

# PROYECTO #2

## JAVA

LUIS BUENO  
ALEJANDRO TRIANA  
SEBASTIAN RINCON  
JOSE PULIDO

# REGRESION LINEAL

```
while (getline(file, line)) {
    stringstream ss(line);
    string nombreProducto;
    string precioStr;
    double precio;

    if (!getline(ss, nombreProducto, ',')) continue;
    if (!getline(ss, precioStr, ',')) continue;

    try {
        precio = stod(precioStr);
    } catch (const invalid_argument &) {
        cerr << "Error: Precio inválido en el
producto " << nombreProducto << endl;
        continue;
    }

    productos[nombreProducto].nombre =
    nombreProducto;
```

El código toma una lista de precios históricos (`vector<double> precios`) y calcula la regresión lineal.

Luego, calcula la predicción para el siguiente valor en la serie de datos (por ejemplo, el próximo precio esperado basado en los precios anteriores).

El programa guarda el resultado de la predicción en un archivo de texto llamado `resultado_prediccion.txt`. Este archivo será leído por el código en Java para obtener el precio predicho

```
        ProcessBuilder compilePB = new
ProcessBuilder("g++", "C++/regresion.cpp", "-o",
"C++/regresion");
        compilePB.inheritIO();
        Process compileProcess = compilePB.start();
compileProcess.waitFor();

        if (compileProcess.exitValue() != 0) {
            System.out.println("Error al compilar el
programa C++");
            return 0;
        }
```

En Java, se puede invocar este código C++ utilizando la clase `ProcessBuilder`. Este método permite compilar y ejecutar programas externos (como el código en C++) y luego capturar los resultados.

# REGRESION LINEAL

```
        BufferedReader br = new BufferedReader(new
FileReader("C++/resultado_prediccion.txt"));
        StringBuilder resultado = new StringBuilder();
String line;

while ((line = br.readLine()) != null) {
    resultado.append(line).append("\n");
}
br.close();

String[] lines =
resultado.toString().split("\n");
if (lines.length == 0) {
    System.out.println("El archivo está
vacío. No se puede obtener el precio.");
    return 0;
}

String lastLine = lines[lines.length -
1].trim();
String[] parts = lastLine.split(":");
```

- ProcessBuildercompilePB= new ProcessBuilder  
Este comando compila el archivo regresion.cpp usando g++ y genera un ejecutable llamado regresion.
- compileProcess.waitFor(): Espera a que la compilación termine antes de continuar

ProcessBuilder runPB = new ProcessBuilder  
Ejecuta el programa regresion generado.

- runProcess.waitFor(): Espera a que el programa termine de ejecutarse

- BufferedReader br = new BufferedReader  
Abre el archivo resultado\_prediccion.txt generado por el código C++.
- Lee el contenido del archivo para obtener el valor predicho
- Extrae el precio predicho de la última línea del archivo, que está en el formato Predicción del precio: <valor>.
- Devuelve el valor convertido a un tipo double para su uso en el programa

El programa permite ver productos disponibles, comprar productos, añadir productos a la lista de la tienda, y ver el registro de ventas de lo que se ha comprado.

## ARRAYLIST

- productosElectronics
- productosAlimenticios
- registro

## FUNCIONES

- verProductos()
- venderProductos()
- anadirProductos()
- imprimirRegistro()

# MAIN JAVA



```
//Creamos los ArrayList de los productos
ArrayList<ProductoElectronico> productosElectronicos;
ArrayList<ProductoAlimenticio> productosAlimenticios;
ArrayList<Producto> registro;

//Constructor
public Main(ArrayList<ProductoElectronico>
productosElectronicos, ArrayList<ProductoAlimenticio>
productosAlimenticios, ArrayList<Producto> registro)
{
    this.productosElectronicos=productosElectronicos;
    this.productosAlimenticios=productosAlimenticios;
    this.registro=registro;
}
```

- ProductosElectronics: Lista donde se almacenan los productos electrónicos.
- ProductosAlimenticios: Lista donde se almacenan los productos alimenticios.
- Registro: Lista que guarda un historial de las ventas realizadas.

El constructor de la clase Main inicializa las listas de productos y el registro de ventas que se pasan como parámetros. Básicamente, prepara las listas para que el programa funcione

```
//Imprimimos ambos arreglos de productos:  
public void verProductos()  
{  
    StringBuilder mensaje = new StringBuilder();  
    mensaje.append("Productos Electrónicos:\n");  
    for (ProductoElectronico productoElectronico :  
        productosElectronicos)  
    {  
  
        mensaje.append(productoElectronico.verInfo()).append("\n");  
    }  
    mensaje.append("\nProductos Alimenticios:\n");  
    for (ProductoAlimenticio productoAlimenticio :  
        productosAlimenticios)  
    {  
  
        mensaje.append(productoAlimenticio.verInfo()).append("\n");  
    }  
    JOptionPane.showMessageDialog(null, mensaje.toString(),  
        "Lista de Productos", JOptionPane.INFORMATION_MESSAGE);  
}
```

- Crea un mensaje con el título "Productos Electrónicos" y luego agrega la información de todos los productos electrónicos.
- Agrega otro título, "Productos Alimenticios", y luego incluye la información de los productos alimenticios.
- Muestra el mensaje completo con todos los productos en una ventana emergente utilizando JOptionPane.showMessageDialog().

```
//Vender productos al usuario
public void venderProductos(int tipo, String nombre, int
cantidad)
{
    //Vendemos el producto en base al tipo
    if(tipo==1)
    {
        for (ProductoElectronico productoElectronico :
productosElectronicos)
        {
            if (productoElectronico.nombre.equals(nombre))
            {
                anadirRegistro(nombre, cantidad);
                productoElectronico.venderProducto(cantidad);
            }
        }
    }
    else if (tipo==2)
    {
        for (ProductoAlimenticio productoAlimenticio :
productosAlimenticios)
        {
            if (productoAlimenticio.nombre.equals(nombre))
            {
                anadirRegistro(nombre, cantidad);
                productoAlimenticio.venderProducto(cantidad);
            }
        }
    }
}
```

## ELECTRONICOS

Luego, se recorre la lista productosElectronicos para buscar un producto que tenga el mismo nombre que el que el usuario quiere comprar

- Llama a anadirRegistro(nombre, cantidad), lo que añade la compra al registro de ventas.
- Luego, llama a venderProducto(cantidad) del producto electrónico, lo que reduce la cantidad de ese producto disponible en la tienda.

## ALIMENTICIOS

Se recorre la lista productosAlimenticios para buscar el producto con el nombre que el usuario indicó.

- Añade la venta al registro con anadirRegistro(nombre, cantidad).
- Reduce la cantidad disponible del producto alimenticio llamando a venderProducto(cantidad).

```
//Añadimos productos en base a lo escogido
public void anadirProductos(int tipo) {
    String nombre = JOptionPane.showInputDialog("Ingrese el
nombre del nuevo producto:");
    String cantidadStr = JOptionPane.showInputDialog("Ingrese
la cantidad:");
    int cantidad = Integer.parseInt(cantidadStr);

    boolean encontrado = false;

    // Verifica si el producto ya existe en productos
alimenticios
    for (ProductoAlimenticio productoAlimenticio :
productosAlimenticios) {
        if (productoAlimenticio.nombre.equals(nombre)) {
            productoAlimenticio.cantidad += cantidad;
            encontrado = true;
            break;
        }
    }
}
```

Verificamos si el producto alimenticio ya existe en la tienda.

```
//Añade elementos a nuestro registro
public void anadirRegistro(String nombre, int cantidad)
{
    Producto producto = new Producto(nombre, cantidad);
    registro.add(producto);
}

//Nos muestra los distintos registros
public void imprimirRegistro()
{
    StringBuilder mensaje = new StringBuilder();
    for (Producto producto : registro)
    {
        mensaje.append(producto.toString());
    }
}
```

Si no lo encontramos en esta lista, se verifica si el producto es electrónico, y si no se encuentra en ninguna lista, se añadirá como un producto nuevo.

Si ya existe, actualizamos la cantidad sumando la nueva cantidad al inventario

The screenshot shows a Java development environment with several windows:

- Code Editor:** Displays the `Main.java` file with Java code for a regression application.
- Console Window:** Shows the application's interaction with the user:
  - La predicción: Manzana
  - Ingrese la cantidad para la predicción: 2
  - Que tipo de producto es:
  - CPU
  - Camera
  - Case
  - Charger
  - Cooling Fan
  - External Hard Drive
  - Game Console
  - Graphics Card
  - Headphones
  - Keyboard
  - Laptop
  - Microphone
  - Monitor
  - Motherboard
  - Mouse
  - Power Supply
  - Printer
  - RAM
  - Router
  - SSD
  - Smartphone
  - Smartwatch
  - Speaker
  - Tablet
  - USB Drive
- Output Window:** Displays the results of the prediction:
  - Productos Electrónicos:**
    - Nombre: Pc
    - Cantidad: 1
    - Precio Base: 728.164
    - Precio Total: 728.164
  - Productos Alimenticios:**
    - Nombre: Manzana
    - Cantidad: 2
    - Precio Base: 14.90645
    - Precio Total: 29.8129

**El código toma una lista de precios históricos (vector<double> precios) y calcula la regresión lineal.**

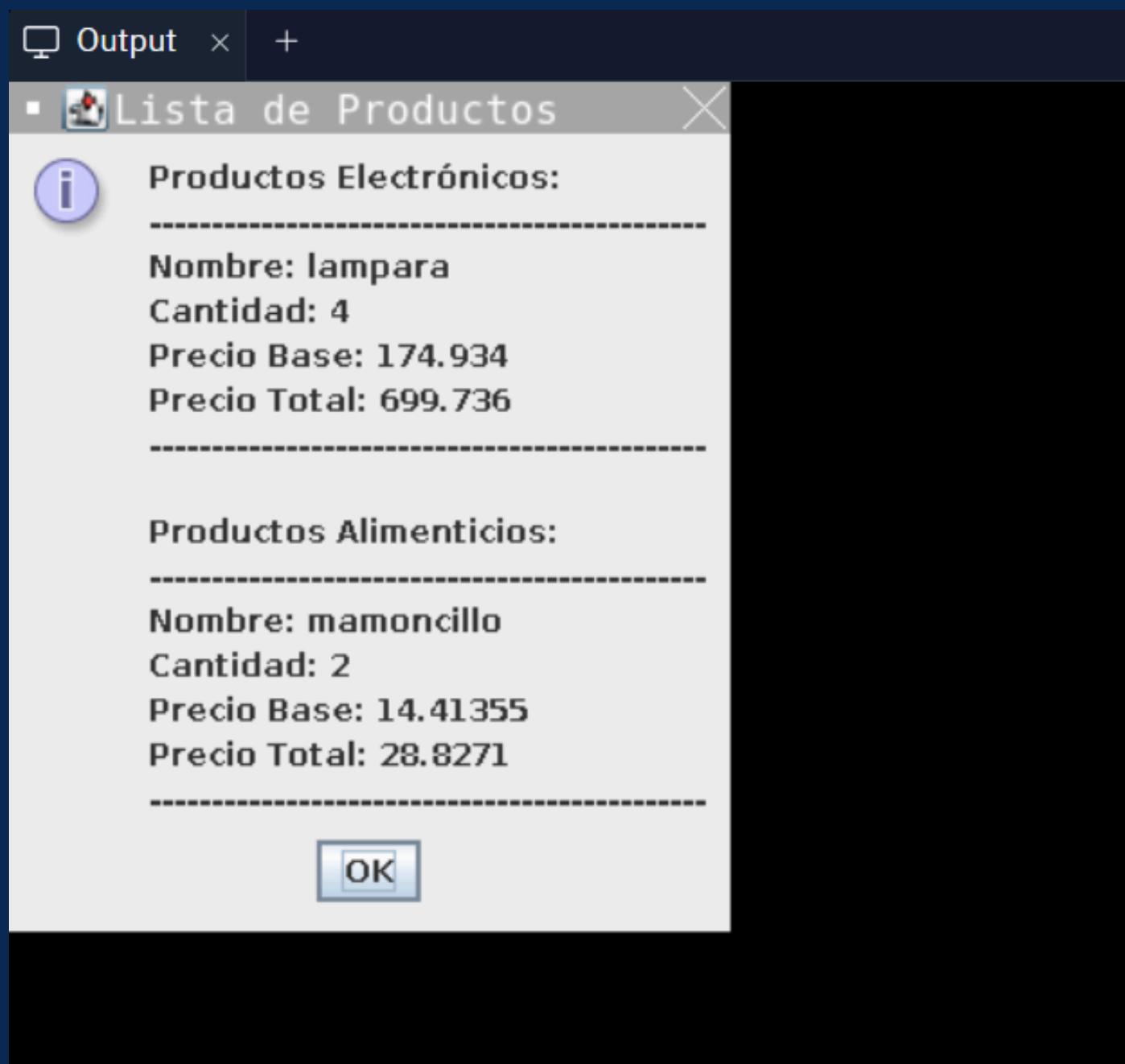
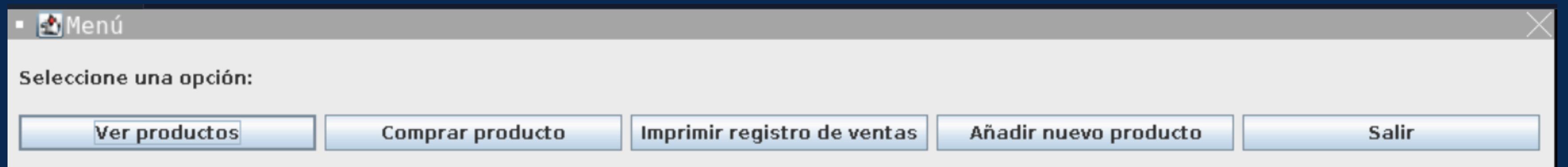
**Luego, calcula la predicción para el siguiente valor en la serie de datos (por ejemplo, el próximo precio esperado basado en los precios anteriores).**

**El programa guarda el resultado de la predicción en un archivo de texto llamado resultado\_prediccion.txt. Este archivo será leído por el código en Java para obtener el precio predicho**

regresion.cpp | regresion | Console | Output |

src > main > java > Main.java

```
1 import java.util.ArrayList;
2 import javax.swing.*;
3
4 public class Main
5 {
6     //Creamos los ArrayList
7     de los productos
8
9     ArrayList<ProductoElectronico>
10    productosElectronics;
11
12    ArrayList<ProductoAlimenticio>
13    productosAlimenticios;
14    ArrayList<Producto>
15    registro;
16
17    //Constructor
18    public
19    Main(ArrayList<ProductoElectronico>
20        productosElectronics,
21        ArrayList<ProductoAlimenticio>
22        productosAlimenticios,
23        ArrayList<Producto>
24        registro)
25    {
26
27        Show Only Latest | Clear History | Run | Ask AI | i
28
29        La predicción: Manzana
30        Ingrese la cantidad para la predicción: 2
31
32        Que tipo de producto es:
33        CPU
34        Camera
35        Case
36        Charger
37        Cooling Fan
38        External Hard Drive
39        Game Console
40        Graphics Card
41        Headphones
42        Keyboard
43        Laptop
44        Microphone
45        Monitor
46        Motherboard
47        Mouse
48        Power Supply
49        Printer
50        RAM
51        Router
52        SSD
53        Smartphone
54        Smartwatch
55        Speaker
56        Tablet
57        USB Drive
58        Seleccione el nombre del producto para hacer la predicción:
59        CPU
60        Ingrese la cantidad para la predicción: 1
61
62        i | Productos Electrónicos:
63        -----
64        Nombre: Pc
65        Cantidad: 1
66        Precio Base: 728.164
67        Precio Total: 728.164
68
69        -----
70        Productos Alimenticios:
71        -----
72        Nombre: Manzana
73        Cantidad: 2
74        Precio Base: 14.90645
75        Precio Total: 29.8129
76
77        -----
78        OK
```



```
=====  
Menú  
=====  
1. Ver productos  
2. Comprar producto  
3. Imprimir registro de ventas  
4. Añadir nuevo producto  
0. Salir  
Seleccione una opción: 4  
Ingrese el tipo de producto (1: Electrónico, 2: Alimenticio): 2  
Ingrese el nombre del nuevo producto: Manzana  
Ingrese la cantidad: 5  
Parece que tenemos un producto nuevo, seleccione cual es:  
Productos disponibles:  
Arroz  
Arvejas  
Banana  
Cebolla  
Huevos  
Leche  
Mantequilla  
Manzana  
Naranja  
Pan  
Papa  
Pasta  
Pepino  
Pescado  
Pollo  
Queso  
Res  
Tomate  
Yogur  
Zanahoria  
Seleccione el nombre del producto para hacer la predicción: Res  
Ingrese la cantidad para la predicción: 8  
Producto añadido exitosamente.
```

# ¡MUCHAS GRACIAS!

PROGRAMACION AVANZADA

```
render() {
  return (
    <React.Fragment>
      <div className="py-5">
        <div className="container">
          <Title name="our" title="product">
            <div className="row">
              <ProductConsumers>
                {(value) => {
                  |   |   console.log(value)
                  |   |
                }}
              </ProductConsumers>
            </div>
          </div>
        </div>
      <React.Fragment>
```