PROYECTOS DE NUMPY

Objetivo del Documento

El presente documento tiene como objetivo explorar la aplicación de la biblioteca NumPy en el análisis estadístico, complementando así los estudios previamente desarrollados mediante pandas y SQL. Esta integración busca aportar una nueva perspectiva al tratamiento y evaluación de los datos, utilizando herramientas estadísticas que permiten una mayor profundidad y precisión en los análisis.

Inicialmente, el enfoque estará centrado en ejemplos y casos ajenos al trading algorítmico, con el propósito de consolidar los fundamentos teóricos y prácticos del uso de NumPy en análisis de datos. Posteriormente, se abordarán posibles aplicaciones dentro del contexto del funcionamiento de un bot de trading, evaluando cómo esta herramienta puede contribuir al desarrollo y optimización de estrategias automatizadas.

Introducción a NumPy

La presente sección de este documento no es un proyecto en sí, este tiene como finalidad analizar y aplicar las principales funciones de la biblioteca NumPy en el ámbito del análisis financiero cuantitativo y el desarrollo de estrategias de trading algorítmico. A través de una aproximación práctica y teórica, se busca no solo comprender el funcionamiento interno de las operaciones vectorizadas y matriciales que ofrece NumPy, sino también su aplicabilidad en contextos reales de mercado.

Para ello, se toma como base conceptual la obra "Guía para principiantes", la cual introduce de forma progresiva los fundamentos necesarios para abordar problemáticas relacionadas con la manipulación de datos financieros, la generación de señales de trading y la optimización de procesos computacionales en entornos Python. La propuesta contempla una revisión crítica y una implementación práctica de los contenidos tratados en dicha bibliografía, con el fin de consolidar conocimientos técnicos y fortalecer competencias analíticas en el uso de herramientas computacionales para las finanzas.

Se diferencia esta sección ya que en una primera instancia, encuentro relevante ver las utilidades que posee numpy para el análisis financiero y trading algorítmico, ya que en una primera instancia esta puede ser difícil de ver de manera específica y no enlatada en otro proyecto.

1.1 Tipos de aplicaciones de NumPy

Para la obtención de los datos necesarios en el análisis, se empleará la funcionalidad de descarga provista por la biblioteca yfinance, tal como se implementa en el proyecto "OHLCV_analisis" desarrollado en el entorno de trabajo con pandas. Esta herramienta permite acceder a información histórica de mercado —incluyendo precios de apertura, máximo, mínimo, cierre, y volumen (OHLCV)— correspondiente a los activos financieros (tickers) seleccionados. Dichos datos constituyen la base sobre la cual se aplicarán las funciones de NumPy, permitiendo explorar sus capacidades en el procesamiento, análisis y transformación de series temporales financieras.

Desde una perspectiva metodológica, las funciones utilizadas en esta sección se encuentran estrechamente vinculadas con los fundamentos de la estadística, materia central en el análisis de datos financieros. Su aplicación resulta particularmente relevante en el contexto del trading algorítmico, dado que muchos sistemas automatizados de inversión —como los bots de trading— toman decisiones basadas en el análisis estadístico de series temporales de precios. A través de parámetros como la media, la varianza o la volatilidad, es posible identificar patrones de comportamiento que permiten modelar escenarios de mercado y optimizar estrategias de entrada y salida.

A modo de ejemplo, a continuación se presenta un fragmento de código que ilustra la aplicación de funciones estadísticas de NumPy sobre los precios de cierre de un activo financiero:

```
t = np.arange(len(data['Close']))
print("Promedio ponderado: ",np.average(data['Close'], weights=t))
print("Promedio: ",np.mean(data['Close']))
print("Mínimo: ",np.min(data['Close']))
print("Maximo: ",np.max(data['Close']))
print("Spread: ",np.ptp(data['Close']))
print("Mediana: ",np.median(data['Close']))
print("Varianza: ",np.var(data['Close']))

retornos = np.diff(data['Close']) / data['Close'][:-1]
print("Desviación estándar: ", np.std(retornos))

volatilidad = np.std(np.diff(np.log(data['Close']))) / np.mean(np.diff(np.log(data['Close'])))
volatilidad = volatilidad / np.sqrt(1./60.)
print("Volatilidad: ",volatilidad)
```

En este bloque, se calcula un conjunto de métricas relevantes para el análisis financiero:

- Promedio ponderado y promedio simple, útiles para identificar precios centrales y tendencias.
- Valores extremos como el mínimo, máximo y el spread, que dan cuenta de la amplitud de los movimientos del activo.
- Mediana y varianza, que ayudan a interpretar la dispersión de los datos.
- Retornos y desviación estándar, fundamentales para evaluar el riesgo asociado a las variaciones de precio.
- Volatilidad logarítmica anualizada, una medida clave en la evaluación del comportamiento del activo frente a escenarios inciertos.

El uso de estas métricas permite dotar al bot de trading de criterios cuantificables sobre los cuales basar sus decisiones, transformando datos en señales operativas concretas. Así, la integración de herramientas estadísticas a través de NumPy representa un puente entre la teoría financiera y su aplicación computacional.

1.2 ¿Por qué la estadística resulta tan útil en los bots de trading?

La estadística es una herramienta esencial en el desarrollo de bots de trading, ya que permite interpretar y dar sentido a los datos financieros que se generan en los mercados en tiempo real. Los precios de los activos no siguen patrones determinísticos; por el contrario, están

sujetos a la incertidumbre y al ruido aleatorio. Frente a este entorno dinámico, la estadística proporciona un marco para identificar regularidades, estimar comportamientos futuros y tomar decisiones informadas.

Uno de los principales usos de la estadística en trading algorítmico es el análisis de series temporales, es decir, el estudio de la evolución de los precios a lo largo del tiempo. A través de medidas como la media móvil, la mediana, la desviación estándar o la regresión lineal, un bot puede detectar tendencias, zonas de consolidación o cambios bruscos en el comportamiento de un activo. Este análisis es crucial para generar señales de compra o venta basadas en criterios cuantitativos y objetivos.

Otro aspecto clave es la medición del riesgo, fundamental en cualquier estrategia financiera. Indicadores como la varianza, la desviación estándar de los retornos o la volatilidad logarítmica permiten estimar cuánto puede fluctuar el precio de un activo en determinado período. Con esta información, un bot puede ajustar el tamaño de las posiciones, establecer límites de pérdida (stop loss) y calcular métricas de rentabilidad ajustada al riesgo, como el ratio de Sharpe.

Además, la estadística permite a los bots evaluar probabilidades y escenarios, lo cual es especialmente útil en enfoques basados en simulación o aprendizaje automático. Al modelar distribuciones de precios, un bot puede estimar, por ejemplo, la probabilidad de alcanzar cierto nivel de beneficio o de sufrir una pérdida, y actuar en consecuencia. Esta capacidad de anticipación es la que convierte a la estadística en el núcleo de muchas estrategias algorítmicas.

2.2 Análisis temporal y manipulación de fechas

El análisis temporal constituye un pilar fundamental en el estudio de mercados financieros, ya que la mayoría de los datos relevantes —como precios, volumen y señales— están indexados por tiempo. Comprender y manipular correctamente estos datos es esencial para detectar patrones, evaluar rendimientos históricos y desarrollar estrategias de trading basadas en series temporales.

En el entorno de desarrollo con Python, es posible llevar a cabo distintos tipos de análisis sobre variables temporales utilizando diversas bibliotecas especializadas. Si bien existen módulos como datetime o numpy.datetime64 para el tratamiento de fechas, en la práctica, la manipulación eficiente de series temporales en el contexto financiero suele realizarse con mayor facilidad a través de la biblioteca de pandas. Esto se debe a que los pandas ofrecen estructuras de datos como DataFrame y Series, que permiten manejar fechas como índices, facilitando operaciones como resampling, rolling windows, filtrado por intervalos, cálculos de variaciones temporales, entre otras.

En el diseño de un bot de trading, es común trabajar simultáneamente con múltiples librerías (por ejemplo, numpy, pandas, matplotlib, ta, etc.). Por ello, no se busca una única herramienta

para resolver todos los problemas, sino un enfoque modular que aproveche lo mejor de cada biblioteca. En este sentido, pandas se presenta como una solución robusta y práctica para el tratamiento de datos temporales, especialmente cuando ya se está operando dentro del paradigma de análisis basado en DataFrames.

Para llevar a cabo análisis temporales, como la segmentación de datos en intervalos semanales o la detección de bloques temporales de actividad, se emplean funciones específicas que permiten agrupar y contabilizar registros en función de su continuidad temporal. Una de estas funciones, implementada mayormente con la biblioteca pandas, es la función contar_fechas.

Esta función tiene como finalidad identificar y agrupar secuencias continuas de fechas dentro de un conjunto de datos, asignando un contador incremental a cada grupo. En concreto, compara cada fecha con la anterior mediante la operación shift(1), calcula la diferencia en días entre ambas y establece un punto de reinicio del grupo cuando esa diferencia supera un umbral definido (en este caso, dos días). A partir de este criterio, se genera una columna de agrupación (group) mediante un acumulado (cumsum) y se numera cada fila dentro del grupo utilizando cumcount().

Este enfoque permite, por ejemplo, segmentar bloques temporales discontinuos (por fines de semana o feriados) y analizar el comportamiento de los datos dentro de esos bloques, lo cual es especialmente útil en contextos donde los datos no siguen una frecuencia estrictamente diaria, como suele ocurrir en mercados financieros que no operan durante los fines de semana.

La lógica implementada en esta función resulta particularmente valiosa para bots de trading o análisis cuantitativos que requieran estudiar la evolución de indicadores financieros en intervalos definidos, calcular métricas de rendimiento semanales, o realizar pruebas sobre la persistencia de señales a lo largo de diferentes ventanas temporales.

```
def contar_fechas(data):
    data['prev_date'] = data['date'].shift(1)
    data['diff'] = (data['date'] - data['prev_date']).dt.days
    data['reset'] = data['diff'] > 2
    data['group'] = data['reset'].cumsum()
    data['day_count'] = data.groupby('group').cumcount() + 1
    data.drop(columns=['prev_date', 'diff', 'reset'], axis=1, inplace=True)

    return data
```

La elección de utilizar pandas en lugar de numpy para realizar análisis temporales en este contexto responde principalmente al formato de los datos proporcionados por yfinance. Esta

biblioteca entrega la información financiera en estructuras del tipo DataFrame, lo que hace que el uso de pandas sea más natural y eficiente para la manipulación y el análisis de fechas.

El manejo de fechas, la indexación temporal y las funciones de agrupamiento se integran de forma nativa en pandas, lo cual permite realizar tareas complejas de manera más concisa y legible que con numpy. Por ejemplo, al segmentar los datos en grupos semanales —como se muestra en la función contar_fechas—, resulta trivial iterar sobre cada bloque utilizando groupby() y calcular estadísticas como el promedio de cierre de cada semana.

```
for i, grupo in data.groupby("group"):
    precios = grupo["Close"]
    avg = precios.mean()
    print(f"Semana {i}, Precios: {precios.values}, Promedio: {avg}")
```

Este fragmento de código permite obtener el valor promedio semanal del precio de cierre, una métrica útil para evaluar la evolución del activo en el tiempo. Gracias a la flexibilidad de pandas, es posible no solo acceder fácilmente a los valores de cada grupo, sino también realizar operaciones estadísticas agregadas sin necesidad de convertir los datos a otros formatos ni reestructurarlos.

En síntesis, el uso de pandas en lugar de numpy para este tipo de tareas responde tanto a la estructura de entrada como a la naturaleza de los análisis temporales requeridos, que se ven facilitados por las capacidades de agrupamiento, manipulación de fechas y cálculo estadístico de esta biblioteca.

Bibliografía

Ivanov, I. (2015). *NumPy beginner 's guide* (3rd ed.). Packt Publishing.

Bruce, P., & Bruce, A. (2017). *Practical statistics for data scientists*. O'Reilly Media.

NumPy community. (2024). *NumPy user guide* (Release 2.2.0).