03_opt_multi_periodo

November 11, 2025

1 03– Optimización multi-período

Objetivo

Optimizar la política de pedidos considerando varios períodos (semanas o meses) y restricciones de inventario, usando Programación Lineal (PL) con PuLP.

[3]:
```
import importlib
import subprocess
import sys

def install_and_import(package, import_name=None):
    """Instala un paquete si no está presente."""
    import_name = import_name or package
    try:
        importlib.import_module(import_name)
        print(f"{package} ya está instalado.")
    except ImportError:
        print(f" Instalando {package}...")
        subprocess.check_call([sys.executable, "-m", "pip", "install", package])
        print(f"{package} instalado correctamente.")

# ------------------------------------------------------------
# Librerías requeridas por los notebooks prescriptivos
# ------------------------------------------------------------
dependencies = {
    "pandas": "pandas",
    "numpy": "numpy",
    "pulp": "pulp",
    "statsmodels": "statsmodels",
    "matplotlib": "matplotlib",
    "seaborn": "seaborn",
    "plotly": "plotly"
}

for pkg, import_name in dependencies.items():
    install_and_import(pkg, import_name)
```

```
print("\n Todas las dependencias están listas.\n")
```

```
pandas ya está instalado.
numpy ya está instalado.
pulp ya está instalado.
statsmodels ya está instalado.
matplotlib ya está instalado.
seaborn ya está instalado.
plotly ya está instalado.

 Todas las dependencias están listas.
```

[4]:
```
# 1 Importar librerías
import pandas as pd
from pulp import LpMinimize, LpProblem, LpVariable, lpSum, value
```

[5]:
```
# 2 Cargar datos base
ventas = pd.read_csv("resultados_modelo_sQ.csv")
print(" Datos cargados:", ventas.shape)
```

```
 Datos cargados: (3000, 24)
```

[6]:
```
# 3 Definir parámetros de optimización
periodos = 12
productos = ventas['sku'].unique()

costo_pedido = 50
costo_mant = 0.25
demanda = ventas.groupby('sku')['demanda_promedio'].mean()
```

[7]:
```
# 4 Crear modelo
modelo = LpProblem("Optimización_Multi_Periodo", LpMinimize)

# Variables: pedidos y stock
pedidos = LpVariable.dicts("pedido", (productos, range(periodos)), lowBound=0)
stock = LpVariable.dicts("stock", (productos, range(periodos)), lowBound=0)
```

[8]:
```
# 5 Función objetivo: minimizar costo total
# Parámetro grande (capacidad máxima de pedido, sirve para vincular binaria)
M = 10000

# Variable binaria: indica si se realiza pedido
y = LpVariable.dicts("pedido_activo", (productos, range(periodos)),␣
 ↪cat="Binary")

# Función objetivo: costo total = costo_pedido * y + costo_mant * stock
modelo += lpSum([
```

```
    costo_pedido * y[p][t] + costo_mant * stock[p][t]
    for p in productos for t in range(periodos)
])

# Restricción de activación: si no hay pedido, cantidad = 0
for p in productos:
    for t in range(periodos):
        modelo += pedidos[p][t] <= M * y[p][t]
```

[9]:
```
!pip install highspy
```

```
Requirement already satisfied: highspy in /opt/conda/lib/python3.11/site-
packages (1.12.0)
Requirement already satisfied: numpy in /opt/conda/lib/python3.11/site-packages
(from highspy) (1.24.4)
```

[10]:
```
# ======================================================
# 6 Restricciones: balance de inventario
# ======================================================
from pulp import LpVariable, LpProblem, LpMinimize, value, PULP_CBC_CMD,␣
 ↪LpStatus
import pandas as pd, time, sys, threading
from datetime import datetime

# --- Verificación y creación de variables base ---
if 'productos' not in locals():
    if 'df' in locals() and 'sku' in df.columns:
        productos = list(df['sku'].unique())
        print(f"Productos detectados: {len(productos)}")
    else:
        raise ValueError("No se encontró la variable 'productos'. Asegúrate de␣
 ↪definirla o cargar 'df' con columna 'sku'.")

if 'periodos' not in locals():
    periodos = 12  # valor por defecto
    print("Periodo no definido, usando 12 por defecto.")

# --- Verificación de demanda promedio ---
if 'demanda' not in locals():
    if 'demanda_promedio' in df.columns:
        demanda = {p: df.loc[df['sku'] == p, 'demanda_promedio'].mean() for p␣
 ↪in productos}
    else:
        raise ValueError("No existe columna 'demanda_promedio' en 'df'.")

# --- Crear modelo y variables si no existen ---
if 'modelo' not in locals():
```

```
    modelo = LpProblem("Optimizacion_MultiPeriodo", LpMinimize)

if 'pedidos' not in locals():
    pedidos = LpVariable.dicts("pedido", (productos, range(periodos)),␣
 ↪lowBound=0, cat="Continuous")
if 'stock' not in locals():
    stock = LpVariable.dicts("stock", (productos, range(periodos)), lowBound=0,␣
 ↪cat="Continuous")

# --- Restricciones ---
for p in productos:
    for t in range(periodos):
        if t == 0:
            modelo += stock[p][t] == pedidos[p][t] - demanda[p]
        else:
            modelo += stock[p][t] == stock[p][t-1] + pedidos[p][t] - demanda[p]

# ======================================================
# 7 Resolver con progreso
# ======================================================
print(f"\nInicio del proceso: {datetime.now()}")

solver = PULP_CBC_CMD(msg=True, timeLimit=600, threads=6)

# --- Barra de progreso en paralelo ---
def barra_progreso(total_tiempo=600, intervalo=5):
    pasos = total_tiempo // intervalo
    for i in range(pasos):
        time.sleep(intervalo)
        progreso = (i + 1) / pasos * 100
        sys.stdout.write(f"\rProgreso estimado: {progreso:5.1f}%")
        sys.stdout.flush()
    print()

progreso_thread = threading.Thread(target=barra_progreso, args=(600, 5))
progreso_thread.start()

inicio = datetime.now()
status = modelo.solve(solver)
fin = datetime.now()

progreso_thread.join()

print(f"\nEstado final del modelo: {LpStatus[status]}")
print(f"Duración total: {fin - inicio}")

# ======================================================
```

```python
# 8 Exportar resultados
# ====================================================
resultados = [
    {"sku": p, "periodo": t + 1, "pedido": value(pedidos[p][t]), "stock":␣
 ↪value(stock[p][t])}
    for p in productos for t in range(periodos)
]

df_result = pd.DataFrame(resultados)
df_result.to_csv("resultados_multi_periodo.csv", index=False)
print("\nArchivo 'resultados_multi_periodo.csv' generado correctamente.")
```

```
Inicio del proceso: 2025-11-11 03:51:38.870277
Progreso estimado: 100.0%

Estado final del modelo: Optimal
Duración total: 0:10:05.251810

Archivo 'resultados_multi_periodo.csv' generado correctamente.
```

[ ]: