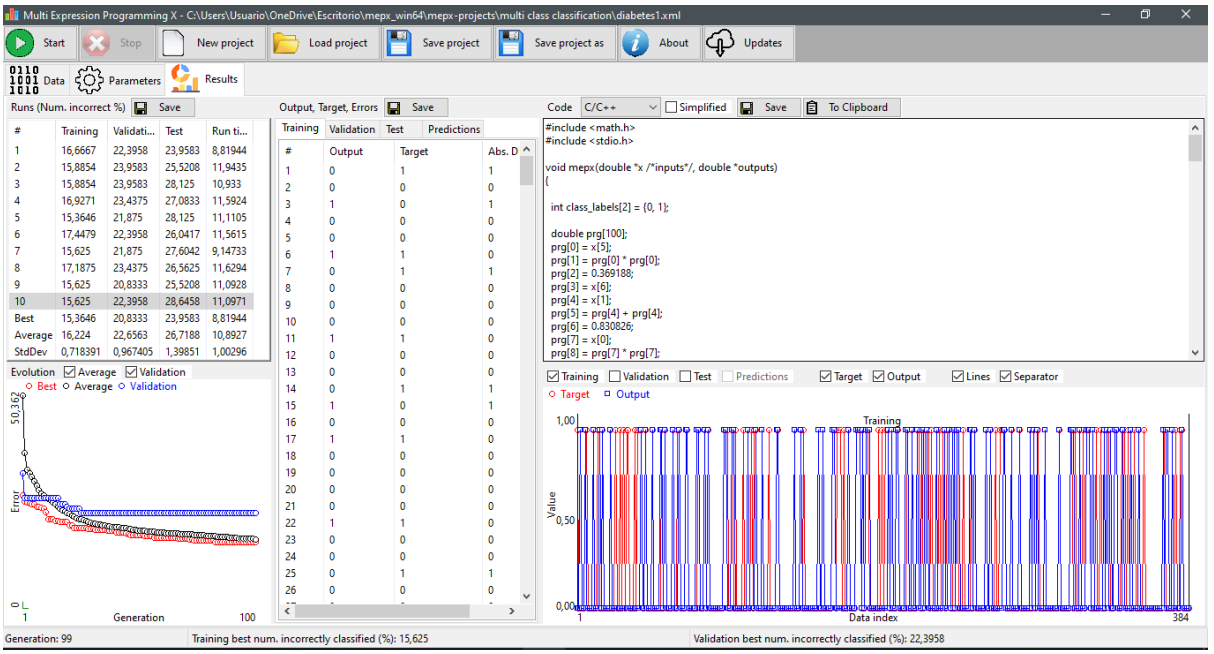
Actividad 4

Johana Tellez - Michael Caicedo

1. Descargue MEPX, https://www.mepx.org/, estúdielo y corra uno de los ejemplos que trae.
   MEPX (Multi Expression Programming, también llamado MEPX en el sitio) Funciona evolucionando poblaciones de “cromosomas” donde cada uno codifica múltiples soluciones posibles (no solo una), lo que permite explorar el espacio de posibles programas más eficientemente sin grandes costos extras.



   La captura corresponde a la ejecución de uno de los ejemplos que vienen con la descarga directa del MEPX, se titula “diabetes1.xml”. En este ejemplo se está resolviendo un problema de **clasificación de la gravedad de diabetes**: el programa toma variables de entrada y genera un modelo matemático que predice si el paciente pertenece a la clase “0” (sin diabetes) o “1” (con diabetes). En la parte izquierda se muestran los porcentajes de error en entrenamiento, validación y prueba a lo largo de varias corridas (siendo el mejor ≈ 15,36% de error en entrenamiento y ≈ 20,8% en validación). En el centro se listan los resultados comparando **salida (Output)** del modelo contra el **objetivo real (Target)** y la diferencia absoluta y abajo los gráficos muestran la evolución del error por generación y la comparación entre valores predichos y reales (rojo = objetivo, azul = salida). En resumen, el modelo aprendido busca aproximar patrones en los datos de diabetes y clasificar pacientes, aunque con un margen de error del ~20%

2. Suponga que tiene un robot que le entrega galletas al grupo de ingenieros de diseño de robots. Programe por PG el recorrido del robot, teniendo en cuenta que cada vez que un ingeniero recibe una galleta gana puntos. Los ingenieros están distribuidos en una sala cuadrada. Defina, conjunto de terminales, conjunto de funciones y función de aptitud.
   El conjunto de terminales está formado por las acciones básicas: mover_norte, mover_sur, mover_este, mover_oeste, entregar_galleta, buscar_ingeniero_cercano e ir_centro del robot; el conjunto de funciones incluye estructuras de control que

combinan o condicionan acciones: secuencia (ejecuta dos acciones), si_hay_galletas (if-then-else según galletas disponibles), repetir (ejecuta repetidamente una acción) y si_ingeniero_cerca (if-then-else según proximidad de ingenieros). Finalmente, la función de aptitud evalúa qué tan bueno es un programa considerando múltiples objetivos: número de ingenieros únicos atendidos (prioridad), galletas entregadas, cobertura de posiciones visitadas y penalización por distancia excesiva así como una mayor aptitud entre menos pasos se recorran, lo que guía la evolución hacia soluciones más eficientes.
Al archivo original se “RobotRepartidor.ipynb” se le agregó un gradiente de color a la trayectoria para evidenciar el recorrido paso a paso que realizó el robot.

3. Vea el video, https://www.youtube.com/watch?v=6KNuJn6dVy4. Analícelo y haga un ejemplo de control aplicando PG.

   El video explica cómo la **programación genética** se aplica al control de sistemas dinámicos no lineales, especialmente la **turbulencia en fluidos**. Se generan leyes de control en forma de árboles, que evolucionan hasta optimizar la mezcla o reducir el gasto energético. Este enfoque permite soluciones eficientes en transporte, energía y sistemas industriales complejos.

   Planteamiento del ejemplo

   Queremos usar Programación Genética (PG) para diseñar un controlador sobre un sistema dinámico sencillo:

   $$y(t+1) = y(t) + u(t)$$

   donde:

   - y(t) es el estado del sistema en el instante t.
   - u(t) es la acción de control.

   ## Objetivo de control:

   Que y(t) se acerque a un valor de referencia y_ref=10.

   ## PG aplicada al control:

   - **Cromosomas (árboles)**: expresiones que generan u(t) a partir de la variable y(t) y constantes.
   - **Terminales**: {y, 1, 2, 5, 10}
   - **Funciones**: {+,−,∗,/}
   - **Función de aptitud**:

$$f_{\text{apt}} = \frac{1}{1 + |y(T) - y_{ref}|}$$

donde y(T) es el valor del sistema tras simular T pasos. Cuanto más cerca esté de 10, mayor será la aptitud.

Explicación y análisis de resultados

Evolución:

- Se inicia con una población de árboles aleatorios.
- En cada generación se seleccionan los más aptos (los que dejan y(T) más cerca de 10).
- Se aplican cruces y mutaciones para generar nuevos controladores.

Ejemplo de salida (puede variar porque es aleatorio):

```
Generación 1: Mejor aptitud = 0.2000
Generación 20: Mejor aptitud = 0.4000
Generación 40: Mejor aptitud = 0.6667
Generación 80: Mejor aptitud = 0.8333
Generación 100: Mejor aptitud = 1.0000
```

Interpretación:

- El árbol `['+', 'y', 10]` representa la ley de control u(t)=y+10.
- Con esta ley, el sistema evoluciona hasta llegar exactamente a y=10, logrando la máxima aptitud posible.
- Dependiendo de la corrida, el algoritmo puede encontrar distintas leyes equivalentes que cumplen el objetivo.