Manual de usuario Tracking Bursátil TB500

Introducción:

El presente documento tiene como objetivo proporcionar instrucciones claras y concisas para utilizar la solución **Tracking Bursátil TB500**, la cual es una API habilitada bajo el endpoint **http://ec2-18-217-18-255.us-east-2.compute.amazonaws.com:5000/api**. Específicamente se proporcionan las instrucciones para consumir el API desde un notebook de Jupyter. El API devuelve datos en formato JSON, los cuales deben convertirse a un formato de tabla (dataframe) para su posterior análisis y procesamiento.

Este manual es una sugerencia de uso del API, ya que como tal puede consumirse de diferentes maneras y desde diferentes aplicativos como Excel, Power BI o aplicaciones web particulares.

En específico se explican los siguientes puntos:

- Requisitos previos
- Parámetros y salida del API
- Pasos para utilizar el API

Requisitos previos:

- Tener instalado Python en su sistema.
- Tener instalado Jupyter Notebook.
- Conexión a Internet para acceder al API.
- Experiencia en el uso y consumo de datos desde notebooks de Python.

Parámetros y salida del API:

A continuación, se listan y explican los parámetros que posee el API:

No.	Nombre	Descripción	Posibles valores	Obligatorio	Valor por defecto (si no se proporciona)
1	url	URL del API	http://ec2-18-217-18-255.us-east-2.compute.amazonaws.com:5000/api	Sí	N/A
2	json	Histórico de precio de cierre ajustado por ticker	Datos en formato JSON	No	Se extrae la data desde el lago de datos
3	ticker	Listado de tickers a evaluar	Tickers separados por coma	No	Todos los tickers disponibles en el lago de datos
4	ini_date	Fecha de inicio de muestra de datos	Fecha en formato 'YYYY-MM-DD'	No	Se toma toda la data disponible en el lago de datos

5	end_date	Fecha de fin de muestra de datos	Fecha en formato 'YYYY-MM-DD'	No	Se toma toda la data disponible en el lago de datos
6	output_part	Detalle para mostrar en la salida	0: Sólo portfolio alloation 1: Sólo métricas	No	Todo el detalle (portfolio allocation + métricas + mensajes informativos)

Ejemplo:

```
# Endpoint
url = 'http://ec2-18-217-18-255.us-east-2.compute.amazonaws.com:5000/api'

# Data and parameters
json_input = data.to_json()
json_input = json.loads(json_input)

ticker = 'NVDA,META'
ini_date = '2022-11-23'
end_date = '2023-05-23'
output_part = '0'

params = {
    'ticker': ticker,
    'ini_date': ini_date,
    'end_date': end_date,
    'output_part': output_part
}

# API consumption
response = requests.get(url, json=json_input, params=params)
print(json.dumps(json.loads(response.json()), indent=2))
```

Una vez se hace una solicitud al API, y si los parámetros son correctos, devuelve la siguiente salida:


```
{
  "ticker": [
    "NVDA",
    "META"
  ],
  "weights": [
    0.5867553538459289,
    0.4132446461540711
  ],
  "return": 0.22494875767506256,
  "volatility": 0.38776773644266416,
```

```
  "sharpe_ratio": 0.5801121045776426,
  "input_data": "User data input",
  "data_range": "N/A",
  "message": "Optimization terminated successfully",
  "success": "True"
}
```


Los elementos de la salida se interpretan de la siguiente manera:

No.	Nombre	Descripción
1	ticker	Listado de los activos (tickers) evaluados
2	weights	Distribución del portafolio en porcentaje para cada uno de los activos (tickers) proporcionados
3	return	Retorno del portafolio
4	volatility	Volatilidad del portafolio
5	sharpe_ratio	Sharpe ratio
6	input_data	Data usada para la optimización: si no se envía como un parámetro, el API consume la data desde el lago de datos
7	data_range	En caso la data se consuma desde el lago de datos, es la fecha inicial de la muestra; si se proporciona data externa (como parámetro), no aplica
8	message	En caso la data se consuma desde el lago de datos, es la fecha final de la muestra; si se proporciona data externa (como parámetro), no aplica
9	success	Booleano que indica si la optimización se dio exitosamente o no

Pasos para utilizar el API desde Jupyter Notebook:

Paso 1: Importar las librerías necesarias

Antes de comenzar, hay que asegurarse de tener instaladas las siguientes librerías de Python, las cuales son esenciales para realizar solicitudes al API y convertir los datos JSON en un dataframe.

```
# Import libraries
import numpy as np
import pandas as pd
import json
import requests
```

Paso 2: Extracción de datos a utilizar

En este documento se coloca como ejemplo el consumo de los datos disponibles en el lago de datos que hace parte del prototipo, sin embargo, los datos a inyectar al API pueden provenir de cualquier otra fuente, siempre y cuando se cumpla con la estructura requerida.

```
# Data Lake params
gold = './data/gold/portfolio-optimization/'
gold_table = 'portfolio_optimization.csv'
```

```
# Read data from data lake
data = pd.read_csv(gold+gold_table)
data['Date'] = pd.to_datetime(data['Date'])
```

Paso 3: Conversión de datos insumo a formato JSON

En caso de que se necesite proporcionar datos al API (como se explicó en la sección de parámetros, si no se proporcionan, el API los consume desde el lago de datos) éstos deben llevarse a formato JSON.

```
# Convert dataframe to JSON
json_input = data.to_json()
json_input = json.loads(json_input)
```

Paso 4: Definición de parámetros

Dependiendo de lo que se necesite hacer y solicitar al API, se deben definir los parámetros de la siguiente manera. Los parámetros que no se necesiten se deben excluir del diccionario que se envía al API.

```
# Parámetros
url = 'http://ec2-18-217-18-255.us-east-2.compute.amazonaws.com:5000/api'
ticker = 'NVDA,META'
ini_date = '2022-11-23'
end_date = '2023-05-23'
output_part = '0'

params = {
    'ticker': ticker,
    'ini_date': ini_date,
    'end_date': end_date,
    'output_part': output_part
}
```

Paso 5: Realizar la solicitud al API

Para realizar una solicitud al API con la URL correspondiente, se debe utilizar la función requests.get()

```
# Send the GET request
response = requests.get(url, params=params)
```

Paso 6: Verificar la respuesta del API

Si la solicitud fue exitosa, se pueden imprimir los resultados.

```
# Print the response
print(json.dumps(json.loads(response.json()), indent=2))
```

Paso 7: Convertir los datos JSON a un dataframe

Para extraer los datos JSON de la respuesta y convertirlos en un dataframe se puede utilizar la función pd.DataFrame().

```
# Data tabulation
pd.read_json(response.json())
```

Paso 8: Explorar y analizar los datos

Una vez convertidos los datos en un dataframe, se pueden explorar y realizar análisis adicionales según sea necesario. A continuación, se muestran ejemplos de las salidas en formato JSON y convertida a tabla (dataframe).

JSON:


```
{
  "ticker": [
    "AAPL",
    "MSFT",
    "AMZN",
    "TSLA",
    "GOOGL",
    "GOOG",
    "NVDA",
    "BRK-B",
    "META",
    "UNH"
  ],
  "weights": [
    0.5864106929510846,
    0.0,
    0.0036080595718128742,
    0.16129142517742895,
    3.415236843329339e-18,
    0.1127960556044357,
    2.1412992906588713e-17,
    1.22514845490862e-17,
    1.3633842319005218e-17,
    0.13589376669523792
  ],
  "return": 0.2950527633261574,
  "volatility": 0.27998613675838224,
  "sharpe_ratio": 1.0538120449184134,
  "input_data": "Data Lake",
  "data_range": "2003-05-21 00:00:00 - 2023-05-19 00:00:00",
  "message": "Optimization terminated successfully",
  "success": "True"
}
```


Dataframe:

	ticker	weights
0	NVDA	0.363554
1	META	0.636446