Becas Santander Tecnología | Desarrollo de competencias de alta empleabilidad – EmTech

Learning Path de Data Science

Python para ciencia de datos

Alan Noe Mazahua Sanchez

EGRESADO DE INGENIERÍA INFORMÁTICA , UNIVERSIDAD VERACRUZANA

DOCUMENTACIÓN

Proyecto Final de Fundamentos de Programación con Python

ÍNDICE

- INTRODUCCIÓN
- DEFINICIÓN DE CÓDIGO
- SOLUCIÓN AL PROBLEMA
- CONCLUSIÓN

INTRODUCCIÓN

Lo que se plantea con el siguiente documento es definir el funcionamiento de un código escrito en python que facilita la obtención de resultados estadísticos, los cuales nos permitirán dar un análisis objetivo a la situación actual de la tienda, de tal forma poder llegar a una conclusión y recomendaciones para mejorar el flujo de venta y evitar pérdidas.

DESCRIPCIÓN DEL CASO

LifeStore es una tienda virtual que maneja una amplia gama de artículos,recientemente, la Gerencia de ventas, se percató que la empresa tiene una importante acumulación de inventario. Asimismo, se ha identificado una reducción en las búsquedas de un grupo importante de productos, lo que ha redundado en una disminución sustancial de sus ventas del último trimestre.

CONSIGNIA

Derivado de la situación, la Gerencia de Ventas te solicita que realices un análisis de la rotación de productos identificando los siguientes elementos:

- 1) Productos más vendidos y productos rezagados a partir del análisis de las categorías con menores ventas y categorías con menores búsquedas.

- 2) Productos por reseña en el servicio a partir del análisis de categorías con mayores ventas y categorías con mayores búsquedas.

- 3) Sugerir una estrategia de productos a retirar del mercado así como sugerencia de cómo reducir la acumulación de inventario considerando los datos de ingresos y ventas mensuales.

DEFINICIÓN DE CÓDIGO

Podemos obtener el código desde la siguiente liga :
https://github.com/Mazahua/Emtech/blob/main/PROYECTO-01-MAZAHUA-SANCHEZ-ALAN-NOE.py

LOGIN

Cabe destacar que el ingreso al programa es con una comparación de campos USER y LOGIN los cuales se encuentran ya definidos por variables globales

```
user = "Mazahua"
user_pass = "3mt3ch"
useri = ""
useri_pass = ""
```

Se realiza una comparación dentro de un ciclo while que nos indica mientras el usuario o contraseña ingresados al sistema sean diferentes de los definidos(globales) se repetirá el ciclo de petición de credenciales.

```
while(user!=useri or user_pass!=useri_pass):  # Bucle mientras usuario o contraseña sean diferentes
de las establecidas
   print()
   print("------------------------------")
   print("|                            |")
   print("|  M E N U  E M T E C H  D S  |")
   print("|                            |")
   print("------------------------------")
   print()

   print("C R E D E N C I A L E S")
   print("Usuario: ", end="")
   useri = input()
   print("Contraseña: ",end="")
   useri_pass = input()

   if user==useri and user_pass==useri_pass:
       print("USUARIO VÁLIDO")



   else:
       print("USUARIO INVALIDO")

   time.sleep(3) # Uso esta funcion ya definida en la biblioteca time para hacer una pausa en este
caso 3 segundos
   borrarPantalla() # Llamo a la función predefinida para limpiar pantalla
```

Si ambos campos coinciden, se imprimirá un mensaje en pantalla como USUARIO VÁLIDO si no se imprimirá USUARIO INVALIDO , después de cualquier impresión sistema tomará 3 segundos de espera , definido por la función time.sleep(3) y borrara la pantalla con una función ya definida.

```
def borrarPantalla(): # Funcion para limpiar pantalla dependiendo del tipo de sistema operativo que
tengas usando la librería os
   if os.name == "posix": # Correspondiente a Windows
       os.system ("clear")
   elif os.name == "ce" or os.name == "nt" or os.name == "dos": # Correspondiente a Linux
       os.system ("cls")
```

Esta función hace una identificación del sistema operativo que estás usando (windows o linux) ya que para cada sistema el método cambia un poco como se muestra en la instrucción.

MENÚ

El cuerpo principal , basado en un ciclo while , que verifica que mientras la variable op sea diferente de 4 se repetirá infinitamente

```
while op != 4:
```

El proceso inicial es un ciclo for que imprime letra por con un lapso de 0.03 segundos creando una animación dentro de la consola de ejecución.
Para explicar la funcionalidad vamos a iniciar con 0 hasta la el tamaño de la cadena usando la función (len) , ocupamos un + 1 como apoyo para imprimir todos los caracteres, dentro del ciclo for vamos a imprimir desde el inicio hasta la posición que nos encontremos actualmente del ciclo colocando texto[0:i] con apoyo de end=’\r’ que realiza la impresión en consola sin dar salto de línea y sobreimprima.
Terminando este ciclo for se espera 0.5 segundos y se limpia la pantalla.

```
   texto = "Se está generando su información por favor elija una opción"
   for i in range(len(texto)+1):
       print(texto[0:i], end="\r") # el "\r" ayuda a imprimir sin dar un salto de linea continuar
imprimiendo donde se quedó el anterior
       time.sleep(0.03)
   time.sleep(0.5)
   borrarPantalla()
```

A Continuación se entra al menú desplegable que te dará a conocer las opciones que tiene el sistema, en este caso son 4

```
   print("       ============================")
   print("           Sistema de Análisis")
   print("       ============================")
   print()
   print("1 | Productos más vendidos, más buscados y rezagados")
   print("2 | Productos top y low por reseña en el servicio")
```

```
print("3 | Estadísticas de ingresos mensual y anual ")
print("4 | Salir del Sistema")
```

Las de la consigna y la última para salir del sistema.

Se encapsula en un try: en el cual si ingresamos un valor no numérico arroja un mensaje "ingrese una opción numérica válida", esto hará que el ciclo se repita y no se interrumpa por error.

También hay una validación dentro del try , que identifica que el valor sea menor a 4 , ya que si el valor ingresado es mayor arrojará un mensaje diciendo que ingrese opción numérica válida,en caso de ingresar un valor menor de 1 , el sistema en consola se volverá a iniciar pidiendo que ingrese una opción válida.

```
try:
    op = int(input("Ingresa la opcion     :    "))
    borrarPantalla()
    if op == 1:
        func_prodtopandlow()
        func_prodtopandlow_search()
        print()
        input(Style.RESET_ALL+"Teclee enter para volver a menu")
    if op == 2:
        func_Storetop()
        print()
        input(Style.RESET_ALL+"Teclee enter para volver a menu")
    if op == 3:
        func_stadistics_sales()
        print()
        input(Style.RESET_ALL+"Teclee enter para volver a menu")
    if op > 4:
        print("No existe una opcion con ese valor vuelva a ingresar una correcta")
except:
    print("Ingrese una opcion numerica valida")

time.sleep(.5)
borrarPantalla()
```

FUNCIONES

Si ingresamos como opción 1, nos llamará dos funciones, las cuales están definidas de la siguiente manera.

La función prodtopandlow , se encarga de encontrar los productos con mayores ventas y los que tienen menores, los ordena de mayor a menor e imprime los 10 productos top solo conformados por id, número de ventas y el nombre del producto iniciamos definiendo dos listas temporales.

```
def func_prodtopandlow():
```

```
list_temp_saletop = []
list_temp_salelow = []
```

en estas dividiremos los datos con las siguientes características en la saletop irán todos los que tienen ventas en la salelow irán todos los que no tienen ventas.

En el siguiente ciclo realizaremos el conteo de ventas por cada producto usando una variable auxiliar count_sale verificando los id de productos que aparecen en la lista lifestore_sales,se hace una comparación si count_sale es mayor de 0 quiere decir que ese producto se vendió al menos 1 vez , por lo cual se guardará en list_saletop como elemento de lista con los siguientes valores el ID de producto , el nombre del producto y la cantidad que se vendió sino se guarda en la lista de list_salelow.

```
#Inicia un ciclo para contar por cada producto cuantas veces se vendio
  for i in lifestore_products:
    count_sale = 0
    for j in lifestore_sales:
      if i[0] == j[1]:
        #print(i)
        count_sale += 1
    if count_sale>0: #si se vendio almenos 1 vez se guarda en la lista saletop
      cadena = str(i[1])
      fin = cadena.find(',')
      #print(fin)
      cadena = cadena[0:fin]
      list_temp=([i[0],cadena,count_sale])
      list_temp_saletop.append(list_temp)
    else: #sino se guarda en lista salelow
      list_temp=([i[0],i[1],count_sale])
      list_temp_salelow.append(list_temp)
```

Una serie de instrucciones para tomar en cuenta es la siguiente

```
      cadena = str(i[1])
      fin = cadena.find(',')
      cadena = cadena[0:fin]
```

En estas instrucciones se pasa a una nueva variable los datos que se encuentran en el espacio 1 de la lista actual que se está corriendo en el ciclo for , que pertenece al nombre del producto , cabe destacar que el nombre del producto incluye todas sus características separadas por “,” por lo tanto en estas instrucciones buscamos la posición dentro de la cadena donde encuentre el primer “,” de esta manera re-asignamos el valor a cadena de la siguiente manera cadema[0:fin] , esto realiza la función de copiar solamente hasta antes de la coma,

se realiza de esta manera para ahorrar espacio al momento de mostrar la impresión en consola.

Pasando con el área de impresión de la función iniciamos con simples print para colocar el título, a continuación usamos .format para asignar valores de separación entre texto

```
print()
print("====================================")
print("TOP 10  DE PRODUCTOS MÁS VENDIDOS")
print("====================================")
print()
print ("{:<3}| {:<10}| {:<5}  ".format('ID', 'N. VENTAS', 'PRODUCTO'))
print ('-------------------------------------------------------')
count_sale = 0
for v in list_temp_saletop:
    if count_sale < 10:
      id, nombre , venta = v
      print ("{:<3}| {:<10}| {:<5}".format( id, venta, nombre))
      count_sale += 1
    else:
      break
```

Usamos un ciclo for para imprimir cada uno de los productos con validación de una variable temporal count_sale la cual aumentará su valor hasta 10 (ya que solo se imprimen 10 productos) al llegar a 10 se manda a cerrar el ciclo , utilizando un orden de impresión, id , venta , nombre.Se realiza un ajuste de lista inverso con la siguiente instrucción.

```
for v in reversed(list_temp_saletop):
```

Se realiza la impresión de 5 productos menos vendidos con el mismo formato y orden, al final imprimimos la cantidad de productos que no han tenido venta alguna en color rojo utilizando la función Fore.RED+ , importante colocar un Style.Restet_All , para restablecer la configuracion de impresion.

```
print ('-------------------------------------------------------')
print(Fore.RED+"Existen",(len(list_temp_salelow)), "productos que no se están vendiendo")
print(Style.RESET_ALL)
```

La función prodtopandlow_search realiza el mismo ciclo que la función anterior pero fijándose en los parámetros de la lista lifestore_search

```
for i in lifestore_products:
  count_sale = 0
  for j in lifestore_searches:
    if i[0] == j[1]:
```

Creando la misma mecánica se imprimen 10 productos más buscados , 10 menos buscados y número de productos ignorados.
Si ingresamos como opción 2, nos llamara a la función func_Storetop().

```
def func_Storetop():
   # la función sorted nos ayuda a ordenar la lista con los parámetros de itemgetter
    lf_s_desc = sorted(lifestore_sales, key=itemgetter(2)) #ordenamos de mayor a menor tomando en
prioridad el campo de id productos
   lf_s_asc = sorted(lifestore_sales, key=itemgetter(2), reverse=True)
   score5_distinct = []
   score123_distinct = []
   top10_products = []
   top10low_products = []
```

Definiremos dos listas cada una ordenada de manera ascendente y descendente con respecto a la calificación, asi como 4 funciones temporales.

```
for i in range(len(lf_s_desc)):
      try: #uso funcion try ya que en un punto i+1 no existe dentro de la lista evita que se corte
el programa
          if lf_s_desc[i][1] != lf_s_desc[(i+1)][1] and lf_s_desc[i][2]== 5 and lf_s_desc[i][4] ==
0:
              score5_distinct.append(lf_s_desc[(i)])
      except: # cuando i+1 de la lista no existe , se agrega el último valor a la lista nueva
          score5_distinct.append(lf_s_desc[(i)])
```

Se realiza un ciclo for para toda la lista de ventas ordenadas de manera descendente, realizando una comparación de valores para evitar repetir productos queda con la siguiente instrucción en literal " el producto actual leido debe ser diferente al siguiente con respecto al id , también en el valor de score debe tener un igual a 5 y verificar que no haya sido devuelto el producto " si cumple con estos parámetros
se guarda en la lista score5_distinct , se utiliza un try y su except para evitar el error de la última verificación cuando acceden a un valor nulo de esta manera el último valor se guarda en la lista sin que se pierda información.

```
print()
   print("====================================")
   print("TOP 10 DE PRODUCTOS MEJOR CALIFICADOS ")
   print("====================================")
   print()
   print ("{:<3}| {:<2}| {:<10} ".format('ID', 'CALF', 'NOMBRE DE PRODUCTO'))
   print ("{:<3} {:<10} {:<10} ".format('----', '------------------', '-----------'))
   for i in range(10):
       for j in lifestore_products:
           if score5_distinct[i][1] == j[0]:
               cadena = str(j[1])
               fin = cadena.find(',')
               cadena = cadena[0:fin]
               list_temp_top=(j[0],cadena,score5_distinct[i][2])
```

```
            top10_products.append(list_temp_top)
  for v in top10_products:
      #print (i)
      id, nombre , calf = v
      print ("{:<3}| {:<4}| {:<10}".format( id, calf, nombre))
```

Para el método de impresión realizaremos funciones similares a la la opción 1 , guardaremos id , nombre (hasta la ‘,’) , y calificación, guardandolos en top_10productos para imprimirlos después en el orden definido.

NOTA: hay que tomar en cuenta que solo se está usando productos que se ha calificado esto no significa que todos los productos están apareciendo en esta intersección en los datos.

Para la parte de impresión de productos con la más baja calificación el único cambio en la estructura del código en la comparación de calificación sea igual a 1 o 2 o 3.

```
          if lf_s_asc[i][1] != lf_s_asc[(i+1)][1] and (lf_s_asc[i][2] == 1 or lf_s_asc[i][2] == 2
or lf_s_asc[i][2] == 3):
              score123_distinct.append(lf_s_asc[(i)])
```

Si ingresamos como opción 3, nos llamara a la función func_stadistics_sales().

Esta función nos mostrará el total de ventas anuales en $, así como el total por meses de enero a diciembre.

```
def func_stadistics_sales():
list_mm_sales=
[['ENE',0],['FEB',0],['MAR',0],['ABR',0],['MAY',0],['JUN',0],['JUL',0],['AGO',0],['SEP',0],['OCT',0]
,['NOV',0],['DIC',0]]
```

Creamos una lista de ayuda donde se almacenarán doce valores correspondientes a cada mes lis_mm_sales.

```
 for i in lifestore_sales:
   mes = str(i[3])
   fin = mes.rfind('/')
   ini= mes.find('/')+ 1
   mes = mes[ini:fin]
   mes = int(mes)
   for j in lifestore_products:
     if i[1] == j[0]:
       for k in range(12):
         if (mes-1) == (k):
           list_mm_sales[k][1]= (int(j[2]+list_mm_sales[k][1]))
```

La función inicia con un ciclo for iterando los valores de lifestore_sales, dentro de este ciclo vamos a extraer el mes

de la venta solo el mes **ya que se maneja solo el año 2020 para las ventas**(en la lista de ventas solo existe una venta en 2019, por lo cual se decide solo usar 2020 , lo que ahorra un método dentro de la función), lo extraemos de la posición 3 del objeto que estamos iterando en este caso sería ejemplo : 08/03/2020 , en este parámetro vamos a buscar el primer '/' y guardar la posición como ini , se buscará el último '/' con rfind y guardaremos la posición como fin, de esta manera extraemos solo el mes del string , una vez obtenido lo volveremos entero colocando la siguiente sintaxis mes = int(mes), dentro del ciclo crearemos otro ciclo en el que iteramos lifestore_products , en este buscaremos obtener el valor del producto de la venta que se realizó comparando los valores ID en cada lista, si la comparación es verdadera entraremos a un nuevo ciclo en el cual sumaremos el valor del producto al respectivo mes de la lista list_mm_sales dependiendo que mes sea se sumará en la posición correcta.

```
print("====================================")
 print("REPORTE VENTAS MENSUALES DEL AÑO 2020")
 print("====================================")
 print()
 print ('ENE : $', list_mm_sales[0],
 '\nFEB : $', list_mm_sales[1],
 '\nMAR : $', list_mm_sales[2],
 '\nABR : $', list_mm_sales[3],
 '\nMAY : $', list_mm_sales[4],
 '\nJUN : $', list_mm_sales[5],
 '\nJUL : $', list_mm_sales[6],
 '\nAGO : $', list_mm_sales[7],
 '\nSEP : $', list_mm_sales[8],
 '\nOCT : $', list_mm_sales[9],
 '\nNOV : $', list_mm_sales[10],
 '\nDIC : $', list_mm_sales[11])
```

en la impresión de datos de manera más rústica imprimimos valor por valor anticipado por el mes al que corresponde.
Realizamos un ciclo para sumar todos los valores de los meses para sacar el total anual y lo mandamos a imprimir.

```
print (Fore.BLUE+'-------------------------------------------')
 for i in list_mm_sales:
   tot_a = tot_a + int(i[1])

 print("LA VENTA ANUAL ES DE $",tot_a, "DEL AÑO 2020")
 print ('-------------------------------------------')
 list_mm_sales = sorted(list_mm_sales, key=itemgetter(1), reverse=True)
```

Para imprimir los meses con mas ventas y menos ventas se ordena la lista con la función sorted a base de los valores de costo.

```
print("LOS MESES CON MAYOR VENTA SON ")
print(list_mm_sales[0][0],"  $",list_mm_sales[0][1])
print(list_mm_sales[1][0],"  $",list_mm_sales[1][1])
print ('-------------------------------------------')
print("LOS MESES CON MENOR VENTA SON ")
print(list_mm_sales[-1][0],"  $",list_mm_sales[-1][1])
print(list_mm_sales[-2][0],"  $",list_mm_sales[-2][1])
print ('-------------------------------------------')
print(Style.RESET_ALL)
```

Una vez ordenada se imprimen los dos primeros lugares y los últimos lugares para acceder a los últimos lugares de la lista se coloca -1 y -2 respectivamente en el espacio a leer de la lista [] .

LIBRERÍAS

```
import time
import os
import getpass
from operator import itemgetter, attrgetter
from colorama import init, Fore, Back, Style
"""
```

- Time para el uso de segundos entre líneas de código ejecutadas realizando pausas.
- Os para reconocer el tipo de sistema operativo que se está utilizando esto solo para la función de borrar pantalla.
- Getpass es la librería utilizada para ocultar lo tecleado en la contraseña.
- Operator es la librería que incluye los métodos itemgetter para organizar las tablas según el parámetro que desees ocupar.
- Colorama esta librería es utilizada para cambiar colores y estilos en la impresión de texto dentro de consola.

SOLUCIÓN AL PROBLEMA

1 Productos más vendidos y productos rezagados a partir del análisis de número de ventas y búsquedas.

```
====================================
TOP 10  DE PRODUCTOS MAS VENDIDOS
====================================

ID | N. VENTAS | PRODUCTO
-----------------------------------------------------------
54 | 50        | SSD Kingston A400
3  | 42        | Procesador AMD Ryzen 5 2600
5  | 20        | Procesador Intel Core i3-9100F
42 | 18        | Tarjeta Madre ASRock Micro ATX B450M Steel Legend
57 | 15        | SSD Adata Ultimate SU800
29 | 14        | Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING
4  | 13        | Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8
2  | 13        | Procesador AMD Ryzen 5 3600
47 | 11        | SSD XPG SX8200 Pro
48 | 9         | SSD Kingston A2000 NVMe
-----------------------------------------------------------

====================================
TOP 10 DE PRODUCTOS MENOS VENDIDOS
====================================

ID | N. VENTAS | PRODUCTO
-----------------------------------------------------------
10 | 1         | MSI GeForce 210
13 | 1         | Tarjeta de Video Asus NVIDIA GeForce GTX 1050 Ti Phoenix
17 | 1         | Tarjeta de Video Gigabyte AMD Radeon R7 370 OC
22 | 1         | Tarjeta de Video MSI NVIDIA GeForce GTX 1050 Ti OC
28 | 1         | Tarjeta de Video Zotac NVIDIA GeForce GTX 1660 Ti
40 | 1         | Tarjeta Madre Gigabyte XL-ATX TRX40 Designare
45 | 1         | Tarjeta Madre ASRock ATX H110 Pro BTC+
46 | 1         | Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2
50 | 1         | SSD Crucial MX500
60 | 1         | Kit Memoria RAM Corsair Dominator Platinum DDR4
-----------------------------------------------------------
Existen 54 productos que no se estan vendiendo
```

Hay que tener en cuenta estos 54 productos que no se están vendiendo.

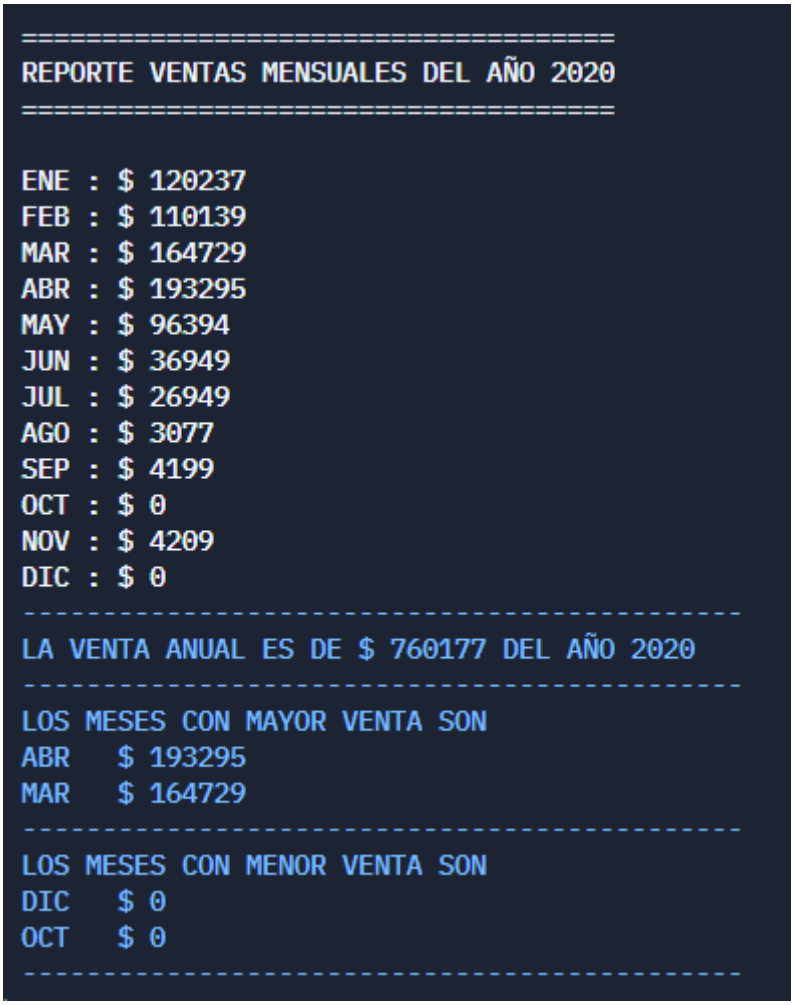
```
====================================
TOP 10  DE PRODUCTOS MAS BUSCADOS
====================================

ID | BUSQUEDAS | PRODUCTO
-----------------------------------------------------------
54 | 263       | SSD Kingston A400
57 | 107       | SSD Adata Ultimate SU800
29 | 60        | Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING
3  | 55        | Procesador AMD Ryzen 5 2600
4  | 41        | Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8
85 | 35        | Logitech Audífonos Gamer G635 7.1
67 | 32        | TV Monitor LED 24TL520S-PU 24
7  | 31        | Procesador Intel Core i7-9700K
47 | 30        | SSD XPG SX8200 Pro
5  | 30        | Procesador Intel Core i3-9100F
-----------------------------------------------------------

====================================
TOP 10 DE PRODUCTOS MENOS BUSCADOS
====================================

ID | BUSQUEDAS | PRODUCTO
-----------------------------------------------------------
9  | 1         | Procesador Intel Core i3-8100
10 | 1         | MSI GeForce 210
27 | 1         | Tarjeta de Video VisionTek AMD Radeon HD5450
35 | 1         | Tarjeta Madre Gigabyte micro ATX Z390 M GAMING
45 | 1         | Tarjeta Madre ASRock ATX H110 Pro BTC+
59 | 1         | SSD Samsung 860 EVO
70 | 1         | Samsung Smart TV LED 43
80 | 1         | Ghia Bocina Portátil BX800
93 | 1         | Ginga Audífonos con Micrófono GI18ADJ01BT-RO
13 | 2         | Tarjeta de Video Asus NVIDIA GeForce GTX 1050 Ti Phoenix
-----------------------------------------------------------
Existen 40 productos que no se estan buscando
```

2 Productos por reseña en el servicio a partir del análisis

```
====================================
TOP 10 DE PRODUCTOS MEJOR CALIFCADOS
====================================

ID | CALF| NOMBRE DE PRODUCTO
---- ----------------- -----------
1  | 5   | Procesador AMD Ryzen 3 3300X S-AM4
2  | 5   | Procesador AMD Ryzen 5 3600
3  | 5   | Procesador AMD Ryzen 5 2600
4  | 5   | Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8
5  | 5   | Procesador Intel Core i3-9100F
6  | 5   | Procesador Intel Core i9-9900K
7  | 5   | Procesador Intel Core i7-9700K
8  | 5   | Procesador Intel Core i5-9600K
11 | 5   | Tarjeta de Video ASUS AMD Radeon RX 570
12 | 5   | Tarjeta de Video ASUS NVIDIA GeForce GTX 1660 SUPER EVO OC

====================================
TOP 5 DE PRODUCTOS PEOR CALIFCADOS
====================================

ID | CALF| NOMBRE DE PRODUCTO
---- ----------------- -----------
31 | 1   | Tarjeta Madre AORUS micro ATX B450 AORUS M (rev. 1.0)
17 | 1   | Tarjeta de Video Gigabyte AMD Radeon R7 370 OC
46 | 2   | Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2
89 | 3   | Cougar Audífonos Gamer Phontum Essential
31 | 3   | Tarjeta Madre AORUS micro ATX B450 AORUS M (rev. 1.0)
```

Este análisis se basa solo en los productos que han sido vendidos , recordemos que hay 54 que no tienen una venta como tal.

3 sugerir una estrategia de productos a retirar del mercado así como sugerencia de como reducir la acumulación de inventario considerando los datos de ingresos y ventas mensuales.



La estrategia de venta es un poco compleja debemos analizar los componentes que lleven relación (los que no se están vendiendo con los que se venden más) , creando promociones bajo la siguiente base:

Llamemos producto principal al que más se vende por ejemplo el sdd kingston a400 , y llamemos producto complementario a alguno de los que no se venden , ejemplo un par de memorias ram ddr3 o ddr4, en función a esto crear lo siguiente.

“Dale una nueva vida a tu laptop” - SSD Kingston 400 + 2 Memorias ram DDR3

El costo se va a basar en los siguientes parámetros

Costo de producto principal + 1.15% valor real del producto complementario.

Esto quiere decir la ganancia será del 15% en el producto rezagado que no se está vendiendo, de esta manera podemos aprovechar la popularidad de los productos más vendidos sin afectar el costo y ganancia que se está obteniendo y sacando del almacén productos que se están quedando recuperando su valor y ganando un porcentaje.
Mostrando siempre los costos de los productos , de esta manera el cliente verá que hay una buena promoción animandose a obtener esta misma con un producto que desean tener.
Con respecto a las fechas estas promociones deben ser ligadas a los meses donde la venta ha sido menor con un porcentaje menor de 20% en el producto complementario , y en fechas de alta venta poder colocar promociones con un rango de más de 50%.
Como ventaja del análisis también dejar de adquirir productos que a pesar de promociones no se están vendiendo y llenar el stock de los productos más populares, siguiendo una regla básica vendo 3 compró 5 siempre y cuando se sigan vendiendo.

CONCLUSIÓN

El problema que tiene la tienda no se puede resolver solo con los parámetros que realizamos en el programa de python , al menos no al 100% de eficacia.
Existen más parámetros a conocer para crear una estrategia con respecto a las ventas , hay valores que no ocupamos que están dentro de la base de datos de la tienda los cuales deben ser útiles para crear una formulación de cómo y cuándo comprar los elementos de la tienda al proveedor , ya que la prioridad de toda tienda es obtener ganancias no solo vender por vender.
Sin embargo es un tema puesto de manera no real , con el puedo observar que la información que nos ofrece puede servir para muchas tomas de decisiones y complementandose con más información podríamos realmente afectar el movimiento a favor de una tienda, persona, institución , etc para obtener su meta con los análisis correctos y la recopilación de la información alrededor del ente que la solicita.
Cabe destacar que python es una de las herramientas que mejor se puede ajustar hoy en día por su sencillez de sintaxis con respecto a lenguajes más viejos , su potencia de usarse en multiplataformas y la velocidad con la que se procesa.