Documentación técnica de la clase Portafolio

Alejandro A. Estrada Franco

8 de enero de 2026

Índice

1. Introducción

La clase *Portafolio* consiste en un módulo programado en python cuya funcionalidad es rebalancear un portafolio que contiene ciertos activos al cambiar los precios de estos.

El módulo esta pensado para integrarse en arquitecturas más amplias diseñadas para resolver o automatizar procesos relacionados con la valuación de portafolios.

Esta documentación se compone de dos secciones principales: *Metodología* y *Funcionamiento*. En *Metodología* se hace un análisis teórico del problema planteado, así que su redacción contiene términos especializados de finanzas cuantitativas. En *Funcionamiento* se describe la operación de la clase.

2. Metodología

De acuerdo al planteamiento del problema, suponemos un portafolio con n activos $X_1, \ldots, X_n$ el precio al tiempo t del activo X_k se denota como S_k^t, en t para cada activo X_k tenemos una cantidad de unidades de este denotada como ϕ_k^t, de tal manera que el valor invertido en el activo X_k al tiempo t es el número $\phi_k^t S_k^t$, es decir el producto entre el precio del activo y la cantidad de unidades de este.

Así el valor inicial total invertido en el portafolio está dado como:

$$V_t = \sum_{k=1}^{n} \phi_k^t S_k^t.$$

Así el porcentaje de participación del activo X_k en el valor total del portafolio se calcula:

$$p_k = \frac{\phi_k^t S_k^t}{V_t}. \tag{1}$$

Ahora supongamos que en el tiempo $t+1$ tenemos una nueva cotización de precios para los activos $S_1^{t+1}, \ldots, S_n^{t+1}$; ahora el portafolio tiene un valor total

$$V_{t+1} = \sum_{k=1}^{n} \phi_k^t S_k^{t+1}$$

. Buscamos que la distribución de unidades de activos este dominada en cada tiempo por los pesos $\mathbf{p}_1, \ldots, \mathbf{p}_n$, donde cada $\mathbf{p_k} \in [0,1]$ y $\mathbf{p}_1 + \cdots + \mathbf{p}_n = 1$, dicho de otra manera queremos que el porcentaje de participación del activo X_k en el valor total del portafolio sea el número $100 \cdot \mathbf{p}_k$.

Así vamos a calcular ϕ_k^{t+1} para cada k despejando de la ecuación (1) ajustada al tiempo $t+1$:

$$\phi_k^{t+1} = \frac{\mathbf{p}_k V_{t+1}}{S_k^{t+1}}.$$

El algoritmo para rebalancear se estructura de la siguiente manera:

Algoritmo: Rebalancear portafolio

Require: Precios $S_1^{t+1}, \ldots, S_n^{t+1}$ en $\mathbf{t+1}$, porcentajes objetivo $\mathbf{p}_1, \ldots, \mathbf{p}_n$

Se calcula el valor de portafolio al tiempo $t+1$:

$$V_{t+1} \leftarrow \sum_{k=1}^{n} \phi_k^t S_k^{t+1}$$

for $k = 1$ to n **do**

Se calculan los pesos para ajustar a la distribución objetivo:

$$\phi_k^{t+1} \leftarrow V_{t+1} \cdot \mathbf{p}_k$$

Se obtienen los movimientos (compra o venta):

$$\text{Si} \quad \phi_k^{t+1} - \phi_k^t < 0 \quad m_k = \textit{vender;} \quad \text{en otro caso:} \quad m_k = \textit{comprar}$$

Unidades a negociar:

$$u_k = |\phi_k^{t+1} - \phi_k^t|$$

end for

return $\{X_1 : \{m_1 : u_1\}, \ldots, X_n : \{m_n : u_n\}\}$

3. Funcionamiento

3.1. Requisitos

- Python 3.9 o superior
- Librerías estándar (no se requieren dependencias externas)

3.2. Descripción

La clase puede ser inicializada sin pasar ningún argumento, en ese caso se carga por defecto una lista de activos.

Para cargar una lista de activos se debe pasar como argumento una lista, cada elemento de la lista es una lista que representa al activo, esta última debe tener cuatro entradas en el siguiente orden: *nombre, unidades, precio y peso objetivo.*

```
1 Portafolio_1 = Portafolio(stocks=[('AAPL', 50, 15.0, 0.4), ('GOOGL',
    80, 25.0, 0.4), ('MSFT', 120, 35.0, 0.1),('AMZN', 60, 45.0, 0.1)])
```

La suma de las entradas de peso objetivo de todos los activos pasados debe ser igual a 1. Si no es así se lanzará una excepción con el siguiente mensaje:

La suma de los pesos debe ser igual a 1.

Para actualizar los precios de los activos se debe usar el método *currentPrice*, como resultado de la aplicación de este método se actualizan los precios de los activos de la instancia.

```
1 Portafolio_1.currentPrice([12.0, 18.0, 33.0, 50.0])
```

El método *currentPrice* debe recibir un argumento, este debe ser una lista con los precios actualizados de los activos. El número de entradas de la lista debe de coincidir con el número de activos en el portafolio (determinado por la lista de activos con la que se construyó la clase), si esto no es así se lanza una excepción con el mensaje

Ingresar una lista de precios con la misma longitud que la lista de acciones.

Para rebalancear el portafolio se usa el método *rebalance* se sugiere primero actualizar los precios; este método no acepta argumentos, como resultado de su aplicación se actualizan las unidades de los activos de la instancia y regresa un diccionario que informa el rebalance.

```
1 movimientos = Portafolio_1.rebalance()
2 print(reporte)
3 out:
4 {'AAPL': {'accion': 'comprar', 'cantidad': 250}, 'GOOGL': {'accion': '
    comprar', 'cantidad': 120}, 'MSFT': {'accion': 'vender', 'cantidad':
     93}, 'AMZN': {'accion': 'vender', 'cantidad': 42}}
```