Informe Técnico: Sistema de Análisis y Predicción de Ventas para una Tienda Virtual

Miembros: Joaquín Bermúdez Murcia, Juan Manuel Martinez Sánchez

Introducción

El sistema desarrollado despliega un menú principal al ejecutar el archivo "main.py". Este menú interactivo incluye **ocho opciones** diseñadas para proporcionar al usuario información clave y funcionalidades avanzadas relacionadas con la gestión de ventas, productos y clientes. A continuación, se describen detalladamente las características técnicas y el funcionamiento de cada opción del menú.

Opción 1: Resumen Inicial

Al seleccionar esta opción, el sistema genera un **resumen inicial** que incluye información esencial sobre el estado actual de las ventas y clientes. Los datos mostrados comprenden:

- Cantidad total de ventas realizadas.
- Número de clientes registrados.
- Cantidad de productos disponibles en inventario.
- Identificación del producto con mayor cantidad de ventas.

Además, se genera una **relación de ventas y clientes**. Esta funcionalidad crea un archivo actualizado que incluye una columna adicional donde se asocia cada venta con el ID de un cliente seleccionado aleatoriamente.

Opción 2: Estimación de Ventas Futuras

Esta funcionalidad calcula la **proyección de ventas futuras** para un producto específico mediante un algoritmo basado en la técnica de diseño **divide y vencerás**. El proceso se detalla a continuación:

1. Se organiza cronológicamente un array con las cantidades de ventas del producto.
2. El array se divide en subarrays más pequeños.
3. Para cada subarray, se calcula el **crecimiento porcentual** entre los valores consecutivos.

4. Si un subarray contiene un único valor, el crecimiento se define como 0.
5. Los crecimientos porcentuales calculados se almacenan en una lista, y el promedio de estos se utiliza para proyectar la venta futura aplicándolo al último valor del array.

Esta técnica asegura estimaciones fundamentadas en patrones históricos de ventas.

Opción 3: Simulación de Compra

En esta opción, el usuario accede a un menú que muestra:

- Todos los productos disponibles en la tienda.
- Precio por unidad de cada producto.
- Cantidad disponible en inventario.

El usuario puede añadir productos al carrito especificando las cantidades deseadas. Durante este proceso:

- Si el producto seleccionado no existe, el sistema notifica al usuario.
- Si la cantidad ingresada supera la disponibilidad, se informa adecuadamente.

Al finalizar la compra seleccionando, el sistema:

1. Muestra en pantalla un resumen de la compra, incluyendo el valor total y los productos adquiridos.
2. Genera un archivo llamado **"simulación_venta.csv"**, que almacena la información detallada de los productos, precio por unidad y cantidades compradas.

Opción 4: Generación de Combinaciones de Compra por Presupuesto

Mediante un algoritmo de **backtracking**, el sistema genera todas las combinaciones posibles de productos que el usuario podría adquirir respetando un presupuesto ingresado. El algoritmo opera de la siguiente manera:

1. Evalúa cada producto dentro del rango presupuestal.
2. Explora combinaciones posibles a partir de ese producto descendiendo en profundidad hasta completar la cantidad disponible o agotar el presupuesto.

3. Considera las restricciones de disponibilidad de productos e informa si alguna combinación no es viable.

Este enfoque permite al usuario optimizar sus compras con base en los recursos disponibles.

Opción 5: Análisis de Compras por Categoría

El usuario puede seleccionar una categoría de productos, tras lo cual el sistema muestra:

- Los nombres de los clientes que han adquirido productos de esa categoría.
- El cliente que ha comprado la mayor cantidad de productos dentro de la categoría seleccionada.

Esta funcionalidad utiliza un enfoque de **fuerza bruta** para:

1. Identificar productos pertenecientes a la categoría seleccionada.
2. Extraer los IDs de estos productos y cruzarlos con el historial de ventas.
3. Determinar al cliente con mayor cantidad de compras dentro de estas ventas.

Opción 6: Análisis Optimizado de Compras por Categoría

Similar a la opción 5, pero con una implementación más eficiente basada en **recursión**. Este enfoque mejora el rendimiento al reducir el número de iteraciones necesarias para encontrar el cliente con mayor cantidad de compras en una categoría seleccionada.

Opción 7: Visualización de Gráficas

El usuario puede generar gráficas interactivas para visualizar diferentes aspectos de la información:

1. **Gráfica de barras:** Muestra las ventas totales por categoría.
2. **Gráfica de barras:** Representa los productos más vendidos, ordenados de mayor a menor.
3. **Gráfica de líneas:** Presenta la cantidad de ventas distribuidas por meses y años.

Estas visualizaciones ofrecen una representación clara y comprensible de los datos, facilitando la toma de decisiones.

Opción 8:

Si el usuario elige la opción 8 saldrá del programa

Retos y Soluciones

Durante la implementación del proyecto, se identificaron varios desafíos:

- **Sintaxis de Python:** La amplia variedad de funciones y características de las librerías requeridas supuso una curva de aprendizaje considerable.
- **Implementación de algoritmos:** Fue necesario investigar y comprender en profundidad la naturaleza de los algoritmos utilizados, como divide y vencerás, backtracking y recursión, para garantizar su correcta aplicación en cada funcionalidad.

El aprendizaje obtenido durante este proceso fortaleció nuestras habilidades en desarrollo y optimización de sistemas.

Macroalgortimos:

Backtracking:

Pre: productos[][] (matriz con la informacion de los productos), carrito[] (donde se guardarán los nombres de los productos a añadir), int k (maneja profundidad del árbol, indica desde donde avanzar en el recorrido de los productos), float presupuesto, acumulado, precioMenor (precio del producto que menos cuesta)

Pos: recorre las posibles combinaciones con cada producto y verifica si es una solucion, una vez la encuentra imprime el carrito y se devuelve en el árbol para hallar más combinaciones

combinacionDeProductos(productos, carrito, k, presupuesto, acumulado, precioMenor)

// Condición de corte

if presupuesto - acumulado < precioMenor or k = Length(productos) then

Display "Carrito: ", carrito, ", Total: ", acumulado, " (Presupuesto: ", presupuesto, ")"

```
  else
    for i from k to Length(productos) - 1 do
      producto, precio, cantidadDisponible <- productos[i]

      // Verificar si el producto es viable
      if esViable1(precio, acumulado, presupuesto) and esViable2(producto,
cantidadDisponible, carrito) then
        // Añadir producto al carrito
        carrito.Append(producto)
        acumulado <- acumulado + precio

        // Llamar recursivamente
        combinacionDeProductos(productos, carrito, i, presupuesto, acumulado,
        precioMenor)

        // Restaurar estado después de la recursión
        carrito.Pop()
        acumulado <- acumulado – precio
      end if
      if not esViable2(producto, cantidadDisponible, carrito) then
        Display "Carrito: ", carrito, ", Total: ", acumulado, " (Presupuesto: ",
        presupuesto, ")"
      end if
    end for
  end if
End
```

Pre: float valorProducto, acumulado, presupuesto

Pos: bool que indica que el producto es viable para añadir al carrito (si al añadirlo no se pasa del presupuesto)

esViable1(valorProducto, acumulado, presupuesto)

return valorProducto + acumulado <= presupuesto

End

Pre: str producto (nombre), int cantidad, carrito[]

Pos: bool que indica que el producto es viable para añadir al carrito (si al añadirlo no se pasa de la cantidad disponible de ese producto)

esViable2(producto, cantidad, carrito)

return carrito.Count(producto) < cantidad

End

Divide y vencerás:

Pre: ventas[] (list que contiene la cantidad de ventas de un producto ordenadas por periodos de tiempo ascendentes), crecimientos[] (list donde se almacenarán los crecimientos porcentuales)

Pos: agrega a la lista crecimientos los crecimientos porcentuales de cada periodo respecto al siguiente, si llega a una sublista con un solo valor devuelve 0 porque no hubo crecimiento

proyeccion_crecimiento_divide_venceras(ventas, crecimientos)

// Caso base: si solo hay un punto, no se puede calcular el crecimiento porcentual

if Length(ventas) <= 1 then

crecimientos.Append(0)

return

// Caso base: si solo hay dos puntos, calcular el crecimiento porcentual entre ellos

elif Length(ventas) = 2 then

crecimientos.Append(crecimiento_porcentual(ventas[0], ventas[1]))

return

else

// Dividir el conjunto de datos en dos mitades

mitad <- Length(ventas) // 2

ventas_izquierda <- ventas[0:mitad]

```
    ventas_derecha <- ventas[mitad:Length(ventas)]
    // Calcular el crecimiento porcentual para cada mitad
    proyeccion_crecimiento_divide_venceras(ventas_izquierda, crecimientos)
    proyeccion_crecimiento_divide_venceras(ventas_derecha, crecimientos)
  end if
End
```

Pre: int menor, mayor (la cantidad de ventas de un periodo de tiempo y las del siguiente a este)

Pos: float crecimiento_porcentual (el crecimiento porcentual de la cantidad de ventas en un periodo de tiempo respecto al siguiente)

```
crecimiento_porcentual(menor, mayor)
  // Validar cual de los dos es el mayor y el menor
  if menor > mayor then
    aux <- menor
    mayor <- aux
    menor <- mayor
  // Retornar el crecimiento porcentual
  return ((mayor - menor) / menor) * 100
End
```

Pre: listaVentas[] (list que contiene las ventas ordenadas por periodo de tiempo)

Pos: calcular y mostrar la proyección de crecimiento de ventas y la proyección de la próxima venta

```
mostrarProyeccion(listaVentas)
  crecimientos <- []
  // Calcular los crecimientos porcentuales usando divide y vencerás
  proyeccion_crecimiento_divide_venceras(listaVentas, crecimientos)

  suma <- 0
  for i in crecimientos do
    suma <- suma + i
```

```
  end for

  // Promediar los crecimientos porcentuales
  crecimiento_promedio <- suma / Length(crecimientos)
  Display "Crecimiento porcentual promedio: ", round(crecimiento_promedio, 1)

  // Proyección de la próxima venta utilizando el crecimiento porcentual promedio
  ultima_venta <- listaVentas[-1]
  proxima_venta <- ultima_venta * (1 + crecimiento_promedio / 100)
  Display "Proyección para el próximo mes: ", round(proxima_venta, 0)
End
```

Fuerza Bruta:

Pre: listaClientes[][] (Contiene un listado con todos los clientes), lVentasActualizadas[][] (Contiene una lista con todas las ventas actualizadas), Productos[][] (Contiene una lista con todos los productos), int categoriaSeleccionada (index de la categoría seleccionada por el usuario)

Pos: Muestra el cliente que más compró de la categoría seleccionada

```
clientes_productos(listaClientes, listaVentasActualizadas, listaProductos, categoriaSeleccionada)
  categorias <- []
  for producto in listaProductos do
    if producto[2] not in categorias then
      categorias.Append(producto[2])
    end if
  end for

  Display "Clientes y productos de la categoría seleccionada:"
```

```
  if categoriaSeleccionada > Length(categorias) then
    Display "Categoría no encontrada"
  else
    clientesDeCategoria <- []
    for producto in listaProductos do
      // Encontrar la categoría en la lista de productos
      if producto[2] = categorias[categoriaSeleccionada - 1] then
        for venta in listaVentasActualizadas do
          // Encontrar el id del producto en la lista de ventas
          if venta[1] = producto[0] then
            for cliente in listaClientes do
              // Encontrar el id del cliente en la lista de ventas
              if cliente[0] = venta[4] then
                clientesDeCategoria.Append([cliente[1], cliente[2], venta[2],
producto[1]])

                Display cliente[1], cliente[2], "compró", venta[2], "unidades de",
producto[1]
              end if
            end for
          end if
        end for
    end for

    // Encontrar el cliente que más compró verificando la cantidad de unidades
compradas
    clienteFinal <- clientesDeCategoria[0]
    for cliente in clientesDeCategoria do
      if int(cliente[2]) > int(clienteFinal[2]) then
        // Guardar el cliente con la mayor cantidad de unidades compradas
        clienteFinal <- cliente
```

```
        end if
    end for
    Display clienteFinal[0], clienteFinal[1], "compró", clienteFinal[2], "unidades de", clienteFinal[3], "siendo el cliente que más compró"
  end if
End
```