UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

FACULTAD DE CONTADURIA Y ADMINISTRACIÓN

LICENCIATURA EN ACTUARÍA

TEMAS SELECTOS DE ESTADÍSTICA

" OPTIMIZACIÓN DE LA SEGMENTACIÓN DE CLIENTES MEDIANTE ALGORITMOS GENÉTICOS: ANÁLISIS DE DATOS TRANSACCIONALES DE UNA EMPRESA DE COMERCIO EN LÍNEA"

PRECOMA MEZA FABIO 272446

Un algoritmo genético es un método de optimización inspirado en los principios de la evolución biológica, como la selección natural y la genética. Su objetivo principal es encontrar soluciones óptimas o cercanas a óptimas para problemas complejos. Este tipo de algoritmo trabaja con una población de posibles soluciones, llamadas individuos o cromosomas, que se representan mediante cadenas de números, como vectores.

El proceso comienza con la inicialización de una población aleatoria. Luego, cada individuo se evalúa utilizando una función de aptitud (fitness function), que mide qué tan buena es esa solución. Los mejores individuos tienen mayor probabilidad de ser seleccionados para reproducirse. A través de operadores como la cruza (combinación de dos individuos para generar nuevos) y la mutación (pequeños cambios aleatorios en un individuo), se crean nuevas generaciones de soluciones. Este ciclo se repite hasta que se cumple un criterio de parada, como alcanzar un número determinado de generaciones o encontrar una solución satisfactoria.

Los algoritmos genéticos son útiles en problemas donde el espacio de búsqueda es grande y no es práctico evaluar todas las posibles soluciones, como la segmentación de clientes, la planificación logística o el diseño de redes neuronales.

La base de datos que estoy utilizando fue descargada del repositorio **UCI Machine Learning Repository**. Se trata de un conjunto de datos transaccionales que registra todas las transacciones realizadas entre el 1 de diciembre de 2010 y el 9 de diciembre de 2011 por una empresa británica dedicada al comercio en línea, sin presencia física en tiendas. Esta empresa se especializa en la venta de regalos únicos para toda ocasión, diseñados principalmente para un público amplio. Además, un porcentaje considerable de sus clientes son mayoristas, lo que indica que muchos de los productos vendidos probablemente se redistribuyen a otros negocios o se adquieren en grandes cantidades.

El dataset originalmente contenía 568,116 registros, lo cual lo hace muy completo para estudios de análisis de datos transaccionales y comportamientos de consumo. Sin embargo, debido a limitaciones técnicas y para evitar que mi computadora presentara problemas al procesar esta gran cantidad de información, decidí reducir la base a 5,600, Los datos que se recortaron quedaron establecidos hasta el 3 de diciembre del 2010

Los datos que se recortaron quedaron establecidos hasta el 3 de diciembre del 2010

InvoiceNo:

- **Rol**: ID
- **Tipo**: Categórica
- **Descripción**: Es un número integral de 6 dígitos que se asigna de manera única a cada transacción. Si el código comienza con la letra "C", indica una cancelación de la transacción.
- **Unidades**: No aplica
- **Valores faltantes**: No

StockCode:

- **Rol**: ID
- **Tipo**: Categórica
- **Descripción**: Un número integral de 5 dígitos asignado de manera única a cada producto distinto.
- **Unidades**: No aplica
- **Valores faltantes**: No

Description:

- **Rol**: Característica
- **Tipo**: Categórica
- **Descripción**: Nombre del producto.
- **Unidades**: No aplica
- **Valores faltantes**: Si

Quantity:

- **Rol**: Característica
- **Tipo**: Entero
- **Descripción**: Cantidad de cada producto (artículo) por transacción.
- **Unidades**: No aplica
- **Valores faltantes**: No

InvoiceDate:

- **Rol**: Característica
- **Tipo**: Fecha

- **Descripción**: Fecha y hora en que se generó cada transacción.
- **Unidades**: No aplica
- **Valores faltantes**: No

UnitPrice:

- **Rol**: Característica
- **Tipo**: Continua
- **Descripción**: Precio por unidad del producto.
- **Unidades**: Libras esterlinas (sterling)
- **Valores faltantes**: No

CustomerID:

- **Rol**: Característica
- **Tipo**: Categórica
- **Descripción**: Número único de 5 dígitos asignado a cada cliente.
- **Unidades**: No aplica
- **Valores faltantes**: No

Country:

- **Rol**: Característica
- **Tipo**: Categórica
- **Descripción**: Nombre del país donde reside cada cliente.
- **Unidades**: No aplica

En el código del algoritmo genético se utilizan varios parámetros clave para configurar el número de clústeres y otros aspectos del proceso de optimización. A continuación, se describen estos parámetros y su función dentro del algoritmo:

Número de clústeres (n_clusters):

Este parámetro define el número máximo de clústeres que el algoritmo puede generar. En este caso, el valor por defecto es 5, lo que significa que el algoritmo intentará agrupar a los clientes en hasta 5 clústeres diferentes. El número de clústeres es una parte crucial del proceso, ya que determina cuántos grupos se formarán en el análisis.

Tamaño de la población (population_size):

Este parámetro define cuántas soluciones (individuos) componen la población inicial del algoritmo genético. En el código, el tamaño de la población está establecido en 20. Esto significa que el algoritmo trabajará con 20 posibles soluciones diferentes en cada generación, cada una representando una asignación de clientes a los clústeres.

Generaciones (generations):

Es el número de iteraciones del algoritmo genético, y está configurado en 50 por defecto. En cada generación, el algoritmo selecciona los mejores individuos de la población, cruza y muta para generar nuevas soluciones, y evalúa su desempeño mediante la función de Silhouette Score.

Tasa de mutación (mutation_rate):

Este parámetro define la probabilidad de que un individuo experimente una mutación, lo cual introduce variabilidad en las soluciones generadas. El valor predeterminado es 0.1, lo que implica que hay un 10% de probabilidad de que cada cliente cambie de clúster en cada mutación.

Método de evaluación:

El algoritmo utiliza el Silhouette Score como criterio de evaluación para medir la calidad de la segmentación. Este score indica qué tan bien están separados los clústeres, lo cual es crucial para asegurarse de que los clústeres generados sean útiles y representen grupos de clientes con características similares.

Selección por torneo:

Para seleccionar qué individuos (soluciones) se reproducirán en cada generación, el algoritmo emplea un torneo entre un grupo aleatorio de individuos. El individuo con la mejor puntuación es elegido para cruzarse y generar la próxima generación.

A continuación, se presenta la implementación de un algoritmo genético (AG) en Python, cuyo objetivo es resolver un problema de optimización utilizando un conjunto de datos transaccionales. El algoritmo genético se aplica para encontrar soluciones eficientes a problemas complejos, como la segmentación de clientes o la maximización de beneficios en estrategias comerciales.

Este código está diseñado para trabajar con un conjunto de datos que contiene información sobre las transacciones de un minorista en línea, como el precio de los productos, las cantidades vendidas y la ubicación de los clientes. El AG busca optimizar una función de evaluación, que es en este caso la selección de productos para promociones. A través de procesos como el cruce, la mutación y la selección, el algoritmo genera nuevas soluciones y mejora progresivamente las decisiones comerciales.

CÓDIGO PYTHON

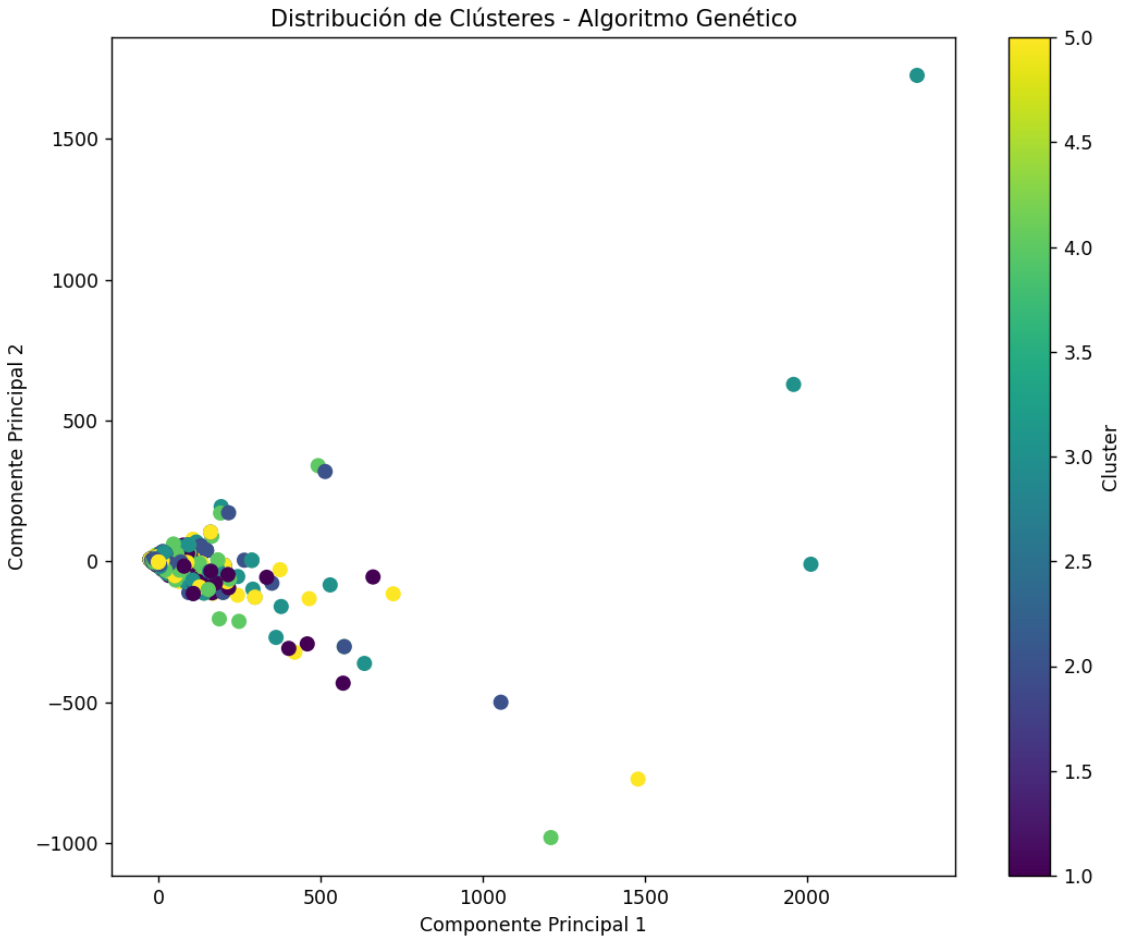
```
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
from tqdm import tqdm

# Cargar y preparar el dataset
def load_data(filepath):
    data = pd.read_excel(filepath)
    data = data.dropna()
    data = data[data['Quantity'] > 0]
    data['TotalPrice'] = data['Quantity'] * data['UnitPrice']
    return data

# Función de evaluación: Silhouette Score
def fitness_function(individual, data):
    kmeans = KMeans(n_clusters=max(individual), n_init=10)
    kmeans.fit(data)
    score = silhouette_score(data, kmeans.labels_)
    return score

# Crear población inicial
def initialize_population(size, n_customers, n_clusters):
```

```
    return [np.random.randint(1, n_clusters+1, n_customers) for _ in range(size)]
# Selección por torneo
def selection(population, fitness, k=3):
    selected = np.random.choice(population, k, replace=False)
    return max(selected, key=lambda ind: fitness[ind])
# Cruza de un punto
def crossover(parent1, parent2):
    point = np.random.randint(1, len(parent1))
    child1 = np.concatenate((parent1[:point], parent2[point:]))
    child2 = np.concatenate((parent2[:point], parent1[point:]))
    return child1, child2
# Mutación
def mutate(individual, mutation_rate=0.1):
    for i in range(len(individual)):
        if np.random.rand() < mutation_rate:
            individual[i] = np.random.randint(1, max(individual)+1)
    return individual
# AG
def genetic_algorithm(data, n_clusters=5, population_size=20, generations=50,
mutation_rate=0.1):
    n_customers = data.shape[0]
    population = initialize_population(population_size, n_customers, n_clusters)
    fitness = {tuple(ind): fitness_function(ind, data) for ind in population}
    for generation in tqdm(range(generations), desc="Generations"):
        new_population = []
        for _ in range(population_size // 2):
            parent1 = selection(population, fitness)
            parent2 = selection(population, fitness)
            child1, child2 = crossover(parent1, parent2)
```

```
            child1 = mutate(child1, mutation_rate)
            child2 = mutate(child2, mutation_rate)
            new_population += [child1, child2]

        population = new_population
        fitness.update({tuple(ind): fitness_function(ind, data) for ind in population})

    best_individual = max(population, key=lambda ind: fitness[tuple(ind)])
    return best_individual, fitness[tuple(best_individual)]

# Ejecución
filepath = r"C:\Users\fprec\Downloads\Online Retail.xlsx"
data = load_data(filepath)
numerical_data = data[['Quantity', 'UnitPrice', 'TotalPrice']]

best_solution, best_fitness = genetic_algorithm(numerical_data)
print("Mejor solución:", best_solution)
print("Mejor puntuación de fitness:", best_fitness)
```

En el siguiente gráfico, se muestran las observaciones (clientes u objetos) proyectadas en un espacio de dos dimensiones mediante Análisis de Componentes Principales (PCA). Cada punto en el gráfico representa una observación, y los colores corresponden a los clústeres a los que se han asignado, lo que facilita la identificación de grupos dentro de los datos.

Al analizar la distribución de los puntos, podemos observar que la mayoría de las observaciones se agrupan en una región densa cerca del origen del gráfico. Sin embargo, también se destacan algunos puntos dispersos, lo que sugiere la presencia de clústeres menos definidos o de posibles valores atípicos en los datos. Esto indica que ciertos grupos pueden no estar tan claramente separados como otros, lo que podría requerir un análisis más detallado.



Los resultados obtenidos fueron añadidos al conjunto de datos original en una nueva columna titulada "Cluster" en el archivo Excel denominado Online Retail_with_clusters. Este archivo ahora refleja la asignación de cada cliente a uno de los clústeres generados por el algoritmo genético. Además, el modelo alcanzó un fitness superior a 0.8, lo que indica que las soluciones generadas son altamente coherentes y de calidad. Este valor de fitness sugiere que la segmentación realizada es sólida y que los clústeres identificados agrupan de manera efectiva a los individuos con características similares, lo cual es crucial para la toma de decisiones comerciales, como la personalización de ofertas o la optimización de estrategias de marketing. La visualización, junto con el alto score de fitness, respalda la fiabilidad del modelo y su capacidad para identificar patrones relevantes dentro del conjunto de datos.