UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE INGENIERÍA
DIVISIÓN DE INVESTIGACIÓN Y POSGRADO

Algoritmos Metaheurísticos: Examen 2. Algoritmo Genético: Optimización de precio de productos lacteos.

Alumno:
Ing. Enrique Mena Camilo

Profesor:
Dr. Marco Antonio Aceves Fernández

Diciembre 2023

Índice

1. Objetivos

El objetivo de esta práctica consiste en optimizar el precio de productos lacteos (lactosa y suero) utilizando la teoría básica de finanzas y el conocimiento adquirido sobre algoritmos genéticos. Esta solución deberá implementar un algoritmo genético sin alguna restricción alguna, sin embargo sí se deberá incluir:

- Análisis estadistico de los datos
- Diseño del modelo de optimización de precio
- Generación de datos sintéticos de inventario
- Algoritmo genético funcional

2. Introducción

Los algoritmos genéticos han emergido como una herramienta poderosa en la optimización de precios de productos, desempeñando un papel fundamental en la toma de decisiones estratégicas para maximizar las ganancias. Inspirados en la evolución biológica, estos algoritmos ofrecen un enfoque innovador para encontrar soluciones óptimas en un espacio de búsqueda complejo y dinámico. En el contexto de la optimización de precios, los algoritmos genéticos permiten explorar eficientemente diferentes configuraciones de precios, adaptándose a las cambiantes condiciones del mercado y las preferencias de los consumidores.

Al aprovechar la selección natural, el cruce y la mutación, estos algoritmos fomentan la evolución de estrategias de precios efectivas. La representación genética de los precios en la población de soluciones potenciales facilita la búsqueda de combinaciones que maximizan las ganancias. Este enfoque innovador no solo optimiza los resultados finales, sino que también se adapta a la dinámica del mercado, permitiendo a las empresas ajustar sus estrategias de precios de manera continua. En resumen, los algoritmos genéticos ofrecen una perspectiva única y eficaz para abordar la complejidad de la optimización de precios, abriendo nuevas posibilidades para la toma de decisiones estratégicas en el ámbito empresarial.

3. Marco teórico

3.1. Modelo básico de optimización de precio

Partiendo de la información proporcionada en [1], se la optimización de precio es un problema en el cual se busca maximizar las ganancias de la empresa. Estas ganancias están establecidas por la Ecuación 1, donde p representa el precio de un producto individual, $D(p)$ representa la función de demanda del producto al precio p, y $R(p)$ representan las ganancias al precio p.

$$R(p) = pD(p) \tag{1}$$

La función de demanda suele asumirse de forma natural que suele disminuir su valor conforme el precio incrementa, por ello es fácil representar dicha función como una función lineal. Esta función se presentan en la Ecuación 2, donde a y b son los parámetros de nuestro modelo que son estimado tras observaciones previas de la variación del precio y la demanda.

$$D(p) = a - bp \tag{2}$$

Implementando la sustitución pertinente, tenemos que el modelo básico optimización de precio consiste en encontrar el p que maximiza el valor de $R(p)$, el cuál está descrito por la Ecuación 3.

$$R(p) = p(a - bp) \tag{3}$$

3.2. Selección poligámica aleatoria

En la selección poligámica aleatoria se forma parejas de forma aleatoria, sin importar si algún individuo es emparejado más de 1 vez. Una desventaja de este método es que existe la posibilidad de no seleccionar a un individuo, lo cuál podría ocasionar que no se explore completamente el espacio de búsqueda. En la Figura 1 podemos observar una representación gráfica de este mecanismo de selección.

3.3. Cruza de dos puntos

La cruza de dos puntos (two point crossover, en inglés) es un método de recombinación comúnmente utilizado en algoritmos genéticos que permite la combinación de características de dos padres para generar una descendencia. Este método permite generar combinaciones de características de ambos padres para explorar y buscar mejores soluciones. En terminos generales, este método consta de los siguientes pasos:

1. Definición de Puntos de Corte: Se seleccionan dos puntos de corte en la representación de los padres. Estos puntos dividen a los padres en tres segmentos: el segmento antes del primer punto de corte, el segmento entre los dos puntos de corte y el segmento después del segundo punto de corte.

2. Creación de Descendencia: La descendencia se crea intercambiando los segmentos intermedios (el segmento entre los dos puntos de corte) de los dos padres. El resultado es un nuevo individuo que combina partes de ambos padres.

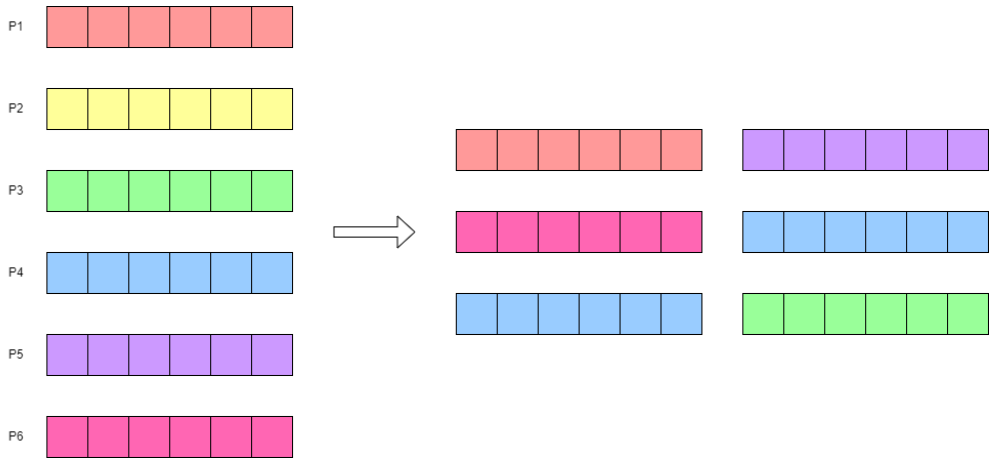

Figura 1: Mecanismo de selección poligámica aleatoria.

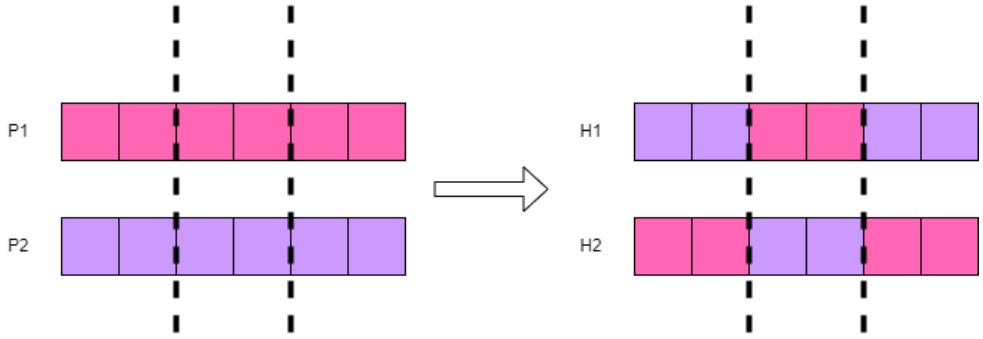

Figura 2: Mecanismo de cruza de dos puntos.

3.4. Mutación por inversión

La mutación por inversión, o "Inverse Mutation", es otra técnica de mutación usada en algoritmos genéticos que trabajan con soluciones representadas como permutaciones. El procedimiento de esta técnica consta de los siguientes pasos:

1. Selección de segmento: Se seleccionan aleatoriamente dos puntos en la permutación. Estos puntos determinarán el inicio y el fin de un segmento.

2. Inversión: El segmento delimitado por estos dos puntos se invierte, es decir, el orden de los alelos dentro de este segmento se revierte.

Este mecanismo se puede observar en la Figura 3.

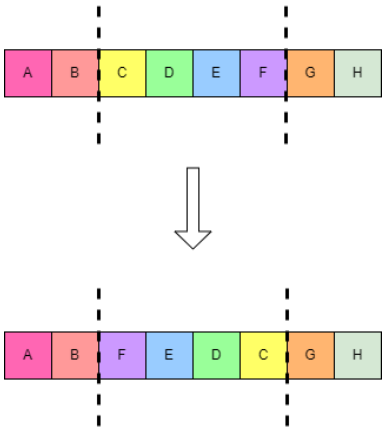

Figura 3: Diagrama de funcionamiento de inverse mutation.

3.5. Competencia genética

El concepto de competencia genética se asemeja a la lucha por la supervivencia en la naturaleza, donde los individuos más aptos tienen una mayor probabilidad de sobrevivir y reproducirse, transmitiendo así sus genes a la siguiente generación. La idea principal detrás de la competencia genética en algoritmos evolutivos es simular este proceso de selección natural para buscar soluciones óptimas o mejores en un espacio de búsqueda.

En este proceso se toma a todos los padres y a todos los hijos, se ordenan de mayor a menor aptitud, y solamente se permitirá que pasen a la siguiente generación aquellos que presentan mejor aptitud, sin importar sin son padres o hijos.

4. Materiales y métodos

4.1. Análisis exploratorio de datos

Se realizó un análisis exploratorio de los datos, el cuál permitió identificar valores de estadística descriptiva de los precios de cada uno de los productos. Para este análisis solamente se tomó en cuenta los precios de los últimos 6 años.

Este proceso adicionalmente implementó un proceso de limpieza de datos, donde se logró estandarizar los diferentes formatos de fecha presentes, así como extraer información como año, mes y trimestre del año.

Teniendo los datos preparados se procedió a obtener gráficas de la distribución del precio de cada producto a lo largo de los diferentes años analizados, así como gráficas que permiten realizar un análisis del precio de los productos en cada trimestre y cómo fue su comportamiento a lo largo de todos los años analizados.

4.2. Funciones de aptitud

Partiendo del modelo básico de optimización de precio, se procedió a generar datos sintéticos para el inventario de cada uno de los productos a lo largo del periodo analizado. Este proceso de generación de datos sintéticos se realizó con las siguientes reglas:

- Valor máximo: $10,000$ toneladas.
- Valor mínimo: $2,000$ toneladas.
- Valor promedio: $5,000$ toneladas.
- Desviación estándar:
  - Lactosa: $2,000$ toneladas.
  - Suero: $1,500$ toneladas.

Teniendo los datos sintéticos de inventario, se insertaron de forma aleatoria a nuestro conjunto de datos para simular la demanda del producto ante cada precio, donde se tomó la asunción de una venta completa, es decir, la demanda del producto fue la misma que le inventario.

Teniendo las consideraciones anteriores, se procedió a obtener un modelo lineal que se ajuste a la variación de la demanda según el precio. Estos modelos lineales siguen la forma de la función de demanda (Ecuación 2).

Una vez estimada la función de demanda para cada producto de interés, se procedió a implementar la función de ganancia (Ecuación 1) con una ligera variación. Considerando que se tienen 2 tipos de cliente (un cliente estándar que compra poco producto y suele tener

un precio estándar; un cliente premium que compra grandes cantidades de producto y suele tener un descuento al precio estándar), se procedió a modificar la función de ganancia para que esta considere el precio mas un 10 %, al realizar esta modificación incrementamos el precio para el cliente estándar, permitiendo también que al cliente premium le apliquemos un descuento y no perdamos ganancias. Tomando esas consideraciones finales, el modelo básico de optimización de precio se estableció como la Ecuación4

$$R(p) = (p + 0{,}10 * p)(a - b(p + 0{,}10 * p)) \tag{4}$$

4.3. Algoritmo genético

Para esta práctica se desarrolló un algoritmo genético que se apega al proceso mostrado en la Figura 4.

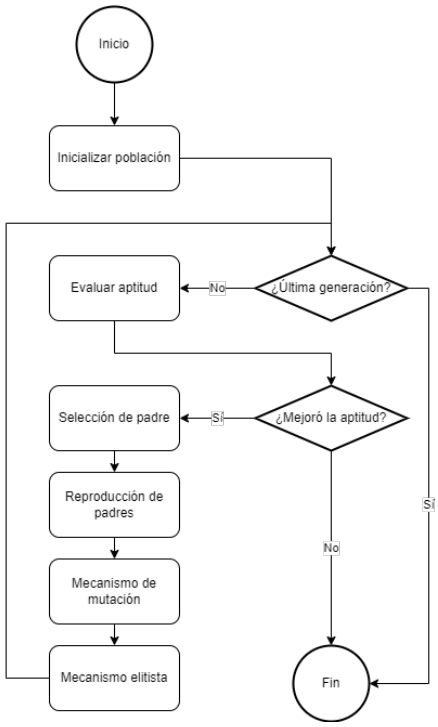

Figura 4: Diagrama de flujo del algoritmo genético implementado.

Para llevar a cabo dicho proceso, se optó por utilizar el lenguaje de programación Python, donde se diseñaron métodos genéricos para realizar los procesos de generación de población, evaluación de aptitud, selección de individuos, reproducción de individuos, mutación y mecanismos elitistas.

4.4. Implementación de algoritmo en Python

4.4.1. Inicializador de población

```python
def init_binary_population(n: int = 10, genes: int = 10):
    min = 0
    max = 2**genes
    population = np.random.randint(min, max, n)
    population = [int_to_bin_array(individue, genes) for individue in population]
    population = np.array(population)

    return population
```

4.4.2. Evaluación de individuos

```python
def evaluate_binary_aptitude(individue: np.ndarray, domain: tuple, aptitude_function: callable):
    bits = individue.shape[0]
    min = 0
    max = 2**bits - 1
    decoded_individue = bin_array_to_int(individue)
    decoded_individue = (decoded_individue - min) / (max - min)
    decoded_individue = (domain[1] - domain[0]) * decoded_individue + domain[0]
    aptitude = aptitude_function(decoded_individue)

    return aptitude


def evaluate_population(population: np.ndarray, domain: tuple, aptitude_function: callable, desc: bool = True):
    aptitudes = np.vstack([evaluate_binary_aptitude(individue, domain, aptitude_function) for individue in population])

    direction = -1 if desc else 1
    sorted_indexes = aptitudes[:, -1].argsort()[::direction]
    sorted_population = population[sorted_indexes]
    sorted_aptitudes = aptitudes[sorted_indexes]
    avg_aptitude = np.mean(sorted_aptitudes)
    max_aptitude = np.max(sorted_aptitudes)

    return sorted_population, sorted_aptitudes, avg_aptitude, max_aptitude
```

4.4.3. Selección de parejas

```python
def polygamous_random_selection(population: np.ndarray):
    n_couples = population.shape[0]//2
    couples = np.random.choice(population.shape[0], (n_couples, 2), replace=True)

    return couples
```

4.4.4. Reproducción de individuos

```python
def two_point_crossover(population: np.ndarray, parents: np.ndarray):
    childrens = np.empty_like(population)
    for idx, couple in enumerate(parents):
        crossover_point = population.shape[1]//3
        childrens[idx*2, :crossover_point] = population[couple[0], :crossover_point]
        childrens[idx*2, crossover_point:2*crossover_point] = population[couple[1], crossover_point:2*crossover_point]
        childrens[idx*2, 2*crossover_point:] = population[couple[0], 2*crossover_point:]
        childrens[idx*2+1, :crossover_point] = population[couple[1], :crossover_point]
        childrens[idx*2+1, crossover_point:2*crossover_point] = population[couple[0], crossover_point:2*crossover_point]
        childrens[idx*2+1, 2*crossover_point:] = population[couple[1], 2*crossover_point:]

    return childrens
```

4.4.5. Mutación

```python
def inverse_mutation(individues: np.ndarray, mr: float = 0.02):
    total_individues = individues.shape[0]
    individue_len = individues.shape[1]
    to_mutate = np.random.choice(total_individues, int(total_individues*mr))

    for idx in to_mutate:
        idx_l, idx_r = sorted(np.random.choice(individue_len, 2))
        idx_r = idx_r if idx_r == 17 else idx_r + 1
        individues[idx][idx_l:idx_r] = individues[idx][idx_l:idx_r][::-1]

    return individues
```

4.4.6. Elitismo

```python
def genetic_competence(population: np.ndarray, childrens: np.ndarray):
    all_population = np.vstack([population, childrens])
    sorted_population, _, _, _ = evaluate_population(all_population, (0, 5), revenue_lactose)
    return sorted_population[:population.shape[0]]
```

4.4.7. Algoritmo completo

```python
lactose_population = init_binary_population(100, 2**4)
generations = 15
epsilon = 3000
last_avg_aptitude = float("inf")
max_stagnant_generations = 3
stagnant_generations = 0
delta = 30

avg_aptitudes_l = []
max_aptitudes_l = []
evolution_l = []

```

```
for _ in range(generations):
    lactose_population, _, avg_aptitude, max_aptitude = evaluate_population(lactose_population, (0, 5), revenue_lactose)
    avg_aptitudes_l.append(avg_aptitude)
    max_aptitudes_l.append(max_aptitude)
    evolution_l.append(lactose_population.copy())

    if abs(avg_aptitude - last_avg_aptitude) < delta:
        stagnant_generations += 1
        if stagnant_generations > max_stagnant_generations:
            break
    else:
        stagnant_generations = 0

    last_avg_aptitude = avg_aptitude
    parents = polygamous_random_selection(lactose_population)
    childrens = two_point_crossover(lactose_population, parents)
    childrens = inverse_mutation(childrens, 0.2)
    lactose_population = genetic_competence(lactose_population, childrens)
```

4.5. Pruebas realizadas

Se implementó una solución en la cuál se utilizan poblaciones con codificación binaria, selección poligámica aleatoria, cruza de dos puntos, mutación por inversión y competencia genética.

Debido a que el objetivo es optimizar el precio de productos lacteos, se realizaron 2 ejecuciones del algoritmo, una para cada producto.

En cada ejecución del algoritmo se generaron gráficas que muestran el proceso de de evolución, en términos de aptitud y convergencia dentro del espacio de búsqueda.

El criterio de paro utilizado se estableció según los siguientes criterios:

- Número de generaciones. Se estableció un límite de 15 generaciones para todos los algoritmos.
- Término por δ. Se estableció un valor $\delta = 30$ con un máximo de 3 generaciones.

5. Resultados

5.1. Análisis de datos

Relacionado al análisis exploratorio de datos, la Tabla 1 muestra las métricas de estadistica descriptiva básica, donde podemos observar que a pesar de existir mínimos y máximos demasiado alejados entre sí, se puede asumir en primer lugar que el precio a lo largo de todo el tiempo ha presentado ligeras variaciones, teniendo una desviación estándar de 0,1037 para la lactosa y 0,1325 para el suero.

Tabla 1: Estadistica descriptiva del precio de los productos.

Producto	Precio			
	Promedio	Std	Minimo	Máximo
Lactosa	0.3873	0.1037	0.1925	0.6350
Suero	0.4404	0.1325	0.2350	0.8050

Por otro lado, la Figura 5 muestra un análisis de distribución del precio en cada uno de los años analizados. Este análisis contradice la información del análisis estadístico básico, ya que muestra que en cada uno de los años sí se han presentando grande variaciones de precio, ya que los valores promedio de cada producto no están cerca de sí en cada uno de los años.

De igual forma se realizó un análisis de la variación de los precios de los productos por trimestre, se realizó por trimestre debido a que es un periodo muy común a analizar en diversas empresas. Este análisis se puede observar en la Figura 6, donde se refuerza el resultado del análisis de distribución por año, ya que se observa que trimestre a trimestre el precio ha mostrado variaciones muy grande, sin mostrar un valor constante entre todos ellos.

Figura 5: Distribución de los precios de lactosa y suero en cada uno de los años analizados.

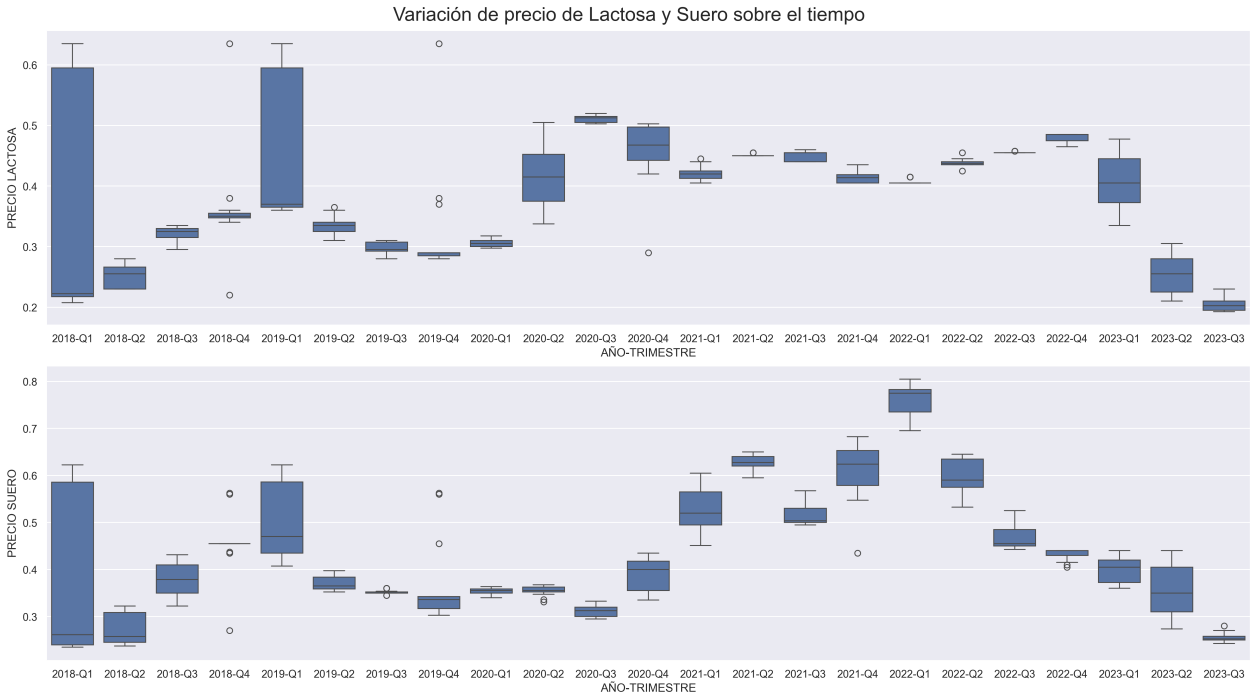

Figura 6: Variación del precio de lactosa y suero en cada trimestre de los años analizados.

5.2. Funciones de aptitud

Relacionado a la función de demanda de lactosa y suero, las Figuras 7 y 8 muestran el resultado de ajustar la ecuación lineal con los datos sintéticos de demanda/inventario.

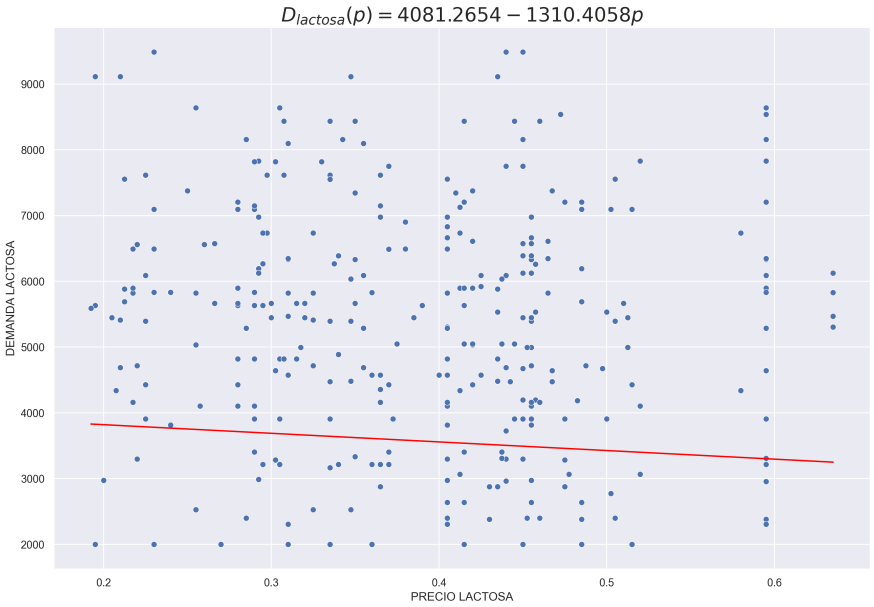

Figura 7: Ecuación lineal ajustada a los datos sintéticos de demanda de lactosa.

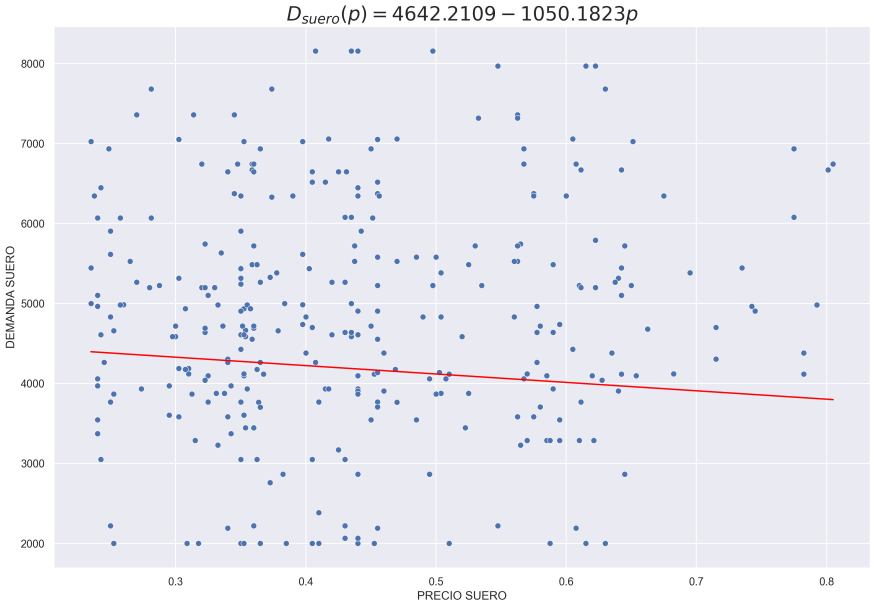

Figura 8: Ecuación lineal ajustada a los datos sintéticos de demanda de suero.

Las funciones de demanda determinadas anteriormente se utilizaron para implementar el modelo básico de optimización de precio, dando como resultado la Ecuacion 5 para la lactosa y la Ecuación 6 para el suero.

$$R_{lactosa}(p) = (p + p * 0{,}10) * (4081{,}2654 - 1310{,}4058 * (p + p * 0{,}10)) \tag{5}$$

$$R_{suero}(p) = (p + p * 0{,}10) * (4642{,}2109 - 1050{,}1823 * (p + p * 0{,}10)) \tag{6}$$

Al graficar dichas los modelos de optimzación de precio se puede observar que sí existe un punto máximo en el cuál las ganancias son las optimas, y podemos también tener una idea del espacio de búsqueda a considerar para encontrar dicho precio optimo. Esta gráfica de los modelos se muestra en la Figura 9.

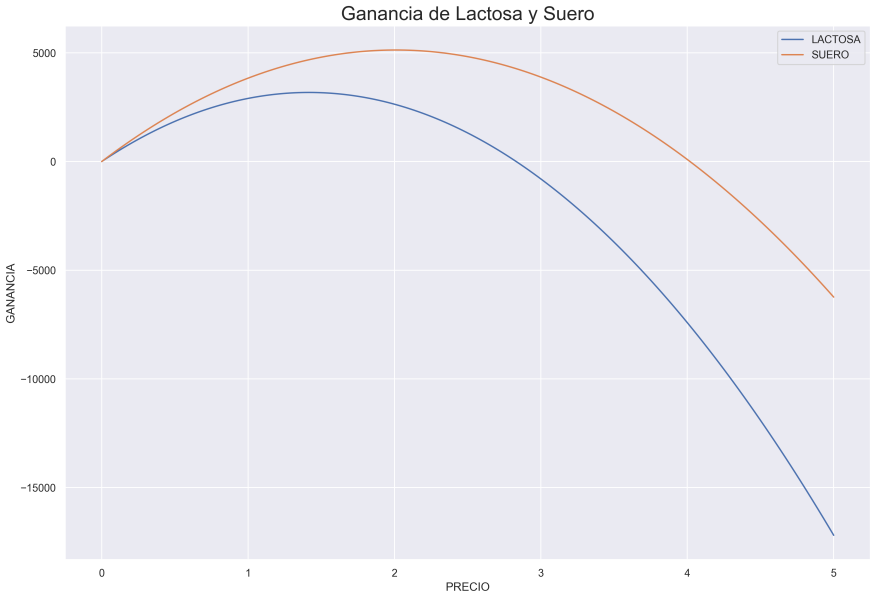

Figura 9: Modelos básicos de optimización de precio ante diferentes precios.

5.3. Algoritmo genético

Al optimizar el precio de cada uno de los productos utilizando un algoritmo genético se obtuvieron los resultados mostrados en las Figura 10 y 11, donde se puede observar que en 8 generaciones se alcanza el precio optimo, el cual permite maximizar las ganancias de cada uno de los productos.

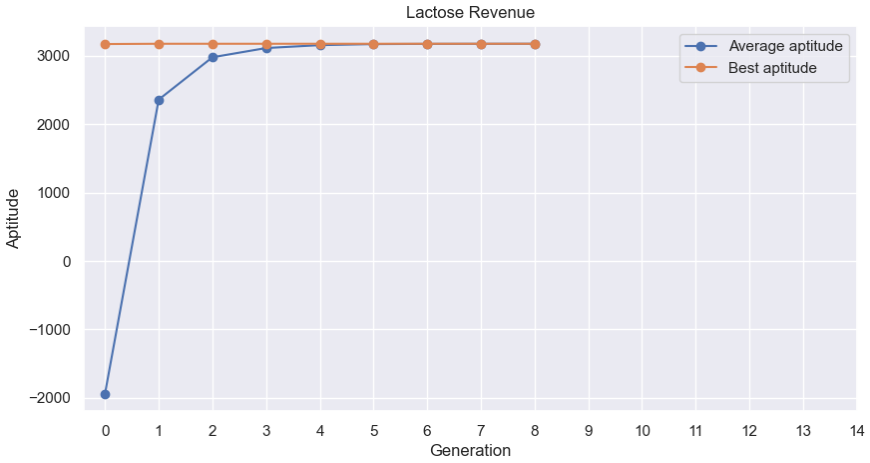

Figura 10: Evolución del algoritmo genético optimizador del precio de la lactosa.

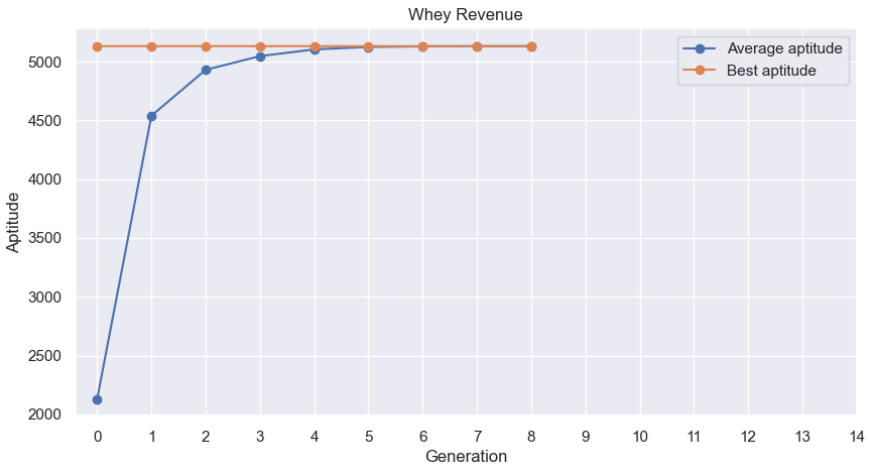

Figura 11: Evolución del algoritmo genético optimizador del precio del suero.

En cuanto al precio optimo para cada uno de los productos, la Tabla 2 muestra el resultado de la última generación de cada algoritmo, donde se puede observar el precio optimo para cliente premium, el precio optimo para cliente estándar y la ganancia para clientes premium.

Tabla 2: Precios finales y ganancia.

Producto	Ganancia	Precio premium	Precio estándar
Lactosa	3,177.7799	1.3926	1.5318
Suero	5,130.0906	2.0259	2.2284

6. Conclusiones

El objetivo principal de este trabajo, encontrar el precio optimo de los productos lacteos, se consiguió de forma satisfactoria. Es cierto que este ejercicio se realizó en gran parte con datos sintéticos, pero la implementación de este se deja abierta para el uso de datos reales del mercado y la industria específica.

Nuevamente se ha observado como los algoritmos genéticos son una herramienta muy valiosa para optimizar procesos en tiempos muy cortos, donde si utilizamos una codificación y función de aptitud adecuada podemos llegar a resultados óptimos.

Para el caso específico del análisis de datos, se puede destacar que siempre es necesario realizar análisis de diversas formas, ya que un análisis básico no siempre nos puede dar la información correcta. En este caso el primer análisis nos indicaba que la variación de precio era mínima, pero al corroborar con análisis más detallados se observó que en realidad no existe un precio que permanezca constante a lo largo del tiempo.

Referencias bibliográficas

[1] K. Talluri and S. Seshadri, *Pricing Analytics*. Springer International Publishing, 2019, p. 793–821. [En linea]. Disponible: http://dx.doi.org/10.1007/978-3-319-68837-4_23

[2] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 2nd ed. Springer, 2015, ISBN: 9783662448731.

[3] K. Dahal, K. C. Tan, and P. I. Cowling, *Evolutionary Scheduling*, 1st ed. Springer, 2007, ISBN: 9783540485827.

[4] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*, 1st ed. Springer, 2008, ISBN: 9783540731894.