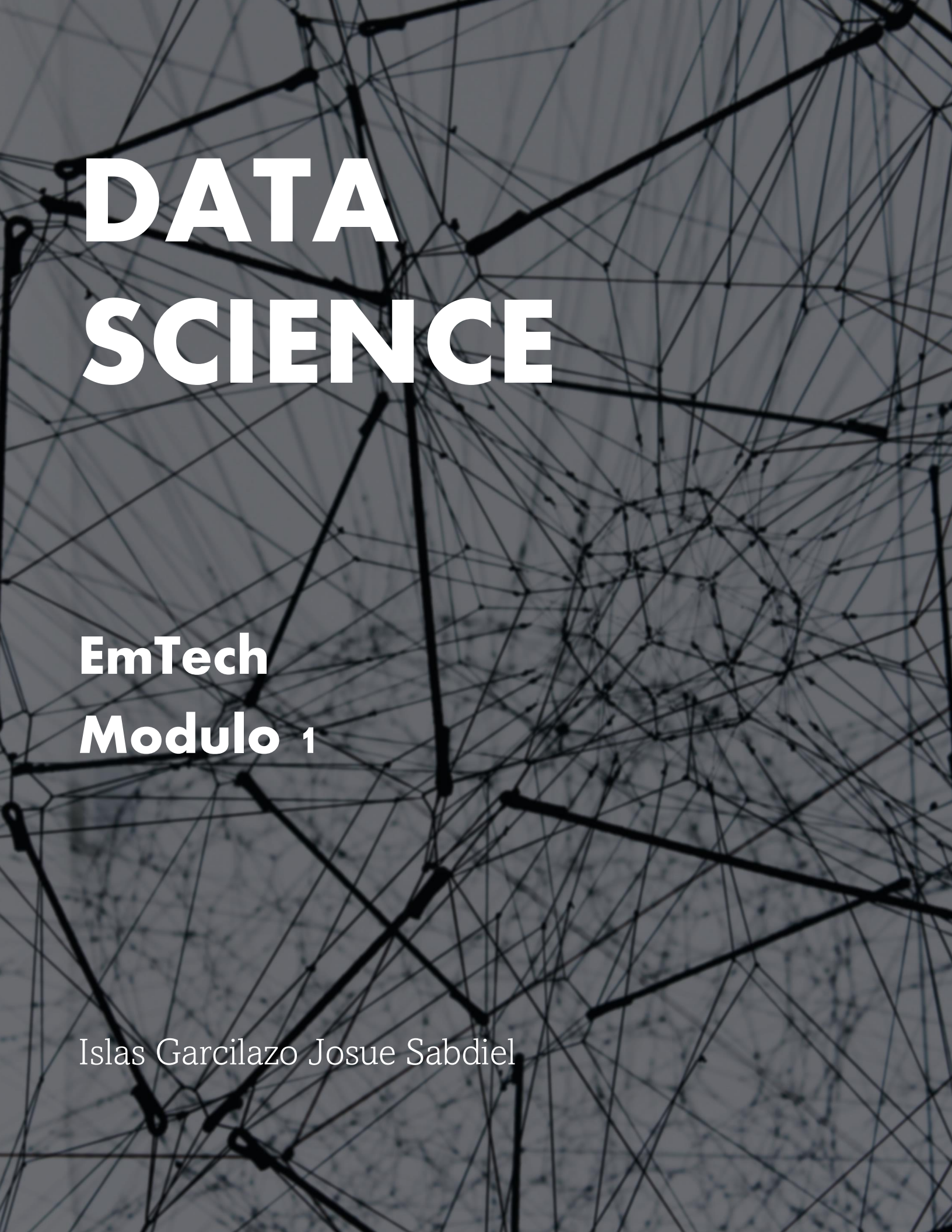

DATA SCIENCE

EmTech
Modulo 1

Islas Garcilazo Josue Sabdiel

ÍNDICE

INTRODUCCIÓN

Dentro de las diferentes formas de solucionar problemas podemos encontrar siempre formas más sencillas y prácticas que otras. Sin embargo, puede ser un reto tratar de adaptarse métodos modernos que desafían los conocimientos vigentes hasta ese momento.

La ciencia de datos ha revolucionado la manera de como se analiza y procesa la información. De hecho, los tiempos de espera que existían hace cincuenta años comparados a los que existen en la actualidad difieren de forma colosal. Tanto es así que la elaboración de tareas manuales como el hacer cálculos matemáticos, tratar la información proveniente de robustas bases de datos usualmente tomaba meses si se abordaba un proyecto robusto, hoy en día se consigue en menos de una hora hacer un análisis igual de robusto con mayor eficacia y precisión.

Tal es el caso que se trata en este documento. Una tienda de electrónicos dedicada específicamente a la venta de componentes de computadoras (procesadores, dispositivos de almacenamiento, tarjetas gráficas, etc.) La cual presenta un grave problema de acumulación en sus inventarios.

El problema radica en los costos adicionales que conlleva el estar conservando productos en almacén debido a su poca rotación y una perdida de ingresos por su estancamiento. Esto la ha llevado a solicitar la ayuda de un especialista en manejo y tratamiento de datos que le brinde una solución al problema vigente.

En este punto se vuelve relevante el conocer las diferentes herramientas que permitan elaborar un análisis de manera eficaz con el propósito de detectar aquellos problemas y ofrecer soluciones optimas que se puedan implementar tan pronto como sea posible.

Lifestore presenta una base de datos robusta con la información de sus productos, ventas y búsquedas, además solicita un análisis para conocer las medidas pertinentes a tomar para mejorar su estrategia de ventas. Por lo que se elaboró un algoritmo para facilitar tanto el análisis de los datos como su comprensión si otros desarrolladores necesitan consultar la forma en como el algoritmo trabaja.

DEFINICIÓN DEL CODIGO

```
'''
Login del programa.
Se presenta un login que solo cuenta con un usuario registrado para dar inicio al programa, mismo que se puede consultar presionando la tecla "u" para
notar el nombre correcto y password que se deben escribir si se decea acceder al programa, de otro modo el login no dejará de preguntar si se quiere
ingresar, validad cualquier otro tipo de usuarios no registrados
'''
[print(' \n') for _ in range(25)]
print('-------------LOGIN')
password_real = '12345'
usuario_real = 'Invitado'
condicion = True
while condicion == True:
    usuario = str(input("          Ingresa el nombre de usuario (Para ver los usuarios presionar la tecla 'u'):  "))
    if usuario == 'u':
        [print(' \n') for _ in range(3)]
        print('Lista de usuarios disponibles:\n------INVITADO\nUsuario: Invitado\npassword: 12345')
        [print(' \n') for _ in range(3)]
    else:
        password = str(input("              Ingresa la password: "))

        if usuario == usuario_real and password == password_real:
            print(f'          bienvenid@: {usuario} ')
            condicion = False
            break
        else:
            [print(' \n') for _ in range(25)]
            print('          Usuario no encontrado')
            [print(' \n') for _ in range(3)]
            continue



''' Iniciamos el programa encontrando los productos mas vendidos y los mayor buscados.

Se debe tener en cuenta que por cuestiones de formato se vera constantemenete el comando:
  [print("\n") for _ in range(25)]
el cual permite hacer una serie de saltos de linea similuando la limpieza de la consola al momento de usar el programa.

Primero se debe conseguir una lista que contenga los productos que efectiamente se vendierón, esto se logra descartando aquellos que se han devuelto despues ser considerados para una compra pero descartados al final del proceso, entonces la compra final no procede y deben eliminarse del listado general.

El vector creado 'producto_ventas' se encargará de contener el id del producto, como las veces que fue vendido y así obetener al final una lista de listas en la que observaremos el id de los productos que mas ventas tuvierón, se debe generar un vector con la siguiente forma:
   producto_ventas = [
                [id_producto_vendido_1, veces_vendido]
                [id_producto_vendido_2, veces_vendido]
                [id_producto_vendido_3, veces_vendido]
                [id_producto_vendido_4, veces_vendido]
                ...
                ]
'''
#-------------------------MAYORES BUSQUEDAS----------------
producto_ventas = [] #Declarando el vector que contendrá los datos deseados con la forma: [id_producto_vendido_1, veces_vendido]
formato_L, formato_R = '<-------', '------->' #Declaramos dos variables que nos permitirán embellecer elcodigo al momento de
                                #Presentar con codigo final.
[print(' \n') for _ in range(25)]           #Serie de espacios que permitem limpiar (de cierta forma) la consola
for producto in lifestore_products:                     #Se inicia un blucle for que iterará en cada no de los elementos
                                        #contenidos en el array "lifestore_products" de donde requerimos
                                        #extraer la información necesaria ya que dentro de si contiene los
                                        #datos de venta y devolución. Recordando que tiene la forma:
```

```
                                        # lifestore_sales = [id_sale, id_product, score (from 1 to 5), date, refund (1 for true or 0
to false)]
                                        #Del cual nos interesan el id_sale y refund para llevar a cabo el conteo.
    lista_interna = [producto[0], 0]          #Se construeye la primera parte del vector deseado con el id del producto vendido,
teniendo la forma hasta ahora
                                    # [id_producto, 0] (cero en el segundo termino hasta que se llene con las veces vendido).
    producto_ventas.append(lista_interna)       #En esta linea se hace el llenado por cada iteración de nuestro bucle en el array
seleccionado

for venta in lifestore_sales:             #Para este punto creamos un nuevo bucle for que permita validar si la compra se
concluyó de manera exitosa o si hubo
    producto_vendido = venta[1]           #una devolución (compra no concluida). Por lo que la variable 'refund' se debe extraer
en una nueva variable llamada
                                #'devuelto' mientras que 'producto_vendido' unicamente contendra el id de la venta que se efectuo
en ese movimiento.
    devuelto = venta[4]                   #Aquí ocurre la asignación de 'refund' a 'devuelto'.

    if devuelto == 0:                     #Nos tomamos un momento para crear una nueva variable llamda 'validez' que contendra
dos valores logicos "True" o
        validez = True                    #"False" dependiendo si la venta se efectuo o no, el numero cero (0) asignado a las ventas
que no se concluyeron
    else:                                 #Recibe "False" y uno (1) en casi de ser un compra exitosa "True".
        validez = False

    if validez == True:                   #En este punto se concreta el llenado del vector deseado incrementando en +1 las unidades
por cada venta exitosa
        producto_ventas[producto_vendido - 1][1] += 1    #para cada id de producto

def segundo_elemento(lista):              #Ahora creamos esta función que unicamente nos retornara la segunda posición de una
lista dada, se verá con relevancia
    return lista[1]                       #unas lineas mas abajo para organizar el interior del vector

producto_ventas.sort(reverse=True, key=segundo_elemento) #Se utiliza la función "sort" que permite organizar un vector de
forma ascendente o descendente dependiendo
                                        # el valor que se le brinde al argumento "key" en donde se introduce la funcion
previamente creada y así
                                        # conseguir una lista ordenada de forma descendente los productos mas vendidos
'''
Una vez concluida la generación del vector con las ventas efectuadas correctamente resta preguntarle al usuario que tantos
items de la lista quiere ver, podria ser
visto como un top de los productos mas vendidos, dejando a elección del usuario el numero a mostrar (top 10, 15, 20, .. etc.)
Debe tenerse en cuenta que al elegir
un numero adecuando para el top, este numero no debe sobrepasar el tamaño maximo de la lista, ni ser un numero negativo.
Por lo que se valida cualquier entrada erroea
a este cuadro de dialogo que surge al usuario preguntando el top productos vendidos a mostrar.
'''
def limite(lista):                        #Se define una nueva función que cuente la longitud maxima de la lista dada, en este casi se le
dará la lista de prodcutos
    return len(lista)                     #totales con los que cuenta la empresa, de esa forma no se puede buscar un top mas allá de
los productos registrados por
                                #en la base de datos de la firma, OJO aun que se asume que no existen valores duplicados.

limite = limite(lifestore_products)       #Se aplica la función de limite antes creada para conocer la longutud maxima del top
haciendo uso de la función nativa
                                #len() que unicamente cuenta el numero de elementos en un vector dado.
while True:                                                 #Para este punto se crea un blucle while que preguntará una
vez se introduzca
    n = int(input(f'Introduce el TOP productos mas buscados (menor a {limite}): '))     #el valor del top deseado, y en caso de dar
numero mayores a los productos
    if n <= limite and n > 0:                               #existentes o numero negativos, volvera a preguntar hasta
que la respuesta
        break                                               #sea un numero positivo menor a la cantidad maxima de
elementos en el registro
    [print(' \n') for _ in range(25)]
    print(f'\n {formato_L} Introduce un valor entre 0 y {limite} {formato_R} \n')
```

```
mas_vendidos = producto_ventas[:n]                                  #Gracias a esta linea se delimita la muestra para que en lugar de mostrar todo
                                                                    #la base de datos generada con los productos mas venidos, unicamente muestra el
                                                                    #numero dado por el usuario en el apartado anterior.
'''
En este punto se crea un algoritmo que permite concatenar dos vectores distintos con un valor común, esto debido a que los datos relevantes como id de los productos, su
nombre, la ventas, el precio o la fecha de vencimiento estan repartidos en diferentes vectores (lifestore_sales, lifestore_products y lifestore_sales). Lo que tienen en
común los tres arrays es el id del producto, por lo que si requerimos una variable de un vector y concatenarla con otra variable diferente ubicada en un vector distinto
podemos usar el id del producto como una especie de llave (key) que haga el emparejamiento satisfactoriamente.

Los pasos a seguir para este ultimo apartado de la primera problematica es juntar el id del producto con mas ventas respetando el orden top que se establecio gracias a
la función sort con su nombre localizado en la lista lifestore_products. Por lo que se deben unir estas dos listas en una sola para lograr una vizualización mas atratctiva
mostrando el nombre del prodcuto y no solo el id que puede resultar dificil de entender a que producto se refiere exactamente.
'''


lista_id_productos = []                                             #Primero, se declara este nuevo vector que contendra unicamente los id de los
                                                                    #productos en la lista de ventas ordenada por mayores veces vendido.
lista_productos = []                                                #Segundo, se declara este segundo array que contendra los nombres de todos los
                                                                    #productos. Como se puede suponer, la mata es hacer que ambas listas previamente
                                                                    #declaradas hagan match en una sola.
[lista_id_productos.append(producto[0]) for producto in lifestore_products]     #Se procede al llenado de las listas declaradas anteriormente gracias a dos
[lista_productos.append(producto[1]) for producto in lifestore_products]        #ciclos for 'in-line' que actuan como mecanismo de llenado de información
#Extraemos id y nombre en lista separadas                           #Una vez terminados los bucles lo que se tiene son dos listas de la siguiente forma:
                                                                    #lista_id_productos = [id del producto] ; lista_productos = [nombre de los productos]
[print(' \n') for _ in range(25)]
'''
Finalmente se hace la conctenación usando las funciones 'Built-in' que Python cuenta por si mismo sin instalar algun tipo de modulo o dependencia externa. La primera función
es ".index()" la cual nos permite conocer la posición en un valor dado en un vector de información, de modo que si se tiene el vector a = ['o', 't', 's', 'k'] y se escribe
a.index('o') Python arrojará 0 (cero) ya que la letra "o" minuscula esta en la posición cero. Pro otro lado, la función __getitem__() los ayudará a obtener el valor de un
vector si conocemos la posición donde este se encuentra ubicado. De modo que al juntar ambas funciones primero debemos conocer la posición de un valor que vamos a buscar
(función: index) para posteriormente extraer ese valor (función: __getitem__). Así es como llevamos a cabo la concatenación de ambos vectores, como ambos cuentan con el id
del producto, facilemnete podemos conocer en que posición se encuentra tal producto en otro vector y pedir nos extraiga la variable que nos interesa y arrojarla en un nuevo
vector que lleve el conteo de cada una de estas consultas, este nuevo vector se llamará 'lista_mas_vendidos' y tendrá la forma:

                    lista_mas_vendidos = [
                        [id producto1, nombre_producto1]
                        [id producto2, nombre_producto2]
                        ...
                        ]

Esta forma de concatenar variables se va a repetir multiples veces a lo largo del programa, por lo que las siguientes veces que se use funcionará de forma similar sin necesidad
de volver a explicar paso a paso su funcionamiento, se referirá a este algorimo como "concatenador"
'''
print(formato_L ,'Top productos mas vendidos (nombre)', formato_R)
```

```
c = 1                                                    #Esta nueva variable unicamente llevará un conteo del top,
incrementando en una
lista_mas_vendidos = []                                  #unidad cada vuelta correspondiente al bucle for para
presentar resultados
for item in mas_vendidos:                                #Se crea un bucle for que iterando sobre la lista con los
productos mas vendidos
   id_product = item[0]                                  #mostrará de forma organizada cada uno de ellos (id y
nombre) en orden de mayores
   posicion = lista_id_productos.index(id_product)       #ventas segun el numero top proporcionado por el
usuario.
   producto_encontrado = lista_productos.__getitem__(posicion)
   lista_mas_vendidos.append([posicion, producto_encontrado])   #Aquí ocurre el llenado del nuevo vector con
la concatnación resultante, ahora solo
   if posicion >= 10:                                    #se deben presentar en un formato agradable a la vista.
      print(f'--------------------|TOP: {c}|\n|id| -------- |Producto|\n|{id_product}| -------- |{producto_encontrado}|\n')      #Se usa una
sentencia if unicamente para
   else:                                                 #respetar la estetica de los id que
suelen
      print(f'--------------------|TOP: {c}|\n|id| -------- |Producto|\n| {id_product}| -------- |{producto_encontrado}|\n')      #modificar la
presentación, pero no repercute
   c += 1                                                #de forma real en el codigo.

'''
En seguida se deben presentar aquellos productos con mayores busquedas, para lograrlo usamos un algoritmo muy similar al
implementado anteriormente para mostrar los
productos mas buscados. Solo que en este caso como no se puede usar el criterio de "refund" para saber cuando un producto
de sebe asignar a la base de datos se decidio
usar unicamenete las veces en que el producto se repite ene l vector de busquedas, de modo que del total de busquedas que se
han hecho, aquel producto que aparezca mas
cantidad de veces (id) será el que se haya buscado mas veces. No representa mas dificultad que implementar la función "count"
que viniendo implementada por default en
Python nos permite conocer el numero de veces que un valor se repite en un vector.

Se busca obtener un vector con la forma:

                    mayores_busquedas = [
                       [id_producto1, veces_aparece1]
                       [id_producto2, veces_aparece2]
                       [id_producto3, veces_aparece3]
                       ...
                    ]

para delimitar cuantas veces se ha buscado un producto en la base de datos.
'''
                                                         #Primero nos enfocamos en la lista lifestore_searches
                                                         #ya que al estar estructurada de la forma [id_search,
id_product]
                                                         #se busca la lista id_producto y asi implementar el
"concatenador"
id_product_search = []                                   #la extracción del id_producto se vaciara en la lista
id_product_search
[id_product_search.append(producto[1]) for producto in lifestore_searches]   #usando el bucle for para vaciar la
información en el nuevo vector declarado

mayores_busquedas = []
for producto in lista_id_productos:                      #Declaramos un bucle for que iterará en el
vector con el id de todos los productos
   veces_aparece = id_product_search.count(producto)     #Contamos las veces que el producto
aparece en la lista de busquedas
   mayores_busquedas.append([producto, veces_aparece])  #Creamos el nuevo vector que se
propuso llamao "mayores_busquedas".

[print(' \n') for _ in range(5)]
mayores_busquedas.sort(reverse=True, key=segundo_elemento)   #Ordenamos los elementos en el
nuevo vector creado para tener una vista
                                                         #de los productos que mas se busca, a los que menos se
```

```
buscan

while True:                                                    #Nuevamente se usa el algoritmo de pregunta validada
para delimitar el
    n = int(input(f'Introduce el TOP productos mas buscados (menor a {limite}): '))          #el tamaño del top productos mas
buscados
    if n <= limite and n > 0:
        break
    [print(' \n') for _ in range(25)]
    print(f'\n {formato_L} Introduce un valor entre 0 y {limite} {formato_R} \n')

mas_busquedas = mayores_busquedas[:n]                                  #Se delimita nuevamente la muestra a
elección previa del usuario

print(formato_L ,'Top productos mas buscados (nombre)', formato_R)
c = 1
lista_mas_buscados = []
for item in mas_busquedas:                                     #En este concatenador se pretende unir el id
del producto con n
    id_product = item[0]                                       #cantidad de veces buscado con los nombres para
una mejor visualización
    posicion = lista_id_productos.index(id_product)                    #de modo que pase de este vector:
[id_producto, veces_buscado] a
    producto_encontrado = lista_productos.__getitem__(posicion)              #[id_producto, nombre_producto]
    lista_mas_buscados.append([posicion, producto_encontrado])
    if posicion >= 10:
        print(f'------------------|TOP: {c}|\n|id| -------- |Producto|\n|{id_product}| -------- |{producto_encontrado}|\n')
    else:
        print(f'------------------|TOP: {c}|\n|id| -------- |Producto|\n| {id_product}| -------- |{producto_encontrado}|\n')
    c += 1

'''
Para el analisis de categorias es necesario obtener una lista de listas con cada uno de los productos corresponientes a cada
categoria.
Este pequeño fracmento de codigo unicamente agrupará los productos por categoria para posteriormenete llevar un análisis de
categorias
con los productos mejor reseñados.
'''
categorias = []
# vamos a revisar la lista entera de productos
for producto in lifestore_products:
    # Leemos la categoria del producto
    categoria_del_producto = producto[3]
    # Revisamos si esa categoria aun no esta en nuestra lista de 'categorias'
    if categoria_del_producto not in categorias:
        # Si aun no esta, la agregamos
        categorias.append(categoria_del_producto)

categoria_productos = []
for categoria in categorias:
    cat_prods = [categoria, []]
    categoria_productos.append(cat_prods)

# Vamos a llenar cada lista con los productos que pertenecen a esa categoria
for lista in categoria_productos:
    # La categoria de la lista es el primer elemento
    categoria_de_lista = lista[0]

    # Vamos a revisar todos los productos de lifestore
    for producto in lifestore_products:
        # Como pretendemos guardar el ID unicamente, obtenemos el ID del producto
        id_del_producto = producto[0]
        # Para saber la categoria del producto lo obtenemos asi:
        categoria_del_producto = producto[3]
```

```
        # Si coinciden ambas categorias, vamos a incluir el producto en la lista
        if categoria_del_producto == categoria_de_lista:
            lista[1].append(id_del_producto)

#------------------------- MENORES VENTAS Y BUSQUEDAS POR CATEGORIA
'''
Dentro del análisis de categorias se busca representar aquellos pructos con mayores y menores busquedas y ventas, difiere del analisis
preliminar de la pimera parte ya que este usa las agrupaciones por cataegoria para presentar de nueva forma cada uno de los productos
así se tiene una vista mas amplia de los grupos potenciales que no estan aportando utilidades a la empresa y lograr una estrategia en aquellas
categorias con menos ventas para promocionarlos o definitivamente retirarlos del catalogo
'''

#INFORMACIÓN
#-------------
#Lista mas vendidos: mas_vendidos
#                           [id_producto, n_ventas]
#Lista mas buscados: mas_busquedas
#                           [id_producto, n_busquedas]
#Lista categorias: categoria_productos
#                           # categoria_productos = [
                                    #     [categoria, []],
                                    #     [categoria, []],
                                    #     [categoria, []],
                                    #     [categoria, []],
                                    #     [categoria, []],
                                    #     ...
                                    #     ]


menos_ventas_full = producto_ventas                                     #Creamos nuevas listas de los productos mas vendidos
menos_busquedas_full = mayores_busquedas                                #que posteriormente seran ordenadas al revez

menos_ventas_full.sort(reverse=False, key=segundo_elemento)                     #Ordenamos la lista para obtener menores ventas
menos_busquedas_full.sort(reverse=False, key=segundo_elemento)                  #Ordenamos la lista para obtener menores busquedas
[print(' \n') for _ in range(5)]
#-----------------------MENORES VENTAS                                 Se pregunta el tamaño de la muestra
while True:
    n = int(input(f'Introduce el TOP productos menos vendidos (menor a {limite}): '))
    if n <= limite and n > 0:
        break
    [print(' \n') for _ in range(25)]
    print(f'\n {formato_L} Introduce un valor entre 0 y {limite} {formato_R} \n')

menos_ventas_muestra = menos_ventas_full[:n] #Delimitar la muestra

#----------------------MENORES BUSQUEDAS                                  Se pregunta el tamaño de la muestra
while True:
    n = int(input(f'Introduce el TOP productos menos buscados (menor a {limite}): '))
    if n <= limite and n > 0:
        break
    [print(' \n') for _ in range(25)]
    print(f'\n {formato_L} Introduce un valor entre 0 y {limite} {formato_R} \n')

menos_busquedas_muestra = menos_busquedas_full[:n] #Delimitar la muestra

#-----------------------------------------MENOS CATEGORIAS
[print(' \n') for _ in range(5)]
print(f'Menores ventas {menos_ventas_muestra}\nMenores busquedas {menos_busquedas_muestra} ')
[print(' \n') for _ in range(5)]
```

```
print(formato_L ,'Top productos menos vendidos (categoria)', formato_R)
c = 1
lista_menos_vendidos = []
lista_categorias = []
[lista_categorias.append(producto[3]) for producto in lifestore_products]
for item in menos_ventas_muestra:
    id_product = item[0]
    posicion = lista_id_productos.index(id_product)
    categoria_encontrada = lista_categorias.__getitem__(posicion) #Localiza la categoria
    producto_encontrado = lista_productos.__getitem__(posicion)  #Localiza el nombre del producto
    lista_menos_vendidos.append([posicion, categoria_encontrada])#Esto se hace para hacer que el nombre del producto haga
match
    if posicion >= 10:
        print(f'----------------|TOP: {c}|\n|id| |Producto| |Categoria|\n|{id_product}| {producto_encontrado} |{categoria_encontrada}|\n')
    else:
        print(f'----------------|TOP: {c}|\n|id| |Producto| |Categoria|\n| {id_product}| {producto_encontrado} |{categoria_encontrada}|\n')
    c += 1

[print(' \n') for _ in range(5)]
print(formato_L ,'Top productos menos buscados (categoria)', formato_R)                    #Nuevamente usamos el
concatenador solo que esta vez se imprime
c = 1                                                                                      #De manera similar
lista_menos_buscados = []
lista_categorias = []
[lista_categorias.append(producto[3]) for producto in lifestore_products]
for item in menos_busquedas_muestra:
    id_product = item[0]
    posicion = lista_id_productos.index(id_product)
    categoria_encontrada = lista_categorias.__getitem__(posicion) #Localiza la categoria
    producto_encontrado = lista_productos.__getitem__(posicion) #Localiza el nombre del producto
    lista_menos_buscados.append([posicion, categoria_encontrada])#Esto se hace para hacer que el nombre del producto haga
match
    if posicion >= 10:
        print(f'----------------|TOP: {c}|\n|id| |Producto| |Categoria|\n|{id_product}| {producto_encontrado} |{categoria_encontrada}|\n')
    else:
        print(f'----------------|TOP: {c}|\n|id| |Producto| |Categoria|\n| {id_product}| {producto_encontrado} |{categoria_encontrada}|\n')
    c += 1

'''
Ahora que ya se presentarón los productos mas vendidos resta mostrar los que tienen mejores reseñas. El algorimo
implmentado es bastante similar al primero solo con algunos cambios.
El mas destacable es que ahora no se podrá usar la variable de "refund" (o devolución) como criterio de selección al momento
de contabilizar las ventas, en este caso las reseñas.
Debido a esto se propuso un nuevo metodo que básicamente suma las reseñas arriba de tres (suponiendo que son estrellas si
asi se quiere ver) de modo que aquellos productos con
mayor cantidad de 4 o 5 estrellas se separan del resto, y una vez separados (y como estan medidos en una misma unidad, en
este caso 4 o 5 estrellas) se procede a sumar las estrellas
que han obtenido de modo que si un producto tiene 3 valoraciones de 5 estrellas y 1 de 4 estrellas (5 + 4 = 9) tendra 9 de status,
esta nueva variable nos permite distinguir aquellos
que reciben mejores valoraciones del resto de productos. Ofrece ventajas de explicación y evita el error de que un producto con
33 reseñas de 1 estrella sea mejor que otro con 2 reseñas
de 4 estrellas (ya que el primero tendrá un status de 33 y el segundo de 8 sin ser necesariamente mejor el primero que el
segundo).

Finalemente se presentan dos listas: mas_busquedas

'''
#------------------------- DOS LISTAS CON MEJORES Y PEORES RESEÑAS
[print(' \n') for _ in range(5)]
producto_score_full = []
[producto_score_full.append([producto[1], producto[2]]) for producto in lifestore_sales]
#Obtenemos esto: [id_producto, score]
```

```
score_nombre_mejores = []
score_nombre_peores = []

for item in producto_score_full:
    id_product = item[0]
    score = item[1]
    posicion = lista_id_productos.index(id_product) #Obtenemos la posición a concatenar
    producto_encontrado = lista_productos.__getitem__(posicion) #Localiza el nombre del producto
    #Realizamos un filtrado de solo los productos con 4 o 5 de score
    if score > 3:
        score_nombre_mejores.append([id_product, producto_encontrado, score])
    else:
        score_nombre_peores.append([id_product, producto_encontrado, score])
#Ya logramos concatenar los arrays de modo que resulta: [id_product, name_product, score]
#Ahora solo buscamos cuantas veces se repite el nombre del producto en el array creado

############################  MEJORES Y PEORES RESEÑAS (LLENADO DE VECTORES)
mejor_score_full = []
peor_score_full = []
muestra_nombre_mejores = []
muestra_nombre_peores = []

[muestra_nombre_mejores.append(producto[1]) for producto in score_nombre_mejores] #Para tener solo los nombres de la lista filtrada
[muestra_nombre_peores.append(producto[1]) for producto in score_nombre_peores] #Para tener solo los nombres de la lista filtrada
for item in lista_productos:
    n_veces_mejor = muestra_nombre_mejores.count(item)
    n_veces_peor = muestra_nombre_peores.count(item)
    mejor_score_full.append([item, n_veces_mejor])
    peor_score_full.append([item, n_veces_peor])
#Ahora ya conseguimos saber cuantas veces el producto tuvo buenas reseñas
#Obtenemos el array: [name_producto, veces buena reseña]
#####################################################################
#Ahora solo hay que ordenarlos
#LISTAS FINALES:
#         mejor_score_full, peor_score_full
#                                    [nombre_producto, status]

#Ordenamos
mejor_score_full.sort(reverse = True, key = segundo_elemento)
peor_score_full.sort(reverse = False, key = segundo_elemento)

#Delimitamos la muestra a solo 20 items ya que así lo establece el ejercicio
mejor_score_muestra = mejor_score_full[:20]
peor_score_muestra = peor_score_full[:20]

#Imprimimos resultados
##############################MEJORES
[print(' \n') for _ in range(5)]
print(formato_L ,'Top productos mejor reseñados (by = nombre del producto)', formato_R)
c = 1
for item in mejor_score_muestra:
    posicion = c
    nombre_producto = item[0]
    status = item[1]

    print(f'----------|TOP: {c}|\n|Producto|----------|Status|\n|{nombre_producto}| |{status}|\n')
    c += 1

##############################PEORES
[print(' \n') for _ in range(5)]
print(formato_L ,'Top productos peor reseñados (by = nombre del producto)', formato_R)
c = 1
for item in peor_score_muestra:
```

```
    posicion = c
    nombre_producto = item[0]
    status = item[1]
    print(f'------------------|TOP: {c}|\n|Producto|--------------------------------------|Status|\n|{nombre_producto}| |{status}|\n')
    c += 1


#------------------------- PROMEDIO MESES
meses = []

# vamos a revisar la lista entera de ventas
for venta in lifestore_sales:
    # Leemos la fecha de la venta
    fecha = venta[3]
    # Las fechas vienen en el siguiente formato:
    # DD/MM/AAAA
    dia = fecha[:2]
    mes = fecha[3:5]
    anio = fecha[-4:]
    # print(f'dia {dia} mes {mes} ano {anio}')

    # # Revisamos si ese mes aun no esta en nuestra lista de 'meses'
    if mes not in meses:
        # Si aun no esta, lo agregamos
        meses.append(mes)
[print(' \n') for _ in range(5)]

"""
Listas que vamos a ocupar:

lifestore_sales = [id_sale, id_product, score (from 1 to 5), date, refund (1 for true or 0 to false)]
lifestore_products = [id_product, name, price, category, stock]
"""
#Necesitamos una lista que quede asi: [id_product, price, date]

id_productos_comprados_full = []
id_product_sale_full = []
fecha_productos_full = []
precios_productos_full = []
#Extraemos el id de los productos que realmente se comprarón
[id_productos_comprados_full.append(producto[0]) for producto in producto_ventas]
#Extraemos solo los precios de los productos
[precios_productos_full.append(producto[2]) for producto in lifestore_products]
#Extraemos solo la fecha de venta efectiva
[fecha_productos_full.append(producto[3]) for producto in lifestore_sales]
#Extraemos id_producto_sale para usarlo como pivote de busqueda en ese array
[id_product_sale_full.append(producto[0]) for producto in lifestore_sales]

ventas_x_fecha = []
for item in id_productos_comprados_full:#iteramos el id de venta efectiva
    posicion = lista_id_productos.index(item)#Agarramos pivote el ID en la lista a buscar     | lifestore_products
    precio_encontrado = precios_productos_full.__getitem__(posicion) #La concatenamos con el precio |

    posicion2 = id_product_sale_full.index(item) #pivote id de la lista sales
    fecha_encontrada = fecha_productos_full.__getitem__(posicion2) #concatenamos con la fecha de la venta

    ventas_x_fecha.append([posicion, precio_encontrado, fecha_encontrada])#Esto se hace para hacer que el nombre del
producto haga match, no solo el numero
```

SOLUCIÓN AL PROBLEMA

El problema principal radica en el poco interés por los usuarios de *lifestore* por las tarjetas gráficas que no solo no se están vendiendo, sino que no cuentan con el mínimo de búsquedas para seguir considerando ventas potenciales. El segundo problema es referente al costo de estos componentes que naturalmente se considera elevado, por lo que mantener la misma cantidad en proveedores puede ser lo menos factible ya que se encuentran acumuladas de forma robusta en los almacenes de la empresa, representando una perdida de ingresos al reducir la cantidad de otros componentes como las unidades de almacenamiento o procesadores dentro del almacén.

Una buena medida es tratar con una solida estrategia de mercadotecnia y publicidad que permita acercar las tarjeas graficas al público y hacerlas más atractivas ante los usuarios que se abstienen tal vez por el poco conocimiento que puedan tener el componente o su precio.

Los precios de los componentes como tarjetas gráficas o algunas tarjetas madre pueden hacerse más atractivo si se combina con diferentes promociones o paquetes dependiendo del tipo de usuario que esta consultando la pagina de internet o sucursal, de ese modo aumenta la probabilidad de adquirir componentes que no se están vendiendo correctamente.

Adicionalmente fomentar la retroalimentación o *feedback* de modo que la participación en las reseñas aumente considerablemente ya que existe una gran cantidad de productos que no son reseñados por los usuarios y eso fomenta el poco interés de compradores potenciales en aquellos productos con cero reseñas. Se recomienda que todos los productos tengan al menos una reseña, de ese modo los clientes pueden sentirse mas seguros de la compra que llevarán a cabo y se asegura con mayor fidelidad la transacción.

CONLUSIONES

La implementación de la ciencia de datos en la solución de problemas cotidianos de la industria y academia representa un paso que difícilmente se jubilará debido a lo eficaz que representa el hacer análisis con precisión elevada en una cantidad de tiempo récord aun que no se cuente con el ordenador más sofisticado.

La manera de como el lenguaje de programación Python fue programado permite tener una sintaxis simple y de fácil aprendizaje sin comprometer la velocidad de manera notable, incluso si se maneja con cantidades de información superior a la convencional.

Conocer el comportamiento de los vectores, ciclos y condicionales en el lenguaje aporta una forma eficaz de desarrollar algoritmos diferentes y cada vez mas complejos que son capaces de solucionar cada vez mas problemas, y debido a la multifuncionalidad de Python es que no solo se puede expandir a la ciencia de datos, sino a otras áreas de interés fuera de este simple reporte.

Al momento de llevar a cabo un análisis sobre las ventas, búsquedas y reseñas de la tienda *lifestore* puede notarse tres puntos fundamentales

- El análisis de vectores facilita el tratamiento una vez que se desarrolla la lógica y pensamiento adecuado para procesar cada dato dentro del vector.
- La incorporación de ciclos es la parte fundamental del procesamiento ya que es el encargado de acelerar la velocidad de ejecución y optimizar la forma en como las tareas se van ejecutando iterativamente.
- El uso de condicionales para evaluar grupos específicos de dato o situaciones separadas a la línea principal es de las mejores formas para simplificar el código y ahorrar cada vez más líneas simplemente considerando diferentes alternativas de dirección al momento de programar.

Debido a que no siempre se logra optimizar el código es recurrente caer en declaraciones que abarcan líneas de código innecesarias o que bien se pueden sustituir por funciones ya establecidas anteriormente en el código. El reto por ahora es optimizar cada vez más el código logrando que este funciones de mejor manera con menos líneas de código y bucles o condicionales más específicas a los casos que se deban tratar. Similarmente, el exceso de variables declaradas presenta un problema angular ya que la cantidad innecesaria de memoria a la que se esta usando bien puede dedicarse a tareas especificas y no solo consumirse en tareas irrelevantes.