

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL ROSARIO

Materia: Algoritmos Genéticos

Comisión: 3EK03

MACHINE LEARNING Y SU VINCULACIÓN CON AG

Proyecto de Investigación

Ciclo Lectivo: 2025

Integrantes del grupo:

Nombre y Apellido	Legajo
Juan Cruz Mondino	51922
Alexis Mateo	51191
Gustavo Giampietro	50671

Profesores:

Daniela Díaz
Víctor Lombardo

Índice

1 Introducción	3
2 Situación Problemática	3
3 Problema	4
4 Objetivos de la Investigación	4
4.1 Objetivo General	4
4.2 Objetivos Específicos	4
5 Marco Teórico	5
5.1 Glosario	5
5.2 Fundamentos Teóricos	5
5.2.1 Algoritmos Genéticos	5
5.2.2 Redes Neuronales Artificiales	6
5.2.3 LSTM	8
5.2.4 GBR	10
5.2.5 Árboles de Decisión	12
6 Metodología	12
6.1 Selección del Área de Cultivo	12
6.2 Recuperación y Procesamiento de Datos Climáticos	13
6.3 Preparación de los Datos de Entrenamiento	13
6.4 Ejecución del Algoritmo Genético	13
6.5 Visualización de Resultados	14
7 Algoritmo Desarrollado	14
8 Resultados	15
8.1 Modelos Climáticos (LSTM/GBR)	15
8.2 Modelo de rendimiento	17
8.3 Optimización con AG	18
9 Conclusiones	19

10 Repositorio del Proyecto

19

11 Referencias

19

1. Introducción

En los últimos años, la evolución de la capacidad de cómputo permitió la modelación de diferentes escenarios del mundo físico. En este trabajo se busca aprovechar dicha evolución con el fin de encontrar la solución óptima al evaluar las diferentes características en la planificación agrícola en el contexto de la Argentina.

En este marco, los algoritmos genéticos, por su capacidad para abordar problemas de optimización con múltiples variables y restricciones, permiten explorar configuraciones eficientes en sistemas agrícolas diversificados. Por su parte, con algoritmos de machine learning se pueden obtener modelos predictivos entrenados sobre datos históricos, útiles para anticipar el comportamiento de los sistemas bajo distintas situaciones posibles en la agricultura.

La articulación entre ambos enfoques, optimización evolutiva y predicción basada en datos, permite avanzar hacia modelos de planificación agrícola más adaptables y contextualizados. Este trabajo explora dicha integración, con el objetivo de contribuir al desarrollo de soluciones computacionales que apoyen la toma de decisiones en unidades productivas reales, contemplando tanto la eficiencia como la sostenibilidad del sistema.

Cabe señalar que, en el presente trabajo, se ha decidido acotar deliberadamente el alcance del modelo propuesto. En particular, se omiten variables relacionadas con la incidencia de enfermedades, la presencia de plagas y los niveles de contaminación ambiental o química. Esta decisión metodológica responde a la necesidad de abordar inicialmente la problemática desde una perspectiva simplificada, que permita focalizar el análisis en los aspectos estructurales de la planificación agrícola, tales como la selección de cultivos, la distribución espacial y temporal, y la consideración de factores geográficos y climáticos.

2. Situación Problemática

En el contexto actual de la agricultura argentina, la planificación eficiente de la producción se ha vuelto una tarea cada vez más compleja. Esta planificación no solo abarca la selección de cultivos, sino también su disposición espacial dentro de las parcelas. Las decisiones en torno a estos aspectos deben considerar simultáneamente múltiples factores: las características del suelo, los cambios de clima, la disponibilidad de agua, las condiciones económicas del entorno productivo.

Pese a esta complejidad, en muchos casos las decisiones agronómicas aún se toman basadas en la experiencia previa del productor o en recomendaciones técnicas generales que no logran captar las particularidades de cada parcela ni adaptarse dinámicamente a condiciones cambiantes. Esto puede derivar en una subutilización del potencial productivo, un manejo ineficiente de los recursos naturales (especialmente agua y nutrientes del suelo) y una pérdida de sustentabilidad del sistema agropecuario a largo plazo.

3. Problema

¿Cómo aplicar modelos de machine learning combinados con algoritmos genéticos para optimizar la planificación agrícola, considerando múltiples variables interdependientes como clima, suelo y siembra de diferentes semillas en simultáneo?

4. Objetivos de la Investigación

4.1. Objetivo General

Desarrollar un modelo de optimización para la planificación espacial de cultivos en parcelas agrícolas, utilizando algoritmos genéticos y técnicas de machine learning.

4.2. Objetivos Específicos

- Obtener y estructurar datos agronómicos relevantes, tales como tipo de cultivo, características del suelo, datos climáticos e historial de producción, para entrenar el modelo de machine learning, mediante la recolección de bases de datos especializadas, APIs públicas y recursos en línea.
- Diseñar e implementar un modelo predictivo basado en machine learning que estime el rendimiento esperado de los cultivos bajo distintas combinaciones de condiciones climáticas y geográficas, de disposiciones en parcelas de cultivos y selecciones de semillas.
- Desarrollar los algoritmos necesarios para la recuperación, limpieza y procesamiento automatizado de datos desde las fuentes recolectadas, garantizando su calidad y consistencia para el análisis.
- Construir un algoritmo genético que utilice como función fitness los resultados del modelo predictivo, con el fin de encontrar combinaciones óptimas de distribución de cultivos para maximizar el rendimiento agrícola.
- Implementar un sistema integrado que combine el modelo predictivo, el algoritmo genético y las herramientas de visualización, incluyendo un mapa interactivo capaz de identificar departamentos y áreas de cultivo a partir de coordenadas geográficas específicas.
- Testear el modelo de optimización propuesto mediante simulaciones, evaluando su rendimiento, eficiencia y sostenibilidad agrícola.
- Generar una herramienta de apoyo a la toma de decisiones que permita a productores y técnicos agrícolas visualizar mediante gráficos interactivos, comparativas de rendimiento y recomendaciones de optimización para la planificación de parcelas.

5. Marco Teórico

5.1. Glosario

Funciones sigmoides: es una función matemática que transforma cualquier valor de entrada en un rango entre 0 y 1, produciendo una gráfica con forma de "S" característica. Se usa comúnmente en redes neuronales artificiales como función de activación, para introducir no linealidad y convertir los valores de salida en probabilidades, así como en algoritmos de machine learning para tareas como la regresión logística y la clasificación binaria.

Tangente hiperbólica: función matemática utilizada como función de activación en redes neuronales, incluyendo las LSTM. Convierte cualquier valor real en un número entre -1 y 1, lo que permite normalizar la información y modelar relaciones no lineales. Su principal ventaja es que, a diferencia de la función sigmoide que solo devuelve valores positivos, la tangente hiperbólica produce salidas centradas en cero, lo que facilita la propagación de gradientes y contribuye a que las celdas de memoria de la LSTM puedan representar incrementos o decrementos en la información de manera más equilibrada.

Naturaleza estocástica: La naturaleza estocástica de un algoritmo se refiere a que su comportamiento incluye elementos de aleatoriedad o probabilidad, de modo que no siempre produce el mismo resultado dado el mismo conjunto de entradas. En el caso de los algoritmos genéticos, esta aleatoriedad se manifiesta en la generación inicial de la población, en la selección de individuos para cruzamiento y en las mutaciones aplicadas, lo que permite explorar de manera más amplia el espacio de soluciones y evita que el algoritmo quede atrapado en óptimos locales.

Gradiente: se refiere al vector de derivadas parciales de una función de pérdida respecto a sus variables o parámetros. En aprendizaje automático y en Gradient Boosting, el gradiente indica la dirección y magnitud del cambio necesario en los parámetros para reducir el error. Por ejemplo, en Gradient Boosting Regression, los modelos sucesivos se entrena para predecir los gradientes (residuos) de la función de pérdida, corrigiendo así los errores de las predicciones anteriores y acercándose progresivamente a una solución óptima.

5.2. Fundamentos Teóricos

Los algoritmos genéticos serán utilizados para explorar combinaciones óptimas de cultivos, ubicaciones, tiempo de siembre y cosecha, evaluando cada una con una función fitness que puede, por ejemplo, maximizar el rendimiento esperado y mejorar la rotación, entre otras cosas.

Se aplicarán algoritmos de machine learning para predecir el rendimiento esperado dado un conjunto de condiciones iniciales basándose en los datos obtenidos de experiencias previas. Este modelo se integraría como parte de la función fitness del algoritmo genético. En nuestro caso, los algoritmos de Machine Learning utilizados son redes neuronales Long Short-Term Memory (LSTM) y Gradient Boosting Regression (GBM).

5.2.1. Algoritmos Genéticos

Los algoritmos genéticos (AG) son una técnica de optimización y búsqueda inspirada en los principios de la evolución natural y la genética. Introducidos por John Holland en la década de 1970, los AG forman parte del paradigma de la computación evolutiva, que busca resolver problemas

complejos explorando iterativamente un espacio de soluciones mediante mecanismos inspirados en la selección natural, la recombinación y la mutación.

En esencia, un algoritmo genético trabaja sobre una población de individuos, donde cada individuo representa una solución candidata al problema a resolver. Esta representación generalmente se codifica mediante estructuras discretas, como cadenas binarias, enteros o vectores de parámetros, denominadas cromosomas. La calidad de cada individuo se evalúa mediante una función de aptitud o fitness, que cuantifica qué tan buena es la solución con respecto al objetivo planteado.

El ciclo básico de un algoritmo genético consiste en los siguientes pasos:

1. Inicialización: se genera una población inicial de manera aleatoria o basada en heurísticas, asegurando diversidad suficiente para explorar el espacio de soluciones.
2. Evaluación: cada individuo se evalúa mediante la función de aptitud, asignándole un valor que refleja su desempeño relativo.
3. Selección: se eligen los individuos que participarán en la generación siguiente. Los métodos más comunes incluyen la ruleta y el torneo, todos orientados a favorecer la supervivencia de los individuos más aptos mientras se mantiene diversidad genética.
4. Reproducción (crossover o recombinación): pares de individuos seleccionados combinan sus cromosomas para generar descendencia. Esta operación permite intercambiar información genética entre soluciones y explorar nuevas regiones del espacio de búsqueda.
5. Mutación: se aplican cambios aleatorios a algunos genes de la descendencia, introduciendo variabilidad y exploración adicional que ayuda a evitar óptimos locales.
6. Reemplazo: la nueva generación sustituye total o parcialmente a la anterior, y el ciclo se repite hasta que se cumple un criterio de convergencia, como un número máximo de generaciones o una solución que alcanza un valor de aptitud satisfactorio.

Una característica clave de los AG es su capacidad de explorar espacios de soluciones altamente complejos y no lineales sin requerir derivadas ni información explícita de la función objetivo. Esto los hace especialmente útiles en problemas donde las relaciones entre variables son inciertas, discontinuas o multidimensionales, como la optimización de rutas, el diseño de sistemas complejos, la selección de características en aprendizaje automático o la planificación agrícola.

Además, los algoritmos genéticos permiten incorporar estrategias como elitismo, donde los individuos más aptos se preservan de generación en generación, aumentando la probabilidad de conservar soluciones de alta calidad. Esta flexibilidad, combinada con su naturaleza estocástica, convierte a los AG en una herramienta poderosa para la resolución de problemas de optimización global en contextos donde métodos tradicionales basados en gradientes o búsqueda exhaustiva serían inviables.

5.2.2. Redes Neuronales Artificiales

Las redes neuronales artificiales se estructuran en capas compuestas por neuronas. Cada neurona se conecta con otras mediante pesos sinápticos, los cuales representan la importancia relativa de cada conexión en el proceso de transmisión de información. Excepto en la capa de entrada, cada neurona posee un sesgo (bias), que es un valor numérico adicional que permite desplazar la función de activación y, por ende, mejorar la capacidad de ajuste del modelo.

El funcionamiento se desarrolla de manera progresiva: el valor de entrada se multiplica por los pesos correspondientes, se le suma el sesgo y este resultado se transmite hacia la siguiente capa. Este proceso se repite sucesivamente hasta llegar a la capa de salida. El valor obtenido en la salida se compara con la referencia esperada (valor real), y en función de la diferencia se realizan ajustes: si el error es considerable, las modificaciones en los parámetros serán mayores; en cambio, si el error es reducido, los ajustes serán mínimos.

Un caso particular de arquitectura es la capa densa (fully connected layer), en la cual todas las neuronas de una capa se conectan con todas las neuronas de la capa siguiente.

El proceso de actualización de pesos y sesgos se lleva a cabo mediante un optimizador (optimizer). La magnitud del paso de ajuste está determinada por la tasa de aprendizaje (learning rate): si esta es demasiado pequeña, los cambios serán muy lentos; si es excesivamente grande, se perderá precisión y estabilidad en la convergencia.

La función de pérdida empleada frecuentemente en problemas de regresión es el error cuadrático medio (MSE, Mean Square Error), que penaliza más fuertemente los errores de gran magnitud respecto a un gran número de errores pequeños.

$$MSN = \frac{1}{n} \sum_{i=1}^n (y_i - y'_i)^2$$

Donde:

- La cantidad de observaciones es representada con n
- El valor real observado es y_i
- El valor predicho por el modelo es y'_i

Finalmente, el entrenamiento de una red neuronal se desarrolla a lo largo de múltiples épocas (epochs), entendidas como la cantidad de iteraciones completas sobre el conjunto de datos, durante las cuales el modelo busca minimizar la función de pérdida y aproximarse al resultado esperado.

Estructura por Capas de una Red Neuronal

Una red neuronal, incluyendo las redes LSTM, está compuesta por tres tipos de capas principales: una capa de entrada, una o más capas ocultas y una capa de salida. Para incorporar estas capas en una red, se sigue el proceso lógico desarrollado a continuación.

1 - Capa de Entrada

Esta es la primera capa de la red y su función es procesar los datos de entrada. En una red LSTM, la capa de entrada recibe la secuencia de datos (por ejemplo, palabras en una oración o valores en una serie temporal) en cada paso de tiempo. Cada elemento de la secuencia se representa como un vector. Esta capa no realiza cálculos complejos, sino que su propósito principal es recibir y formatear los datos para las siguientes capas. [1]

2 - Capa Oculta

La estructura de una capa oculta en una red neuronal consiste en un conjunto de neuronas interconectadas que procesan la información de manera jerárquica. Cada neurona recibe señales de las neuronas de la capa anterior, combina estas señales mediante pesos y un término de sesgo, y las transforma utilizando una función de activación no lineal. Las conexiones entre neuronas se ajustan durante el entrenamiento para optimizar el rendimiento del modelo. La salida de la capa oculta se

transmite a la siguiente capa, formando un flujo de datos que permite a la red aprender patrones complejos. Esta estructura es clave para extraer características relevantes y representar las relaciones en los datos.

3 - Capa de Salida

La capa de salida es la capa final de la red. Su función es tomar la salida de la última capa oculta y transformarla en un formato que sea útil para la tarea que estás tratando de resolver.

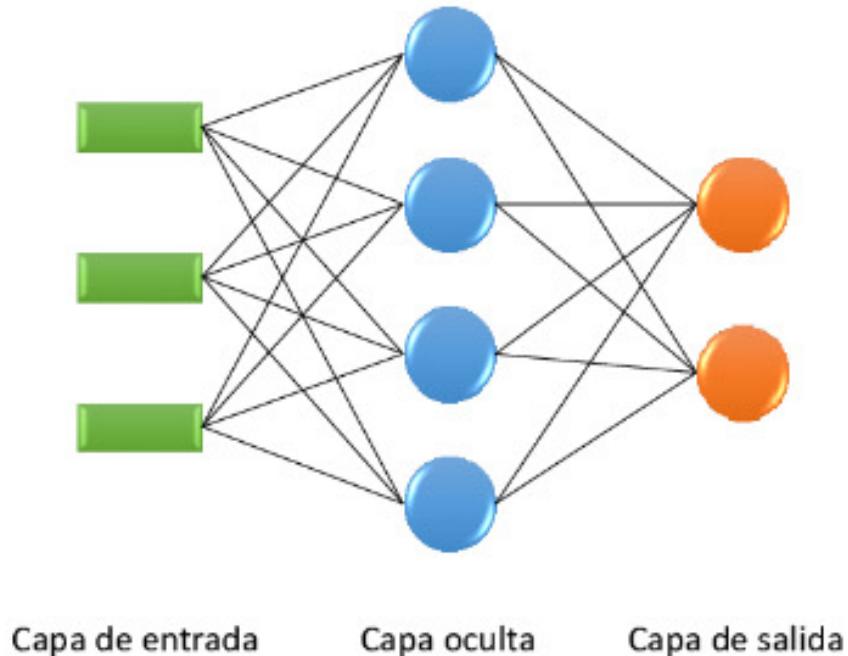


Figura 1: Representación gráfica las capas de una red neuronal

5.2.3. LSTM

Las redes neuronales recurrentes (RNN, por sus siglas en inglés) constituyen una clase de modelos diseñados específicamente para trabajar con datos secuenciales, como series temporales, lenguaje natural o señales biológicas. Sin embargo, las RNN tradicionales presentan limitaciones importantes al intentar capturar dependencias de largo plazo debido al problema del desvanecimiento o explosión del gradiente (Gradient Boosting) durante el entrenamiento. Para superar estas dificultades, Hochreiter y Schmidhuber (1997) propusieron la arquitectura Long Short-Term Memory (LSTM), que ha demostrado ser especialmente efectiva en el modelado de secuencias complejas.

Las redes LSTM incorporan una estructura de celdas de memoria que permiten almacenar y transmitir información relevante a lo largo de la secuencia, regulando explícitamente qué información se conserva y cuál se descarta. Esto se logra mediante compuertas que se implementan como funciones sigmoides y tangentes hiperbólicas que actúan sobre los vectores de entrada y el estado oculto, permitiendo una modulación adaptativa del flujo de información.

La principal ventaja de las LSTM es su capacidad de capturar dependencias a largo plazo en series temporales, sin que el gradiente se desvanezca de forma crítica. Esto las convierte en modelos muy utilizados en tareas como:

- Predicción de series temporales (ej. variables climáticas, financieras, agrícolas).
- Traducción automática y procesamiento de lenguaje natural.
- Reconocimiento de voz y señales biomédicas.

Estructura de una Red LSTM

Una LSTM (Long Short-Term Memory) extiende la RNN básica añadiendo una celda de estado que actúa como una cinta transportadora para información relevante. Mediante compuertas (forget, update/input y output), siendo cada una implementada como una pequeña red + función sigmoidal + multiplicador, la LSTM decide qué olvidar, qué incorporar y qué exponer, permitiendo memoria a corto y largo plazo y mitigando problemas como el desvanecimiento del gradiente. [2]

La estructura de una red LSTM está conformada por los siguientes elementos:

- Celda de estado c_t : : almacén lineal de información a lo largo de la secuencia.
- Compuerta de Olvido (Forget Gate) $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$: filtra (elimina) partes de c_{t-1} , determina qué fracción de la información previamente almacenada se debe descartar.
- Compuerta de Entrada (Update/Input Gate) $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$ y candidato $\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$: proponen y seleccionan nueva información a añadir, controla qué información nueva debe incorporarse a la celda de memoria.
- Compuerta de Salida (Output Gate) $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$: decide qué parte de la celda se convertirá en el nuevo estado oculto, regula qué información de la celda se utilizará para generar la salida en el paso actual.

Para comprender cómo una celda LSTM mantiene y actualiza su memoria, es necesario detallar el procedimiento matemático que combina la información anterior con la nueva entrada. Este proceso está gobernado por las compuertas y se resume en los siguientes pasos, donde cada fórmula representa una transformación clave del estado de la red.

1. Calcular f_t con h_{t-1} y x_t .
2. Eliminar lo irrelevante: $f_t \odot c_{t-1}$.
3. Calcular candidato \tilde{c}_t e i_t
4. Filtrar e incorporar: $i_t \odot \tilde{c}_t$
5. Sumar para obtener la nueva celda:

$$c_t = f_t \odot C_{t-1} + i_t \odot \tilde{c}_t$$

Donde:

- σ corresponde a la función sigmoidal definida como

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

que toma valores entre 0 y 1. Esto permite interpretar la salida como un “filtro” o “válvula”, ya que valores cercanos a 0 bloquean la información, mientras que valores cercanos a 1 la dejan pasar.

- El símbolo \odot representa el producto elemento a elemento (también llamado Hadamard product). A diferencia de la multiplicación matricial, este operador multiplica cada compo-

nente del vector de manera independiente, lo que permite aplicar máscaras o filtros sobre la información.

Una vez actualizada la celda de estado c_t , el siguiente paso es calcular el nuevo estado oculto h_t , que será utilizado tanto como salida de la celda en el instante actual como entrada para el siguiente paso temporal de la red. Este cálculo se realiza en dos etapas: primero, se transforma el contenido de la celda mediante la función tangente hiperbólica para acotar sus valores en el rango $[-1, 1]$; luego, la compuerta de salida o_t actúa como filtro, determinando qué fracciones de la memoria interna se expondrán al exterior.

$$h_t = o_t \odot \tanh(C_t)$$

De este modo, el estado oculto h_t constituye una versión controlada y filtrada de la celda de estado, que refleja únicamente la información relevante para el aprendizaje y la predicción en cada paso de la secuencia.

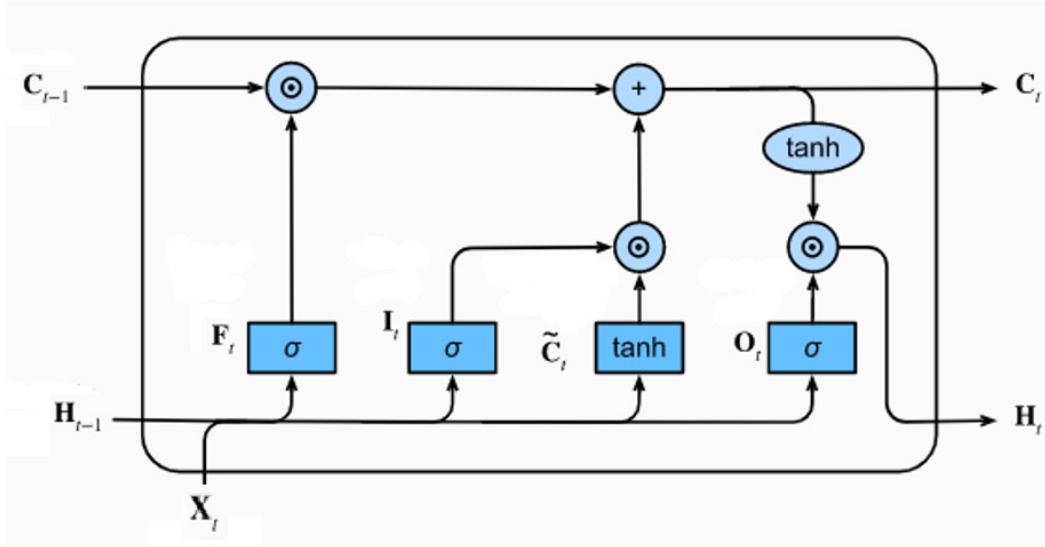


Figura 2: Procedimiento paso a paso para actualizar la celda de estado en una Red LSTM

5.2.4. GBR

El Gradient Boosting Regression (GBR) es un algoritmo de aprendizaje supervisado basado en la técnica de boosting, introducida por Friedman (2001). El boosting es una estrategia de ensamblado que consiste en construir un modelo fuerte combinando secuencialmente múltiples modelos débiles, generalmente árboles de decisión de baja profundidad. [3] [4]

La idea central del Gradient Boosting es ajustar iterativamente nuevos modelos a los residuos del modelo anterior, de manera que cada paso corrige los errores cometidos previamente. El procedimiento general puede resumirse en los siguientes pasos:

1. Se entrena un modelo inicial (por ejemplo, un árbol de decisión poco profundo).
2. Se calculan los residuos (errores) de las predicciones respecto a los valores reales.
3. Se entrena un nuevo modelo para predecir esos residuos.
4. El nuevo modelo se combina con los anteriores mediante una tasa de aprendizaje (learning rate) que controla la contribución de cada estimador.

5. El proceso se repite iterativamente hasta alcanzar un número predefinido de iteraciones o hasta que el error converja.

El término “gradient” en Gradient Boosting proviene de que, en lugar de ajustar los modelos directamente a los residuos, la optimización se formula como un descenso por gradiente sobre una función de pérdida arbitraria (por ejemplo, error cuadrático medio en regresión). Esto hace que el método sea altamente flexible y aplicable a una amplia variedad de problemas de regresión y clasificación.

Las principales características del GBR son:

- Alta capacidad predictiva, incluso en escenarios con **relaciones no lineales** complejas (como lo son en nuestro caso).
- Manejo eficiente de variables heterogéneas (numéricas y categóricas).
- Posibilidad de ajustar hiperparámetros como el número de árboles, la profundidad máxima, la tasa de aprendizaje y la función de pérdida.

En aplicaciones agrícolas y climáticas, el GBR ha demostrado gran efectividad para modelar relaciones no lineales entre variables ambientales y rendimientos productivos, complementando a modelos más secuenciales como las LSTM. Por ejemplo, puede ser utilizado para estimar el rendimiento esperado de un cultivo en función de variables agregadas como temperatura, precipitación acumulada, humedad relativa o velocidad de los vientos.

Modelos débiles en Gradient Boosting

El algoritmo de Gradient Boosting se fundamenta en la construcción de un ensamble de predictores simples, conocidos como modelos débiles. En el caso de las implementaciones utilizadas en este trabajo, dichos modelos corresponden a árboles de decisión de regresión poco profundos. La elección de esta estructura no es arbitraria: los árboles de baja profundidad son lo suficientemente flexibles como para capturar relaciones no lineales en los datos, pero lo bastante simples como para no sobreajustar individualmente. Su verdadera potencia surge al ser combinados de manera secuencial.

Cada árbol se entrena con el objetivo de corregir los errores residuales cometidos por el ensamble acumulado hasta el momento. De esta forma, el primer árbol intenta explicar la diferencia entre la predicción inicial (generalmente el promedio del valor objetivo) y los valores reales. El segundo árbol se ajusta para modelar los residuos restantes después de incorporar el primer árbol, y así sucesivamente. En cada iteración, la salida del nuevo árbol se agrega a la predicción acumulada, ajustada mediante un factor de aprendizaje (learning rate) que regula la magnitud del aporte de cada modelo débil al resultado final.

El proceso iterativo de corrección convierte al conjunto de árboles en un modelo fuerte capaz de capturar dependencias complejas y patrones de difícil detección. En términos conceptuales, se puede entender a cada árbol débil como una capa adicional que refina progresivamente la aproximación, similar al rol que cumplen las capas ocultas en una red neuronal. Sin embargo, la diferencia principal radica en la forma en que se realiza la optimización: mientras que en una red neuronal la retropropagación ajusta pesos de manera global, en Gradient Boosting cada nuevo árbol se entrena localmente para minimizar el gradiente de la función de pérdida.

Esta metodología asegura que el modelo final no dependa de un único árbol complejo, sino de la combinación de múltiples árboles simples. Así, se logra un equilibrio entre sesgo y varianza, lo que

convierte al Gradient Boosting en un enfoque altamente efectivo para problemas de predicción tanto en regresión como en clasificación. [5] [6]

5.2.5. Árboles de Decisión

Los árboles de decisión constituyen uno de los modelos más utilizados y versátiles en el campo del aprendizaje automático. Su funcionamiento se basa en representar el proceso de toma de decisiones mediante una estructura jerárquica en forma de árbol, en la cual los nodos internos corresponden a condiciones sobre los atributos de entrada, las ramas representan los resultados de esas condiciones, y las hojas indican la predicción final.

En el caso de la clasificación, las hojas contienen etiquetas de clase y cada decisión en el recorrido del árbol conduce a asignar un ejemplo a una categoría determinada. En el caso de la regresión, las hojas almacenan valores numéricos y el árbol predice un valor continuo. En ambos contextos, la idea central es dividir recursivamente el espacio de datos en regiones cada vez más homogéneas en relación con la variable objetivo.

El entrenamiento de un árbol de decisión se realiza mediante un proceso denominado partición recursiva. En cada nodo, el algoritmo selecciona la característica y el umbral de división que mejor separan los datos según un criterio de impureza o error. Entre los criterios más utilizados se encuentran el índice Gini y la entropía para problemas de clasificación, mientras que para regresión suele emplearse el error cuadrático medio. Una vez definida la mejor división, los datos se reparten en ramas y el proceso se repite hasta alcanzar un criterio de detención, como puede ser la profundidad máxima del árbol, el número mínimo de muestras por nodo o la ausencia de mejora en la partición.

Los árboles de decisión presentan varias ventajas: son fáciles de interpretar, ya que la estructura resultante puede representarse gráficamente y entenderse como un conjunto de reglas de tipo “si... entonces”. Además, pueden manejar tanto variables categóricas como numéricas y requieren poca preparación de los datos. Sin embargo, también poseen limitaciones, entre las que se destacan la tendencia al sobreajuste cuando el árbol es demasiado profundo, la inestabilidad frente a pequeñas variaciones en los datos de entrenamiento, y la dificultad para modelar relaciones demasiado complejas sin volverse excesivamente grandes.

6. Metodología

El presente proyecto se desarrolló con el objetivo de determinar la distribución óptima de la siembra de distintas semillas en un área específica, con el fin de maximizar la rentabilidad del productor en función de los precios actuales del mercado.

6.1. Selección del Área de Cultivo

Inicialmente, al ejecutar el sistema, se despliega una ventana emergente que permite al usuario calcular la superficie total y la localización exacta del campo. Esta interfaz incorpora un Sistema de Información Geográfica (GIS), un mapa interactivo que posibilita la delimitación de un polígono mediante la selección de puntos vértices. Una vez definida el área de interés, el usuario confirma la selección mediante un clic derecho, lo que genera automáticamente las coordenadas de los vértices y procede al cierre del GIS. Posteriormente, dichas coordenadas son procesadas para calcular la

superficie total en metros cuadrados, identificar el departamento administrativo al que pertenece el campo y obtener el centroide del polígono, definido como el punto medio que representa la posición promedio de todos los puntos que conforman la figura.

6.2. Recuperación y Procesamiento de Datos Climáticos

Con las coordenadas del centroide, se recuperan los registros climáticos de la estación meteorológica más cercana mediante un servicio web proporcionado por la NASA, obteniéndose datos diarios. Estos datos son transformados en series mensuales mediante el cálculo de promedios cada 30 días, generando un archivo histórico de condiciones climáticas en el punto considerado. Dicho archivo se emplea para entrenar modelos predictivos de clima basados en redes neuronales: las *Long Short-Term Memory* (LSTM) se utilizan para predecir humedad y temperatura, dada su capacidad de modelar secuencias temporales y ciclos anuales a partir de ventanas de catorce meses; por su parte, las variables de precipitación y viento, que presentan mayor variabilidad y carecen de patrones secuenciales definidos, son estimadas mediante modelos de *Gradient Boosting Regression* (GBR). Una vez entrenados, los modelos se almacenan en la carpeta `model` para su posterior reutilización sin necesidad de reentrenamiento. [7]

6.3. Preparación de los Datos de Entrenamiento

Se valida la existencia de los archivos requeridos para entrenar el modelo principal de predicción de rendimiento. El archivo central, `df_semillas_suelo_clima.csv`, integra datos de suelo promedio por departamento, registros de semillas cosechadas y series climáticas correspondientes a los catorce meses previos a la cosecha. A partir de este archivo se filtran las variables irrelevantes, generando un conjunto de datos depurado con las entradas necesarias y la variable objetivo para el entrenamiento.

El entrenamiento del modelo principal se realiza únicamente si no existe un modelo previamente almacenado, dado que este proceso es computacionalmente costoso y puede requerir varias horas, especialmente cuando se aplica *hyperparameter tuning*, procedimiento que consiste en evaluar múltiples combinaciones de hiperparámetros para seleccionar aquellas que maximicen el desempeño del modelo. Tanto el modelo entrenado como el diccionario que mapea las semillas a valores enteros se almacenan en `model`, permitiendo su posterior utilización en el algoritmo genético para codificar los nombres de semillas de manera numérica. [8] [9] [10]

6.4. Ejecución del Algoritmo Genético

El algoritmo genético se centra en siete semillas, dado que únicamente para estas se dispone de información confiable de precios de mercado. Cada individuo de la población inicial está representado por un arreglo de siete valores reales que indican la superficie destinada a cada semilla. Estos valores se generan aleatoriamente y deben sumar la totalidad de hectáreas calculadas previamente mediante la aplicación GIS.

Una característica particular del algoritmo implementado es la estrategia de mutación aplicada: dado que los genes corresponden a números reales, se utiliza una mutación de tipo *swap*, que consiste en intercambiar las áreas de siembra de dos semillas seleccionadas aleatoriamente. La función objetivo se define como la ganancia económica esperada para el productor. Para cada individuo, se construye un *dataframe* con los valores de entrada, que incluyen el identificador numérico de la semilla, los datos del suelo, la superficie asignada y las predicciones climáticas de los próximos catorce meses

obtenidas a partir de las redes neuronales previamente entrenadas. El modelo principal devuelve las toneladas estimadas por semilla, las cuales se multiplican por el precio de mercado correspondiente. La suma de estas ganancias parciales constituye el valor objetivo a maximizar en las sucesivas generaciones del algoritmo genético.

6.5. Visualización de Resultados

Al finalizar las iteraciones especificadas, el sistema genera gráficos para el usuario: uno que representa la evolución de la población a lo largo del proceso y otro que muestra la distribución final de la superficie por semilla correspondiente al mejor individuo identificado en la última generación.

7. Algoritmo Desarrollado

El sistema desarrollado tiene como finalidad optimizar la planificación agrícola de un campo específico en Argentina, recomendando la mejor combinación de cultivos en función de las condiciones climáticas y del suelo. Para ello, integra modelos de machine learning con un algoritmo genético que busca maximizar la productividad y la adaptabilidad de los cultivos a cada terreno. El lenguaje de programación utilizado para su implementación es Python.

En primera instancia, el usuario interactúa con el sistema a través de una interfaz gráfica basada en GIS (Sistema de Información Geográfica). Allí selecciona en el mapa la ubicación del campo a analizar, obteniendo de manera automática el área del terreno y su localización geográfica. Esta información inicial resulta clave, ya que determina el contexto en el que se realizarán las predicciones y recomendaciones posteriores.

El siguiente paso consiste en el procesamiento de datos. El sistema integra distintas fuentes: bases históricas de rendimientos de cultivos, información sobre características del suelo y registros climáticos. Estos datos se unifican para generar datasets de entrenamiento que servirán a los modelos predictivos. La limpieza y normalización de estas fuentes permite garantizar que la información sea consistente y representativa de la realidad del terreno.

La predicción de condiciones climáticas constituye un componente fundamental. Para ello, se emplean modelos de machine learning especializados, entre ellos redes LSTM (Long Short-Term Memory), útiles para capturar dependencias temporales, y algoritmos de Gradient Boosting Machines (GBM), que mejoran la precisión mediante ensambles de modelos. De este modo, se generan proyecciones sobre variables críticas como temperatura, precipitaciones, humedad y viento, elementos que afectan directamente el rendimiento agrícola.

En paralelo, el núcleo del sistema está compuesto por un algoritmo genético. Este modelo de optimización combinatoria opera considerando simultáneamente las características del suelo, las predicciones climáticas y el área disponible. La idea central es evolucionar una población de configuraciones de cultivos, donde cada individuo representa una posible distribución de especies agrícolas en el campo. El algoritmo aplica operadores de selección, cruce y mutación para mejorar progresivamente la aptitud de la población, definida en función del rendimiento esperado.

La selección privilegia aquellas configuraciones que logran una mayor productividad y adaptación a las condiciones climáticas previstas. El cruce combina las mejores características de distintos individuos para explorar nuevas alternativas, mientras que la mutación introduce variaciones aleatorias que permiten escapar de óptimos locales, garantizando una búsqueda más exhaustiva.

El flujo completo de ejecución puede resumirse en las siguientes etapas:

1. El usuario selecciona un campo en la interfaz GIS.
2. El sistema identifica la ubicación y obtiene datos del suelo.
3. Se predicen las condiciones climáticas a partir de modelos de machine learning.
4. El algoritmo genético optimiza la distribución de cultivos considerando maximización de producción, adaptabilidad climática y características edáficas o de suelo.
5. Se generan recomendaciones concretas sobre qué cultivos sembrar y en qué proporción hacerlo.

De esta manera, el sistema constituye una herramienta inteligente de apoyo a la toma de decisiones agrícolas, capaz de combinar conocimiento histórico, predicción climática y optimización evolutiva en un entorno único.

8. Resultados

En esta sección se reportan los resultados de la evaluación predictiva (clima y rendimiento) y de la optimización con el algoritmo genético (AG). Todos los gráficos corresponden al conjunto de test o a una ejecución base del AG, según aplique.

8.1. Modelos Climáticos (LSTM/GBR)

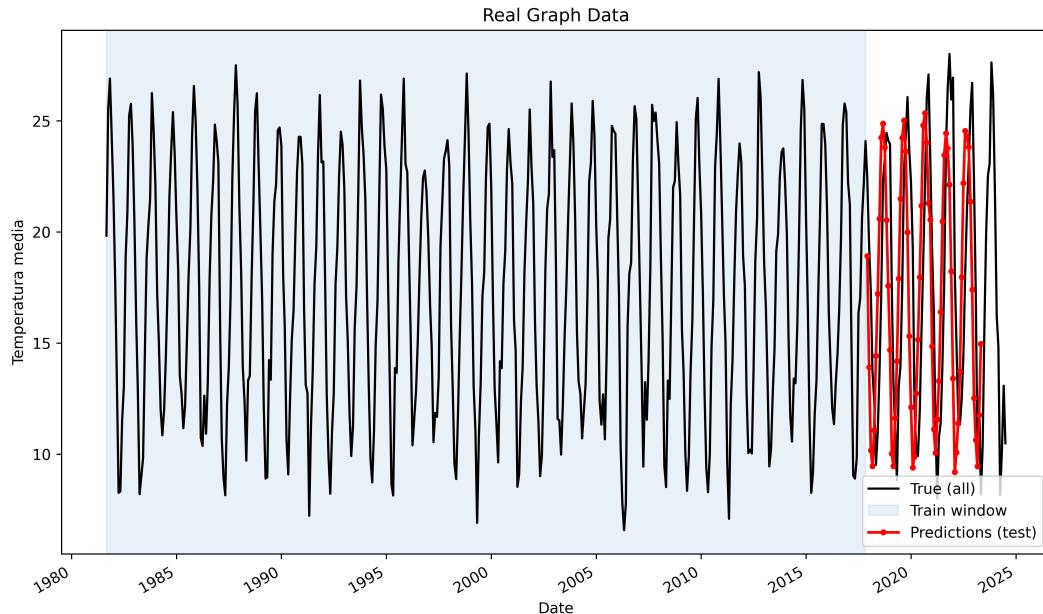


Figura 3: Temperatura media: real vs. predicho (test) en $H = 14$

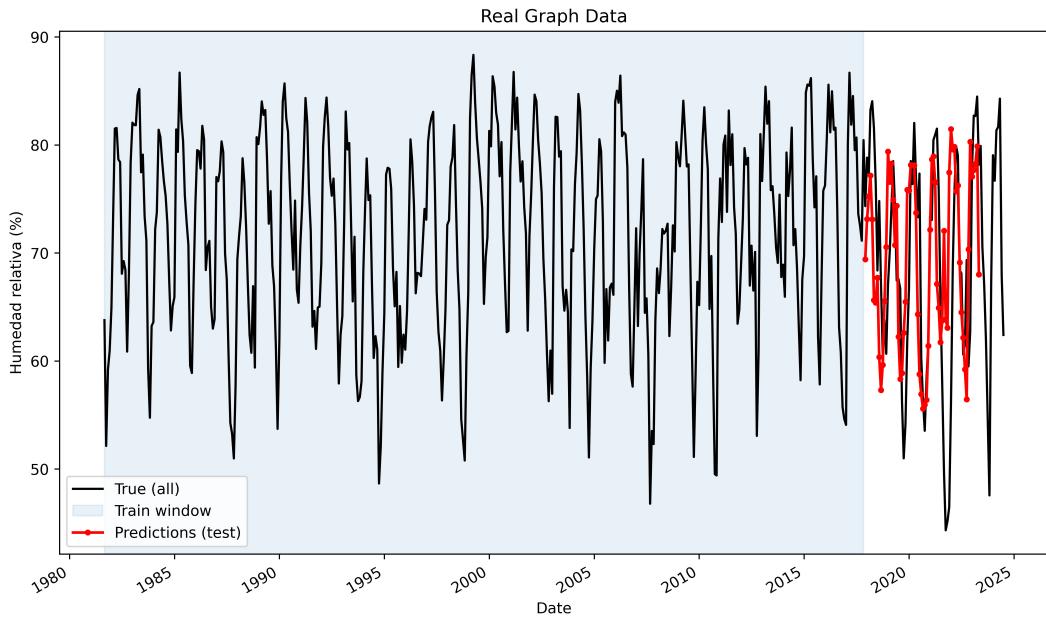


Figura 4: *Humedad relativa: real vs. predicho (test) en $H = 14$*

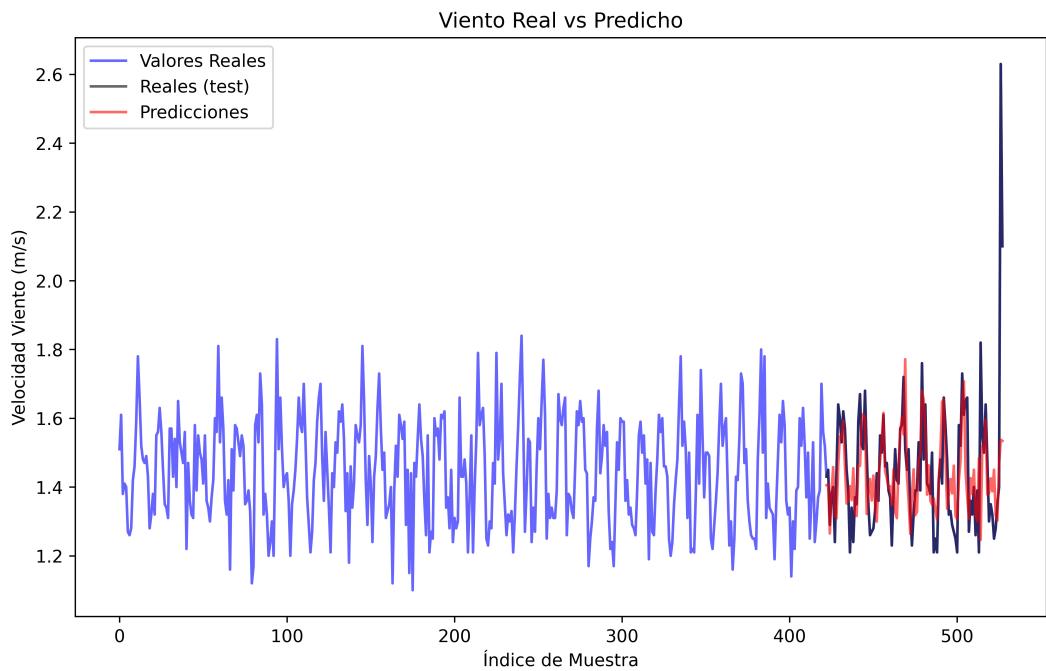


Figura 5: *Viento: real vs. predicho (test) en $H = 14$*

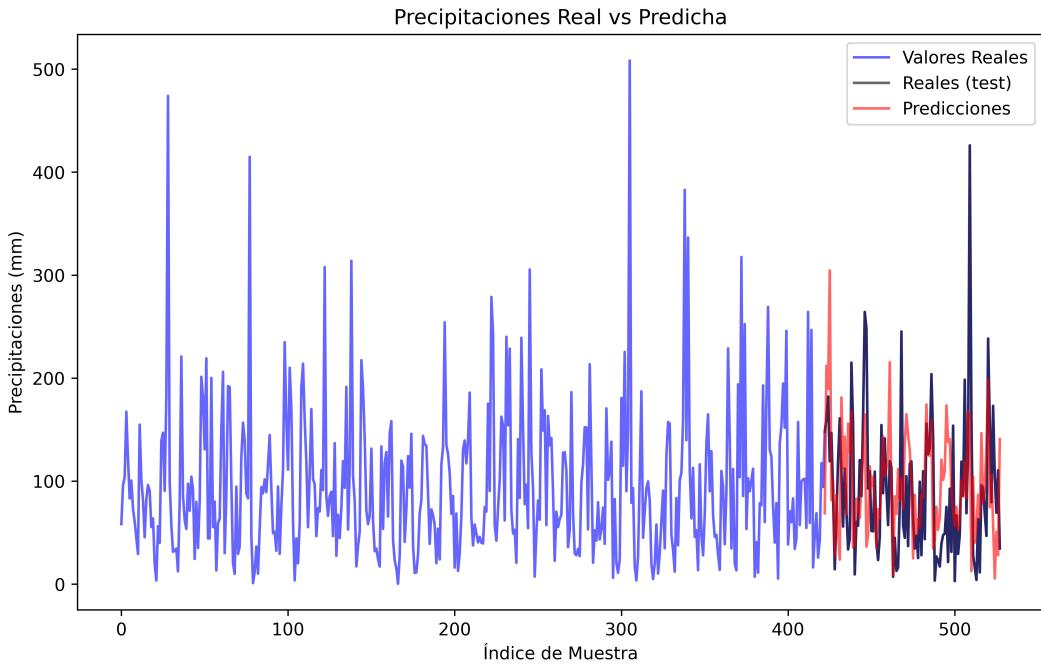


Figura 6: *Precipitaciones: real vs. predicho (test) en $H = 14$*

Las Figuras 3 y 4 muestran el ajuste real vs. predicho para temperatura y humedad (LSTM) en el horizonte $H = 14$. Se observa que el modelo captura de manera consistente la estacionalidad anual y las fluctuaciones intraanuales.

Las Figuras 5 y 6 presentan el desempeño del GBR para viento y precipitación. En viento, el ajuste es estable en el rango observado; en precipitación se aprecia mayor dispersión (propia de su naturaleza intermitente), manteniendo no obstante la tendencia y los niveles medios.

8.2. Modelo de rendimiento

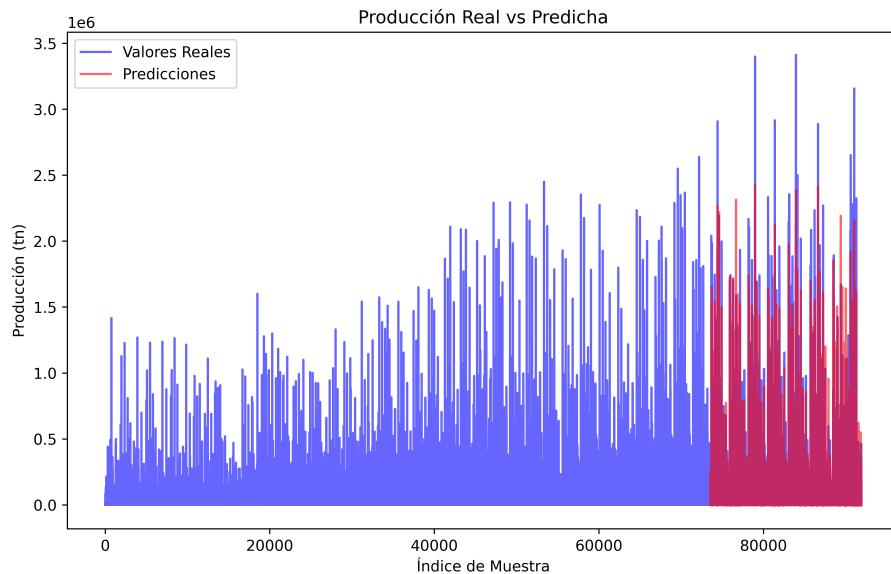


Figura 7: *Producción: real vs. predicha en el conjunto de test*

La Figura 7 contrasta la producción estimada con la serie real. El modelo reproduce las variaciones de escala y la tendencia general; las discrepancias locales se asocian a picos de precipitación y a la variabilidad interanual.

8.3. Optimización con AG

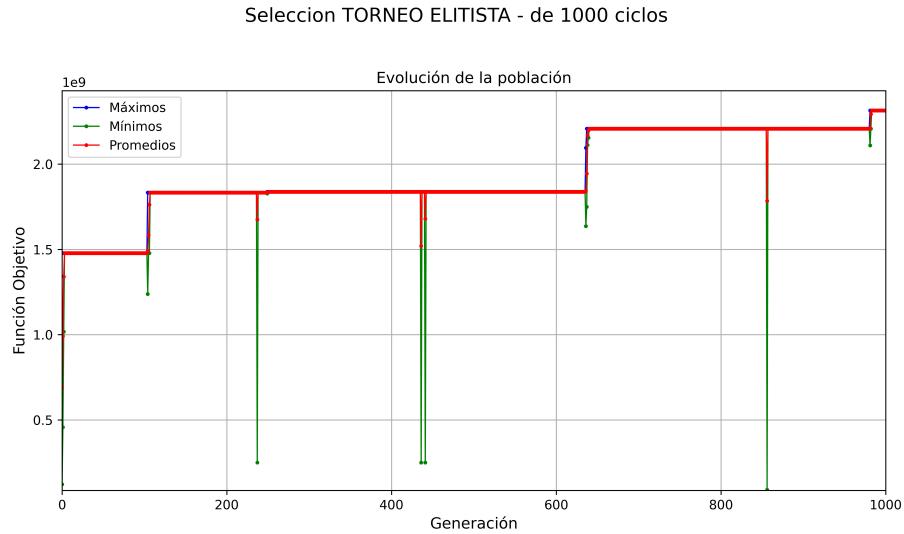


Figura 8: *Convergencia del algoritmo genético: mejor/promedio/peor por generación*

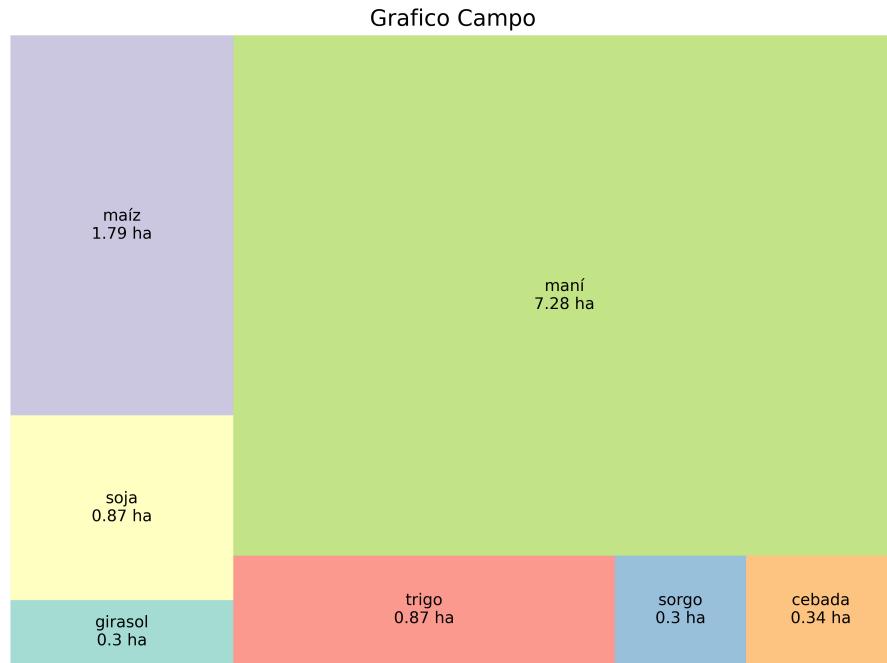


Figura 9: *Distribución de hectáreas por cultivo en la solución final (treemap)*

La Figura 8 muestra la convergencia del AG (selección por *torneo elitista*), con mesetas y saltos característicos al descubrir mejores combinaciones. La solución final se resume en la Figura 9, que

presenta la distribución de hectáreas por cultivo. En la corrida base, el maní domina la asignación, seguido por maíz, con menores áreas para trigo, soja, sorgo, cebada y girasol, consistente con el ingreso esperado dado el rendimiento estimado y los precios vigentes.

9. Conclusiones

En este trabajo se abordó la problemática de la planificación agrícola en un contexto de alta complejidad, integrando predicciones climáticas, modelos de aprendizaje automático y algoritmos genéticos para la optimización espacial de cultivos. La propuesta permitió capturar la interacción entre factores edáficos, climáticos y económicos de manera más realista que con enfoques tradicionales, y sentó las bases para un esquema de planificación flexible y adaptativo.

Los resultados muestran que la combinación de predicciones climáticas y modelos de machine learning mejora la estimación de rendimientos, mientras que los algoritmos genéticos facilitan la búsqueda de configuraciones de cultivo más eficientes en términos de ingreso esperado y aprovechamiento de recursos. Asimismo, la interfaz GIS constituye un aporte práctico, acercando la herramienta al usuario final.

Sin embargo, el estudio enfrentó una limitación importante: la escasez de datos económicos detallados sobre algunas semillas (por ejemplo, cultivos menos difundidos o con escasa trazabilidad en mercados locales). Esta carencia restringió la capacidad del modelo para evaluar alternativas productivas de manera integral, ya que el ingreso esperado no pudo calcularse con la misma precisión para todos los cultivos considerados. De este modo, el análisis económico resultó más robusto en los granos principales que en las opciones secundarias.

En futuras líneas de trabajo será esencial ampliar la base de datos incorporando precios históricos y proyecciones más completas de todos los cultivos relevantes, junto con indicadores de costos asociados. Además, deberían explorarse enfoques multiobjetivo que ponderen no sólo los beneficios económicos, sino también criterios ambientales y sociales.

En síntesis, la investigación confirma el potencial de las herramientas de inteligencia artificial y optimización para transformar la toma de decisiones en la producción agropecuaria, aunque también deja en claro que la disponibilidad y calidad de los datos —especialmente los económicos— constituye un factor crítico para alcanzar recomendaciones más equitativas, precisas y sustentables.

10. Repositorio del Proyecto

El repositorio que contiene el código fuente y los archivos de este trabajo está disponible en:
https://github.com/MondinoJuan/Algoritmos_Geneticos.git

11. Referencias

Referencias

- [1] MSMK University, “Hidden Layer”, disponible en: <https://msmk.university/layer-hidden-layer/>

- [2] CodificandoBits, “Redes LSTM”, disponible en: <https://codificandobits.com/blog/redes-lstm/>
- [3] Wikipedia, “Gradient boosting”, disponible en: https://en.wikipedia.org/wiki/Gradient_boosting
- [4] Scikit-learn, “Ensemble methods — scikit-learn 1.5.2 documentation”, disponible en: <https://scikit-learn.org/stable/modules/ensemble.html>
- [5] Neptune.ai, “Gradient Boosted Decision Trees [Guide]”, disponible en: <https://neptune.ai/blog/gradient-boosted-decision-trees-guide>
- [6] Mehmet H. Toprak, “Gradient Boosting and Weak Learners”, Medium, disponible en: <https://medium.com/@toprak.mhmt/gradient-boosting-and-weak-learners-1f93726b6fb>
- [7] NASA, ”Power Data Access Viewer (API)”, disponible en: <https://power.larc.nasa.gov/2025>
- [8] Bolsa de Cereales de Córdoba, Cotizaciones de cereales”, disponible en: <https://www.bccba.org.ar/todas-las-pizzarras/>
- [9] Bolsa de Cereales y Productos de Bahía Blanca, Cotizaciones de cereales”, disponible en: <https://www.bcp.org.ar/informes.asp>
- [10] Datos Argentina, ”Datasets de semillas, departamentos y suelos argentinos”, disponible en: <https://datos.gob.ar/dataset,2025>