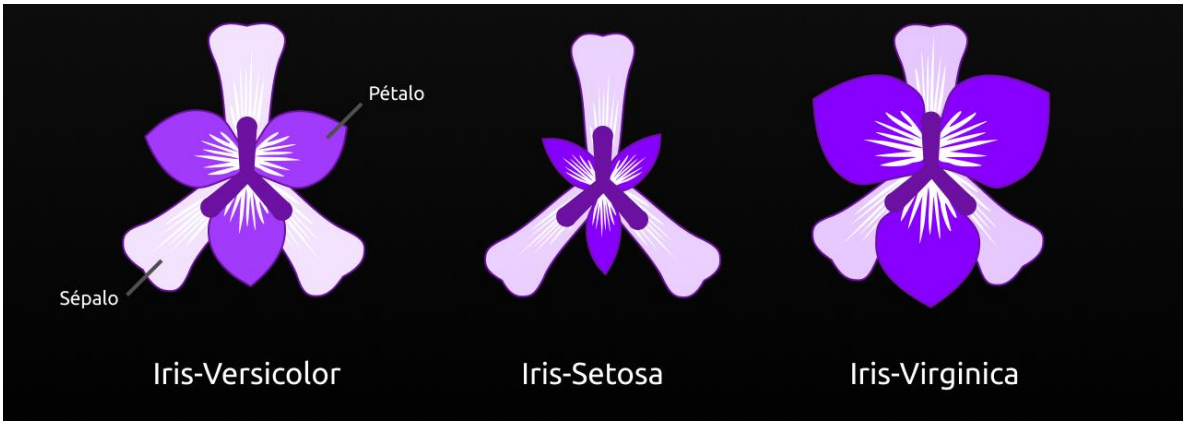

Figura 1 Planta Iris:

El conjunto de datos contiene 50 muestras de cada una de tres especies de Iris (Iris setosa, Iris virginica e Iris versicolor). Se midió cuatro rasgos de cada muestra: el largo y ancho del sépalo y pétalo, en centímetros. Basado en la combinación de estos cuatro rasgos, Fisher desarrolló un modelo discriminante lineal para distinguir entre una especie y otra.

	sepal_length	sepal_width	petal_length	petal_width	species
114	5.8	2.8	5.1	2.4	virginica
62	6.0	2.2	4.0	1.0	versicolor
33	5.5	4.2	1.4	0.2	setosa
107	7.3	2.9	6.3	1.8	virginica
7	5.0	3.4	1.5	0.2	setosa

Figura 2 Conjunto de datos.

3 Introducción

La presente práctica se enfoca en el procesamiento y análisis del dataset Iris Plant, un conjunto de datos ampliamente utilizado en el campo del aprendizaje automático y la estadística. Este dataset contiene información sobre diferentes especies de flores iris, incluyendo mediciones específicas de sus características físicas como la longitud y ancho del sépalo y pétalo.

El objetivo principal de esta práctica consiste en desarrollar programas en tres lenguajes de programación diferentes: C, Java y Python, para leer y procesar el archivo “bezdekIris.data”. Este archivo, disponible en el repositorio UCI Machine Learning, contiene 150 registros con cinco campos cada uno, proporcionando una base sólida para el estudio de técnicas de procesamiento de datos en diferentes entornos de programación.

La implementación en múltiples lenguajes permite explorar y comparar diferentes enfoques para el manejo de datos. El lenguaje C ofrece un control directo sobre la memoria y las operaciones básicas de lectura de archivos. Java proporciona un enfoque orientado a objetos con estructuras de datos dinámicas. Python, mediante la biblioteca pandas, presenta una solución moderna para el análisis de datos con funcionalidades integradas de procesamiento estadístico.

La práctica aborda aspectos fundamentales del procesamiento de datos, incluyendo la lectura de archivos CSV, el almacenamiento estructurado de información, la conversión de tipos de datos y la verificación de la integridad de los datos. Estos conceptos son esenciales en el desarrollo de aplicaciones que manejan conjuntos de datos estructurados.

El dataset Iris representa un caso de estudio ideal para esta práctica debido a su estructura bien definida y su tamaño manejable. Los datos incluyen tanto valores numéricos como clasificaciones categóricas, lo que permite explorar diferentes aspectos del procesamiento de datos en cada lenguaje de programación. La naturaleza estructurada del dataset facilita la comparación directa entre las diferentes implementaciones y sus respectivos enfoques para el manejo de datos.

4 Material y equipo.

Para esta práctica se usaron las siguientes herramientas de software y hardware necesarias para realizar la práctica.

4.1.1 Hardware

- Computadora.

4.1.2 Software

- Visual Studio Code.
- Phyton 3.

5 Desarrollo de la práctica.

El desarrollo de esta práctica se centró en la lectura y procesamiento del conjunto de datos Iris, un dataset clásico en el campo del aprendizaje automático. El archivo “bezdekIris.data” contiene mediciones de diferentes características de flores iris, incluyendo longitud y ancho del sépalo y pétalo, así como la especie correspondiente.

5.1 Proceso de Obtención del Dataset

Se accedió al repositorio UCI Machine Learning para obtener el conjunto de datos Iris Plant. El archivo descargado “bezdekIris.data” se encontraba en formato CSV sin encabezados, conteniendo 150 registros con 5 campos cada uno: cuatro mediciones numéricas y una clasificación de especie.

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
```

Figura 3 Archivo “bezdekIris.data”.

5.2 Implementación en Lenguaje C

El programa en C se desarrolló utilizando una estructura fundamental para el manejo de archivos de datos. La implementación se centró en la creación de una matriz bidimensional para almacenar los datos numéricos del dataset Iris. El código utiliza la función FILE para abrir el archivo "bezdekIris.data" en modo lectura. La matriz se definió con dimensiones fijas de 150 filas y 5 columnas, correspondientes al tamaño del dataset.

La parte central del código implementa un bucle while que lee cada línea del archivo utilizando fgets(). Para cada línea, se emplea sscanf() para extraer los valores numéricos y almacenarlos en la matriz data[][]. La última columna, que contiene el

nombre de la especie, se maneja de manera separada utilizando un buffer de caracteres. El programa verifica la lectura exitosa imprimiendo las primeras 5 filas de datos como control de calidad.

```c
int main() {
    char line[100];
    float data[MAX_ROWS][MAX_COLS];
    int row = 0;

    file = fopen("bezdekIris.data", "r");
    if (file == NULL) {
        printf("Error: No se puede abrir el archivo.\n");
        return 1;
    }

    char label[20]; // Buffer to store the string
    while (fgets(line, sizeof(line), file) && row < MAX_ROWS) {
        sscanf(line, "%f,%f,%f,%f,%s",
               &data[row][0], &data[row][1], &data[row][2], &data[row][3], label);
        row++;
    }

    fclose(file);
```

Figura 4 Código en lenguaje C.

5.3 Implementación en Java

La implementación en Java se desarrolla mediante una clase denominada IrisDataset dentro del paquete com.ejemplo. El programa está diseñado para leer y procesar el archivo "bezdekIris.data", utilizando estructuras de datos dinámicas para almacenar la información del dataset. La clase emplea la biblioteca java.io para el manejo de archivos y java.util.ArrayList para el almacenamiento dinámico de datos.

El programa utiliza una estructura ArrayList anidada para almacenar los datos numéricos del dataset. La variable data se declara como ArrayList<ArrayList<Float>>, lo que permite almacenar múltiples filas de valores flotantes. Esta estructura proporciona flexibilidad para manejar un número variable de registros sin necesidad de definir un tamaño fijo.

La lectura del archivo se realiza mediante un BufferedReader, que se inicializa con un FileReader apuntando al archivo "bezdekIris.data". El programa implementa el patrón try-with-resources para garantizar el cierre adecuado del archivo después de su uso. La lectura se realiza línea por línea utilizando el método readLine(), procesando cada línea hasta alcanzar el final del archivo.

El procesamiento de cada línea incluye múltiples verificaciones de validación. El programa verifica si la línea está vacía mediante el método trim().isEmpty(),

omitiendo las líneas vacías para mantener la integridad de los datos. Cada línea se divide en valores individuales utilizando el método split(","), y se verifica que existan al menos 4 valores numéricos.

```java
String fileName = "src/bezdekIris.data";
ArrayList<ArrayList<Float>> data = new ArrayList<>();

try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
    String line;
    while ((line = br.readLine()) != null) {
        if (line.trim().isEmpty()) {
            System.out.println(x:"Línea vacía encontrada y omitida.");
            continue;
        }

        String[] values = line.split(regex:",");
        if (values.length < 4) {
            System.out.println("Línea con valores insuficientes encontrada y omitida: " + line);
            continue;
        }

        ArrayList<Float> row = new ArrayList<>();
        for (int i = 0; i < 4; i++) {
            if (!values[i].trim().isEmpty()) {
                try {
                    row.add(Float.parseFloat(values[i].trim()));
                } catch (NumberFormatException e) {
                    System.out.println("Error al parsear el valor: " + values[i]);
                }
            } else {
                System.out.println("Valor vacío encontrado en la línea: " + line);
            }
        }
        data.add(row);
    }
} catch (IOException e) {
    System.out.println("Error al leer el archivo: " + e.getMessage());
}
```

Figura 5 Código en Java.

5.4 Implementación en Python

La implementación en Python se realizó utilizando la biblioteca pandas, que proporcionó una solución robusta para la lectura y análisis de datos. Se definieron nombres de columnas específicos (sepal_length, sepal_width, petal_length, petal_width, species) para facilitar el procesamiento posterior. La lectura del archivo se realizó mediante pd.read_csv().

```python
import pandas as pd
import numpy as np

# Definir los nombres de las columnas
columnas = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
# Leer el archivo CSV sin encabezados y asignar nombres de columnas
df_iris = pd.read_csv('bezdekIris.data', header=None, names=columnas)
```

Figura 6 Código en Python.

5.5 Resultados obtenidos

Implementación en C

```
PS C:\Users\ivan-\PycharmProjects\AI\8_Datasets> cd 'c:\Users\ivan-\PycharmProjects\AI\8_Datasets\output'
PS C:\Users\ivan-\PycharmProjects\AI\8_Datasets\output> & .\'iris.exe'
5.100000 3.500000 1.400000 0.200000
4.900000 3.000000 1.400000 0.200000
4.700000 3.200000 1.300000 0.200000
4.600000 3.100000 1.500000 0.200000
5.000000 3.600000 1.400000 0.200000
PS C:\Users\ivan-\PycharmProjects\AI\8_Datasets\output>
```

Figura 7 Implementación en C

Implementación en Java

```
PS C:\Users\ivan-\PycharmProjects\AI\8_Datasets>  & 'C:\Program Files\Amazon Corretto\jdk21.0.4_7\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\ivan-\AppData\Roaming\Code\User\workspaceStorage\bf92ce4b5f368169d7a04647fe949355\redhat.java\jdt_ws\8_Datasets_df666b87\bin' 'com.ejemplo.IrisDataset'
Línea vacía encontrada y omitida.
[5.1, 3.5, 1.4, 0.2]
[4.9, 3.0, 1.4, 0.2]
[4.7, 3.2, 1.3, 0.2]
[4.6, 3.1, 1.5, 0.2]
[5.0, 3.6, 1.4, 0.2]
PS C:\Users\ivan-\PycharmProjects\AI\8_Datasets>
```

Figura 8 Implementación en Java

Implementación en Python

```
PS C:\Users\ivan-\PycharmProjects\AI\8_Datasets> & C:/Users/ivan-/AppData/Local/Programs/Python/Python312/python.exe c:/Users/ivan-/PycharmProjects/AI/8_Datasets/iris.py
Primeras 5 filas del dataset:
   sepal_length  sepal_width  petal_length  petal_width      species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
```

Figura 9 Implementación en Python

6 Conclusiones

La implementación de la lectura del dataset Iris en tres diferentes lenguajes de programación demuestra las características y capacidades particulares de cada uno. El lenguaje C proporciona un control directo sobre la memoria y una ejecución eficiente, siendo adecuado para sistemas con recursos limitados o donde el rendimiento es crítico.

La versión en Java presenta una solución robusta con manejo de excepciones y tipos de datos bien definidos. La utilización de estructuras de datos dinámicas como ArrayList facilita el manejo de datos sin preocuparse por el tamaño inicial del dataset. El enfoque orientado a objetos de Java permite una mejor organización del código y mantenibilidad a largo plazo.

La implementación en Python mediante pandas resulta la más eficiente en términos de desarrollo y análisis de datos. La biblioteca proporciona funcionalidades integradas para el análisis estadístico y la manipulación de datos, reduciendo significativamente la cantidad de código necesario. Esta versión facilita la exploración y el análisis del dataset con funciones predefinidas para estadísticas descriptivas y visualización de datos.

Para futuras implementaciones similares, se recomienda seleccionar el lenguaje de programación según los requisitos específicos del proyecto. Si el proyecto requiere un control preciso sobre los recursos del sistema o se ejecutará en un entorno con limitaciones de memoria, la implementación en C sería la más apropiada.

7 Referencias.

- *UCI machine learning repository*. (s/f). Uci.edu. Recuperado el 28 de octubre de 2024, de https://archive.ics.uci.edu/dataset/53/iris
- Wikipedia contributors. (s/f). *Conjunto de datos flor iris*. Wikipedia, The Free Encyclopedia. https://es.wikipedia.org/w/index.php?title=Conjunto_de_datos_flor_iris&oldid=159237500

8 Código

-