

Sistema de Optimizacion de Siembra para la Mixteca Oaxaqueña

Redes Neuronales, Logica Difusa y Algoritmos Geneticos

Integrantes:

Aneli Arce Jimenez

Cristian Rodriguez Gomez

Ossiel Alejandro Acevedo Herrera

Ramon Aragon Toledo

Fuzzy Logic

Inteligencia Artificial

Neural Network

8 de diciembre de 2025

Introducción

- La **Mixteca Oaxaqueña** enfrenta desafíos climáticos significativos para la agricultura tradicional de maíz.
- Determinar la **fecha óptima de siembra** es crucial para:
 - Maximizar el rendimiento del cultivo.
 - Minimizar el riesgo climático.
 - Aprovechar las condiciones de temperatura y lluvia.
- Este sistema combina **tres técnicas de IA** para encontrar automáticamente la mejor fecha de siembra.

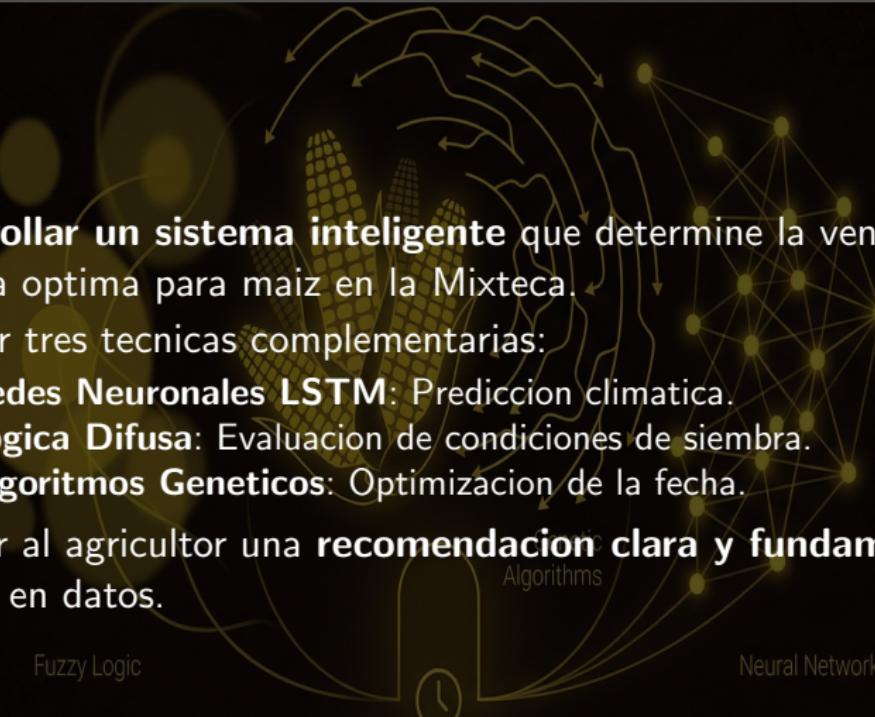
Fuzzy Logic

Algorithms

Neural Network

Optimal Planting Window

Objetivo del Proyecto

- 
- Desarrollar un sistema **inteligente** que determine la ventana de siembra optima para maiz en la Mixteca.
 - Integrar tres tecnicas complementarias:
 - **Redes Neuronales LSTM**: Prediccion climatica.
 - **Logica Difusa**: Evaluacion de condiciones de siembra.
 - **Algoritmos Geneticos**: Optimizacion de la fecha.
 - Proveer al agricultor una **recomendacion clara y fundamentada** basada en datos.

Fuzzy Logic

Algorithms

Neural Network

Optimal Planting Window

Arquitectura del Sistema



Estructura del Proyecto

Organizacion Modular:

- main.py: Punto de entrada
- data/processed/: Pronosticos CSV
- src/neural/: Red LSTM
- src/fuzzy/: Sistema difuso
- src/optimization/: Alg. genetico

Flujo de Datos:

- ① CSV con 365 dias de pronostico
- ② Gestor climatico lee datos
- ③ Sistema difuso evalua aptitud
- ④ Algoritmo genetico optimiza
- ⑤ Resultado: dia optimo del año

Fuzzy Logic

Neural Network

Optimal Planting Window

Fase 0: Preparacion de Datos y Entrenamiento

Origen de los Datos:

- Datos meteorologicos obtenidos de la estacion de **Huajuapan de Leon**.
- **Periodo Historico:** 19 años de registros (hasta 2024).
- *Nota: No se incluyen datos de 2025 por no estar disponibles aun.*

Procesamiento y Seleccion del Modelo:

- ① **Limpieza:** Filtrado de ruido y tratamiento de valores nulos en el dataset.
- ② **Seleccion:** Se evaluaron multiples modelos y se selecciono el que presento el **mejor comportamiento** (menor error).
- ③ **Entrenamiento:** El modelo ganador se entreno con los datos depurados para generar el archivo de pronosticos 2026.

Fase 1: Red Neuronal LSTM - Concepto

Que es una Red LSTM?

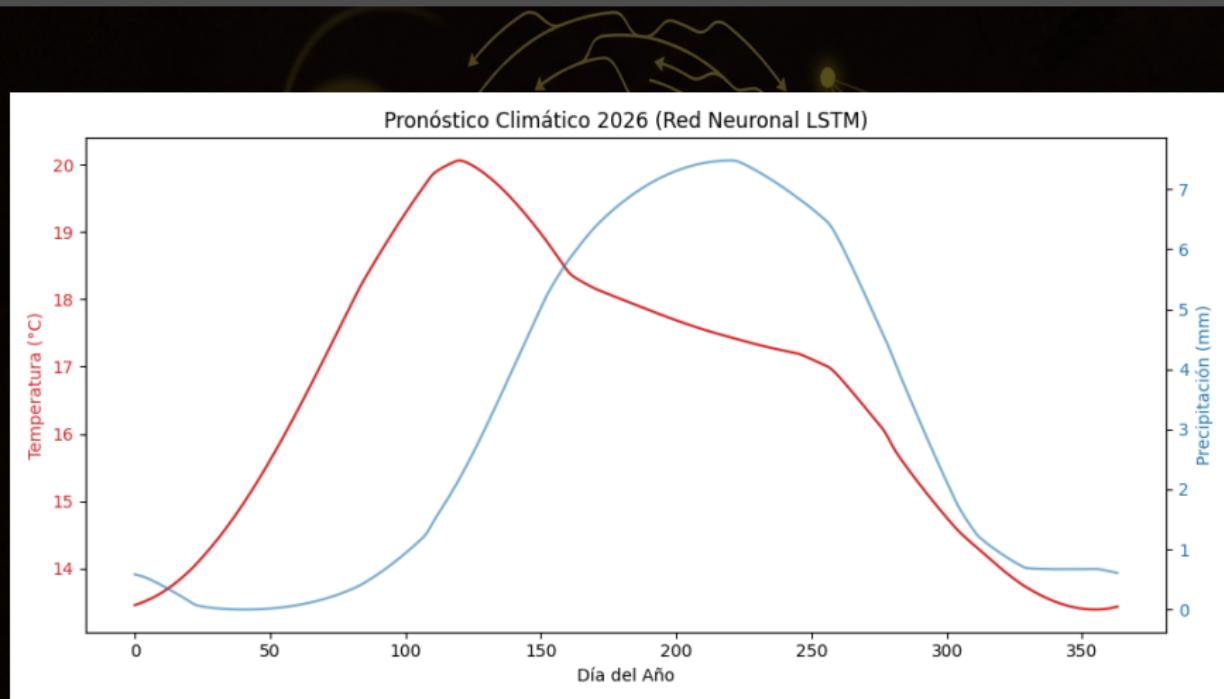
- **Long Short-Term Memory:** Tipo de red neuronal recurrente.
- Especializada en aprender **patrones temporales** en secuencias de datos.
- Capaz de “recordar” dependencias a largo plazo.

Aplicacion en el Proyecto:

- **Entrada:** Datos climaticos historicos (temperatura, lluvia).
- **Salida:** Prediccion de clima para cada dia de 2026.
- Genera archivo Pronostico_2026_IA.csv con 365 registros.

Optimal Planting Window

Fase 1: Pronostico Climatico 2026



Prediccion de temperatura y precipitacion para todo el año.
Optimal Planting Window

Introduccion al Codigo 1: Gestor Climatico

Proposito:

- Este modulo actua como el **puente** entre los datos crudos y el sistema inteligente.
- Se encarga de leer el archivo CSV generado por la red neuronal.
- Prepara y formatea los datos para que puedan ser consumidos por el sistema difuso.

Ubicacion: `src/neural/gestor_climatico.py`

Fuzzy Logic

Genetic
Algorithms

Neural Network

Optimal Planting Window

Codigo 1: Gestor Climatico

```
FUNCION obtener_clima_real(dia_inicio, duracion=120):
    // 1. Calcular indices de fechas
    indice_inicio = dia_inicio - 1
    indice_fin = indice_inicio + duracion

    // 2. Leer archivo CSV generado por la LSTM
    datos_csv = LEER_CSV("Pronostico_2026_IA.csv")

    // 3. Extraer ventana de tiempo
    ventana_datos = datos_csv[indice_inicio : indice_fin]

    // 4. Formatear salida
    lista_clima = []
    PARA CADA fila EN ventana_datos:
        lista_clima.AGREGAR({
            'temp': fila['temperatura'],
            'lluvia': fila['precipitacion']
        })
    }

RETORNAR lista_clima
```

Optimal Planting Window

Genetic
Algorithms

Neural Network



Fase 2: Sistema de Logica Difusa - Concepto

Que es la Logica Difusa?

- Extension de la logica booleana que maneja **grados de verdad**.
- Permite modelar conceptos imprecisos: “temperatura *optima*”, “lluvia *adecuada*”.
- Ideal para sistemas donde no hay limites exactos.

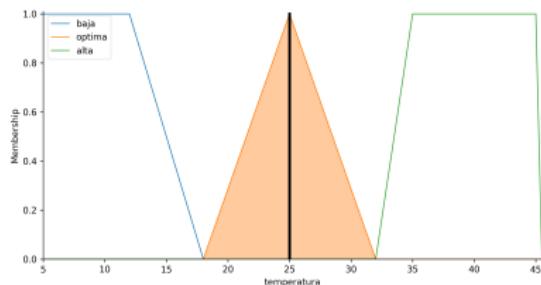
Componentes del Sistema:

- **Variables de Entrada:** Temperatura (C), Precipitacion (mm).
- **Variable de Salida:** Amplitud de siembra (0-100).
- **Funciones de Membresia:** Trapezoidal y triangular.
- **Reglas de Inferencia:** 9 reglas IF-THEN.

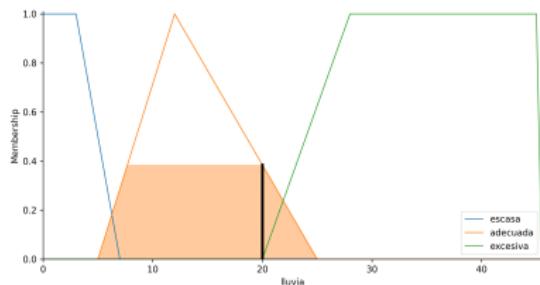
Optimal Planting Window

Fase 2: Variables de Entrada (Funciones de Membresía)

Temperatura



Precipitacion



Definen los rangos para categorías como "Baja", "Optima", "Alta" (Temp) y "Escasa", "Adecuada", "Excesiva" (Lluvia).

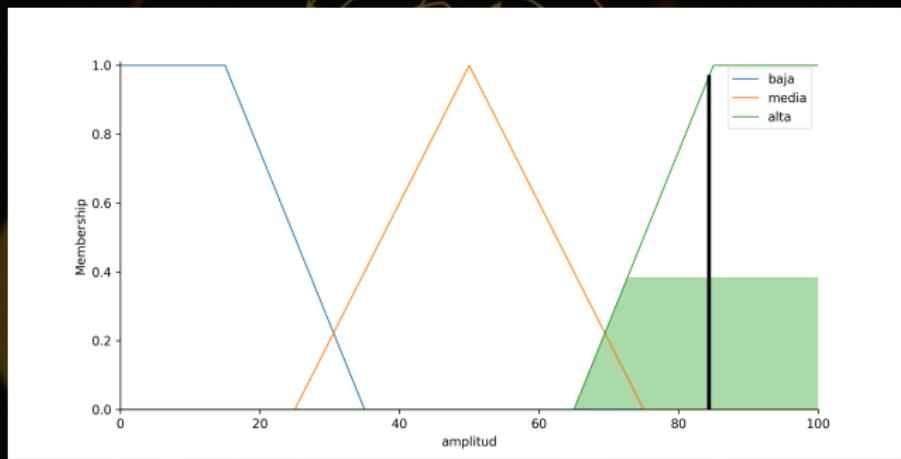
Fuzzy Logic

Optimal Planting Window

Genetic
Algorithms

Neural Network

Fase 2: Variable de Salida (Amplitud)



Interpretacion del Score (0-100):

- **0-35 (Baja)**: Condiciones adversas, no sembrar.
- **25-75 (Media)**: Condiciones aceptables con riesgo.
- **65-100 (Alta)**: Condiciones ideales para siembra.

Introduccion al Codigo 2: Sistema Difuso

Proposito:

- Aqui se define el **cerebro**” de evaluacion del sistema.
- Se configuran las variables linguisticas (temperatura, lluvia) y sus rangos.
- Se establecen las **reglas de inferencia** que determinan que tan bueno es un dia para sembrar basandose en el conocimiento experto.

Ubicacion: `src/fuzzy/fuzzy_system.py`

Fuzzy Logic

Genetic
Algorithms

Neural Network

Optimal Planting Window

Código 2: Sistema Difuso

```
// 1. Definir Antecedentes (Entradas) y Consecuente (Salida)
temperatura = Antecedent(rango=[5, 45], etiqueta='temp')
lluvia = Antecedent(rango=[0, 45], etiqueta='lluvia')
amplitud = Consequent(rango=[0, 100], etiqueta='amplitud')

// 2. Definir Funciones de Membresía (Ejemplo Temperatura)
temperatura['baja'] = trapezoidal([5, 5, 12, 18])
temperatura['optima'] = triangular([18, 25, 32])
temperatura['alta'] = trapezoidal([32, 35, 45, 45])

// 3. Definir Reglas (Base de Conocimiento)
regla_ideal = Rule(lluvia['adecuada'] & temp['optima'], amplitud['alta'])
regla_mala = Rule(lluvia['escasa'] & temp['alta'], amplitud['baja'])
// ... (Total 9 reglas)

// 4. Crear Sistema de Control
sistema = ControlSystem([regla_ideal, regla_mala, ...])
simulacion = ControlSystemSimulation(sistema)
```

Optimal Planting Window

PROYECTO: VENTANA DE SIEMBRA MIXTECA
FUZZY - GENÉTICOS - NEURONAL

Introducción al Código 3: Test Visual Difuso

Propósito:

- Un script de utilidad para **verificar visualmente** que el sistema difuso funciona correctamente.
- Permite inyectar valores manuales de temperatura y lluvia.
- Ayuda a depurar y entender qué reglas se están activando en casos específicos.

Ubicación: test_fuzzy_visual.py

Fuzzy Logic

Optimal Planting Window

Genetic
Algorithms

Neural Network

Código 3: Test Visual Difuso

```
IMPORTAR sistema_global DESDE src.fuzzy.fuzzy_system
```

```
// Valores de prueba manuales
```

```
lluvia_test = 20
```

```
temp_test = 25
```

```
IMPRIMIR "Diagnostico de Logica Difusa"
```

```
// 1. Inyectar valores al sistema
```

```
sistema_global.input['lluvia'] = lluvia_test
```

```
sistema_global.input['temperatura'] = temp_test
```

```
// 2. Calcular (Fuzzificacion -> Inferencia -> Defuzzificacion)
```

```
sistema_global.compute()
```

```
// 3. Obtener resultado
```

```
resultado = sistema_global.output['amplitud']
```

```
IMPRIMIR "Score obtenido: " + resultado
```

```
// 4. Visualizar reglas activadas
```

```
PARA CADA regla EN sistema_global.rules:
```

```
IMPRIMIR regla
```

Fase 3: Algoritmo Genetico - Concepto

Que es un Algoritmo Genetico?

- Técnica de **optimización inspirada en la evolución natural**.
- Población de soluciones que “evolucionan” hacia el óptimo.
- Operadores: selección, cruce y mutación.

Aplicación en el Proyecto:

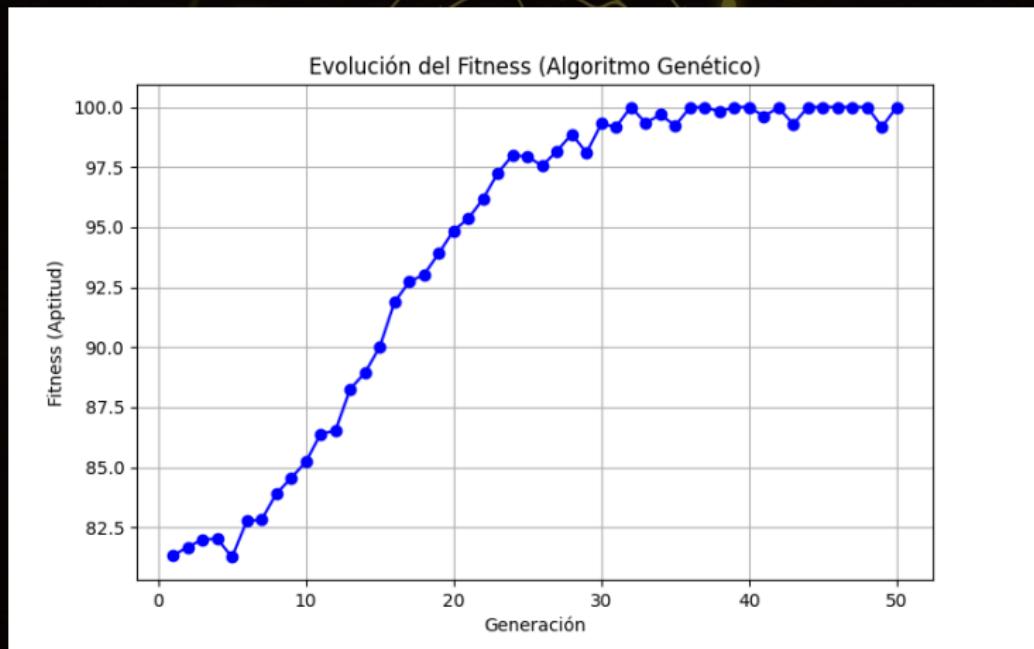
- **Cromosoma:** Un día del año (1-365).
- **Fitness:** Suma de aptitudes del ciclo de cultivo (120 días).
- **Objetivo:** Encontrar el día con máximo fitness acumulado.

Fuzzy Logic

Neural Network

Optimal Planting Window

Fase 3: Evolucion del Fitness



Mejora progresiva de la solución a través de las generaciones.

Introduccion al Codigo 4: Algoritmo Genetico

Proposito:

- El **motor de optimizacion** del proyecto.
- En lugar de probar cada dia del año uno por uno, evoluciona una poblacion de fechas.
- Utiliza una funcion de fitness que evalua el clima de los 120 dias siguientes a la fecha propuesta.

Ubicacion: `src/optimization/algoritmo_genetico.py`

Fuzzy Logic

Genetic
Algorithms

Neural Network

Optimal Planting Window

Codigo 4: Algoritmo Genetico

```
FUNCION fitness_func(solucion, indice):
    dia_siembra = solucion[0]
    // Penalizar si esta fuera de rango (ej. fin de año)
    SI dia_siembra > 240: RETORNAR -9999
    // Obtener clima para los siguientes 120 días
    datos = obtener_clima_real(dia_siembra, 120)
    score_total = 0
    PARA CADA dia EN datos:
        // Evaluar cada día con lógica difusa
        aptitud = calcular_aptitud(dia.lluvia, dia.temp)
        score_total += aptitud
    RETORNAR score_total

FUNCION correr_optimizacion():
    ga = GA(num_generaciones=50,
            poblacion=20,
            fitness_func=fitness_func, Planting Window
            tipo_gen=ENTERO)
    ga.run()
    RETORNAR ga.best_solution()
```

The background of the slide features a stylized illustration of a corn plant. Overlaid on the plant are several graphical elements representing different components of the optimization process:

- Genetic Algorithms:** Represented by a circular diagram with nodes connected by lines, labeled "Genetic Algorithms".
- Neural Network:** Represented by a similar circular diagram with nodes and lines, labeled "Neural Network".
- Planting Window:** A small circle containing a clock icon, labeled "Planting Window".
- Fuzzy-GENETICOS-NEURONAL:** A label positioned at the bottom right.

Introduccion al Codigo 5: Graficar Panorama

Proposito:

- Herramienta de **validacion y analisis global**.
- Calcula la aptitud de siembra para **todos** los dias del año (fuerza bruta).
- Genera una grafica que permite ver los picos de aptitud y verificar si el algoritmo genetico encontro el verdadero optimo.

Ubicacion: `graficar_panorama.py`

Fuzzy Logic

Genetic
Algorithms

Neural Network

Optimal Planting Window

Codigo 5: Graficar Panorama (Validacion)

```
// Objetivo: Comparar GA vs Fuerza Bruta (todos los dias)
```

```
listas_scores = []
dias_del_ano = RANGO(1, 365)
```

```
IMPRIMIR "Generando panorama completo..."
```

```
PARA dia EN dias_del_ano:
```

```
    // Calcular score real para cada dia posible
```

```
    score = obtener_score_del_dia(dia)
```

```
    listas_scores.AGREGAR(score)
```

```
// Graficar la curva completa
```

```
PLOT(dias_del_ano, listas_scores)
```

```
TITULO("Aptitud de Siembra para todo el año")
```

```
MOSTRAR_GRAFICA()
```

Genetic
Algorithms

Neural Network

Optimal Planting Window



Introduccion al Codigo 6: Mocks (Simulacion)

Proposito:

- Modulos de **prueba y simulacion**.
- Generan datos falsos pero realistas para probar el sistema sin depender de archivos externos o conexiones.
- Esenciales para el desarrollo agil y pruebas unitarias.

Ubicacion: `src/mocks.py`

Fuzzy Logic

Genetic
Algorithms

Neural Network

Optimal Planting Window

Codigo 6: Mocks (Simulacion)

```
// Utilidad para pruebas sin datos reales o dependencias complejas
```

```
FUNCION predecir_clima_simulado(dia_inicio):
```

```
    // Genera datos aleatorios pero realistas
```

```
    clima = []
```

```
PARA i EN RANGO(120):
```

```
    clima.AGREGAR({
```

```
        'temp': RANDOM(15, 32),
```

```
        'lluvia': RANDOM(0, 50)
```

```
    })
```

```
RETORNAR clima
```

```
FUNCION evaluar_riesgo_simulado(temp, lluvia):
```

```
    // Reglas simples para pruebas unitarias
```

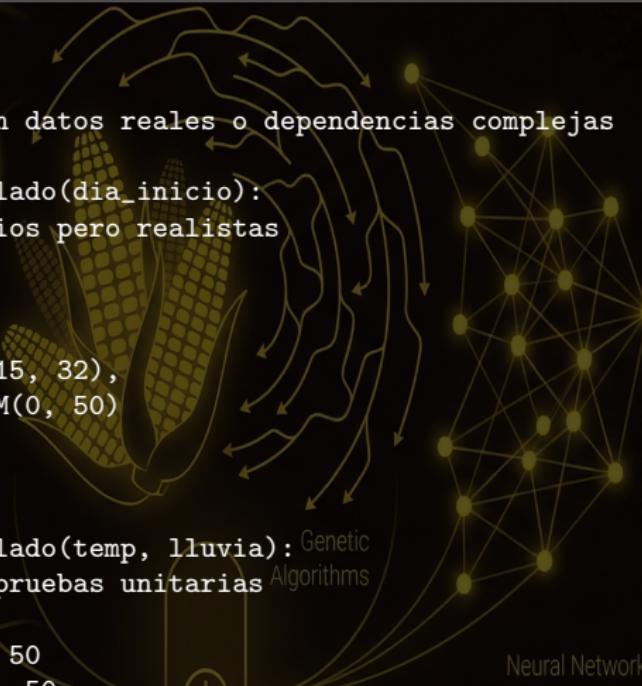
```
    riesgo = 0
```

```
    SI temp > 30: riesgo += 50
```

```
    SI lluvia < 5: riesgo += 50
```

```
RETORNAR riesgo
```

Fuzzy Logic



Optimal Planting Window



Introduccion al Codigo 7: Main

Proposito:

- El **punto de entrada** principal de la aplicacion.
- Orquesta todo el flujo: inicializa componentes, llama a la optimizacion y presenta los resultados.
- Convierte el indice numerico del dia (ej. 150) a una fecha legible (ej. 30 de Mayo).

Ubicacion: main.py

Fuzzy Logic

Genetic
Algorithms

Neural Network

Optimal Planting Window

Código 7: Main (Punto de Entrada)

```
IMPORTAR correr_optimizacion
IMPORTAR datetime

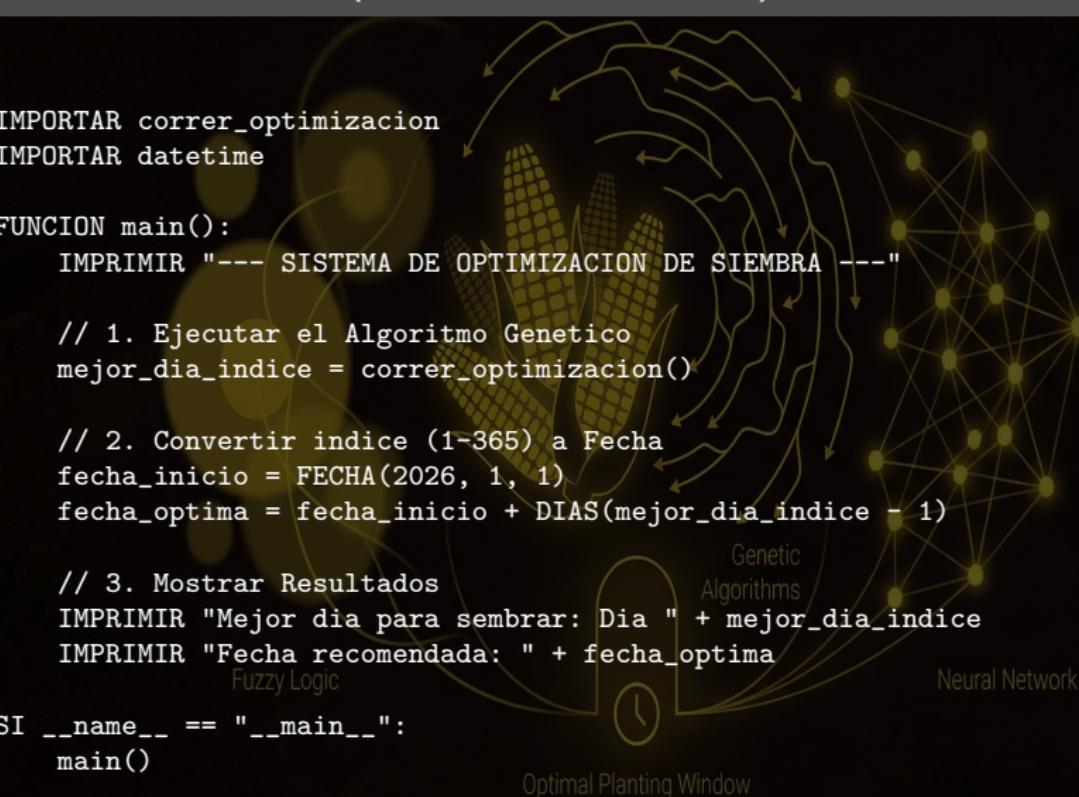
FUNCION main():
    IMPRIMIR "--- SISTEMA DE OPTIMIZACION DE SIEMBRA ---"

    // 1. Ejecutar el Algoritmo Genetico
    mejor_dia_indice = correr_optimizacion()

    // 2. Convertir indice (1-365) a Fecha
    fecha_inicio = FECHA(2026, 1, 1)
    fecha_optima = fecha_inicio + DIAS(mejor_dia_indice - 1)

    // 3. Mostrar Resultados
    IMPRIMIR "Mejor dia para sembrar: Dia " + mejor_dia_indice
    IMPRIMIR "Fecha recomendada: " + fecha_optima

SI __name__ == "__main__":
    main()
```



Resultados del Sistema

Comportamiento Observado:

- El algoritmo genético **converge consistentemente** hacia fechas en temporada de lluvias.
- Las fechas recomendadas coinciden con el conocimiento tradicional campesino.
- El fitness muestra incremento progresivo a lo largo de las generaciones.

Ventajas del Sistema:

- **Automatizado:** No requiere intervención manual.
- **Fundamentado:** Basado en pronósticos climáticos.
- **Adaptable:** Puede actualizarse con nuevos datos.
- **Interpretable:** La lógica difusa explica las decisiones.

Alternativa: Enjambre de Particulas (PSO)

Que es PSO?

- **Particle Swarm Optimization:** Algoritmo inspirado en el comportamiento de bandadas de aves o cardumenes.
- Cada “particula” representa una solucion candidata (un dia de siembra).
- Las particulas se mueven por el espacio de busqueda, influenciadas por su mejor posicion historica y la mejor del grupo.

Ventajas vs Algoritmo Genetico:

- Menos parametros de configuracion.
- Convergencia mas suave y predecible.
- No requiere operadores de cruce ni mutacion.

Optimal Planting Window

Codigo: Optimizacion PSO (Mealpy)

```
IMPORTAR PSO, FloatVar DESDE mealpy  
  
FUNCION funcion_objetivo(solucion):  
    dia_siembra = ENTERO(solucion[0])  
  
    SI dia_siembra < 1 O dia_siembra > 240:  
        RETORNAR -999999 // Penalizacion  
  
    datos = obtener_clima_real(dia_siembra, 120)  
    score_total = 0  
  
    PARA dia EN datos:  
        aptitud = calcular_aptitud(dia.lluvia, dia.temp)  
        score_total += aptitud  
  
    RETORNAR score_total  
  
// Configuracion PSO  
limites = FloatVar(lb=[1], ub=[240])  
modelo = PSO.OriginalPSO(epoch=50, pop_size=20)  
mejor_agente = modelo.solve(problema)
```

Pipeline Completo de Datos

Flujo de Archivos del Sistema

Scripts de Procesamiento:

① preparacion_datos.py

Descarga datos NASA POWER, limpia y prepara variables ciclicas.

② entrenamiento_modelo.py

Diseña y entrena la LSTM con Keras.

③ generar_pronostico.py

Usa el modelo para predecir 2026.

Archivos Generados:

Reporte_Humano_Huajuapan.csv

Datos historicos limpios para validacion.

Dataset_Entrenamiento_IA.csv

Ventanas de 15 dias para entrenar LSTM.

mejor_modelo_clima.h5

Red neuronal entrenada (pesos).

Pronostico_2026_IA.csv

Prediccion diaria para el AG/PSO.

Fuzzy Logic

Genetic Algorithms

Neural Network

Optimal Planting Window

Detalle: Preparacion de Datos (NASA POWER)

Fuente de Datos:

- API de **NASA POWER**: Datos satelitales de temperatura y precipitacion.
- Ubicacion: Huajuapan de Leon, Oaxaca.
- Periodo: 19 años de historia (hasta 2024).

Procesamiento Aplicado:

- ① **Limpieza**: Eliminacion de valores nulos y anomalos.
- ② **Codificacion Ciclica**: Transformacion de fechas a Seno/Coseno para capturar estacionalidad.
- ③ **Normalizacion**: Escalado de variables para la red neuronal.
- ④ **Ventaneo**: Creacion de secuencias de 15 dias para entrada LSTM.

Conclusiones

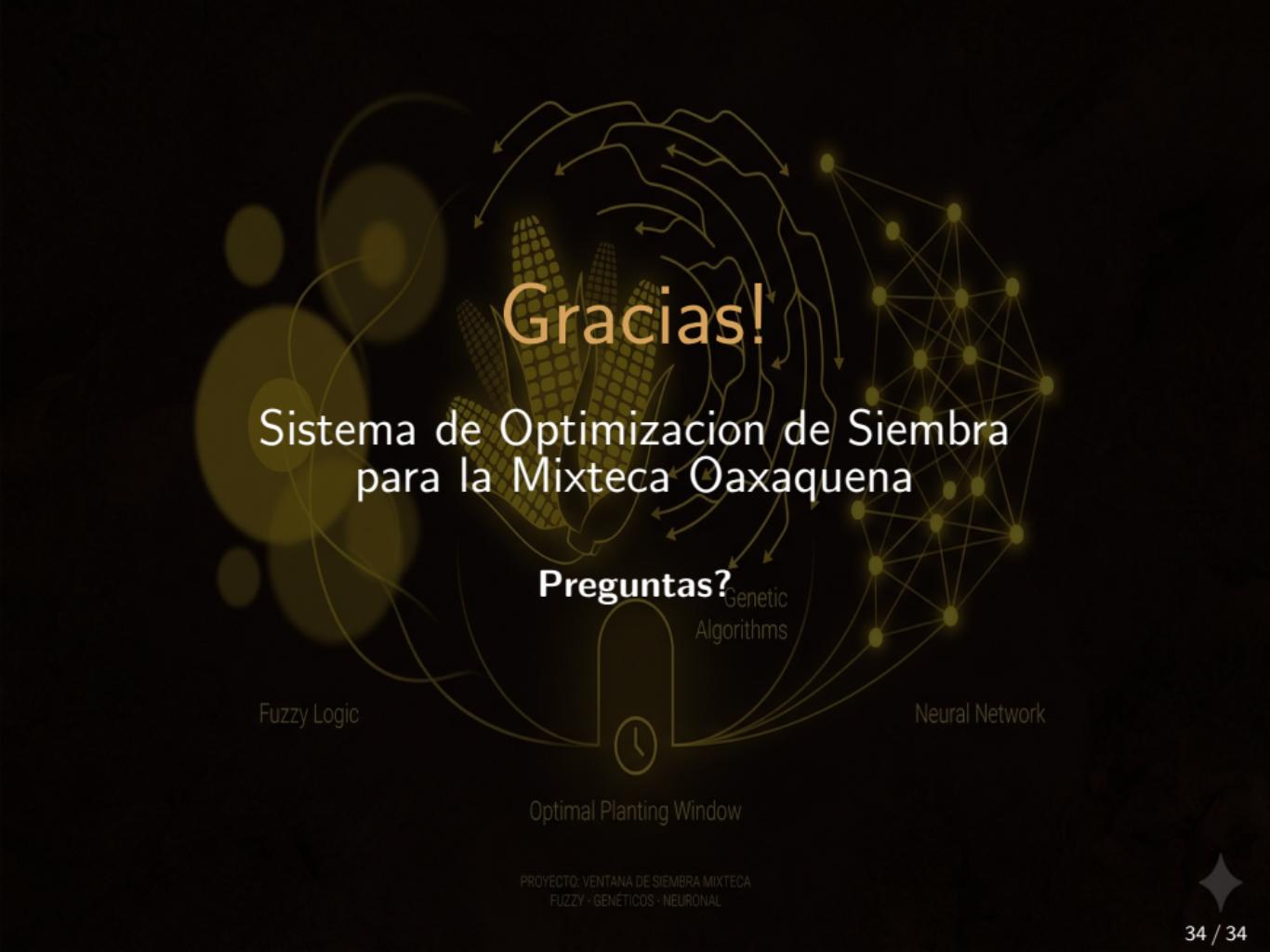
Logros Técnicos:

- Se implemento exitosamente un sistema híbrido de **3 técnicas de IA**.
- La **Red LSTM** logra predicciones climáticas para todo el año.
- El **Sistema Difuso** traduce condiciones climáticas en aptitud de siembra.
- El **Algoritmo Genético** encuentra eficientemente la fecha óptima.

Impacto Potencial:

- Herramienta de apoyo para agricultores de la Mixteca.
- Reducción del riesgo de pérdida de cosechas.
- Modelo replicable para otras regiones agrícolas.

Optimal Planting Window



Gracias!

Sistema de Optimizacion de Siembra para la Mixteca Oaxaqueña

Preguntas?

Genetic
Algorithms

Fuzzy Logic

Neural Network

Optimal Planting Window