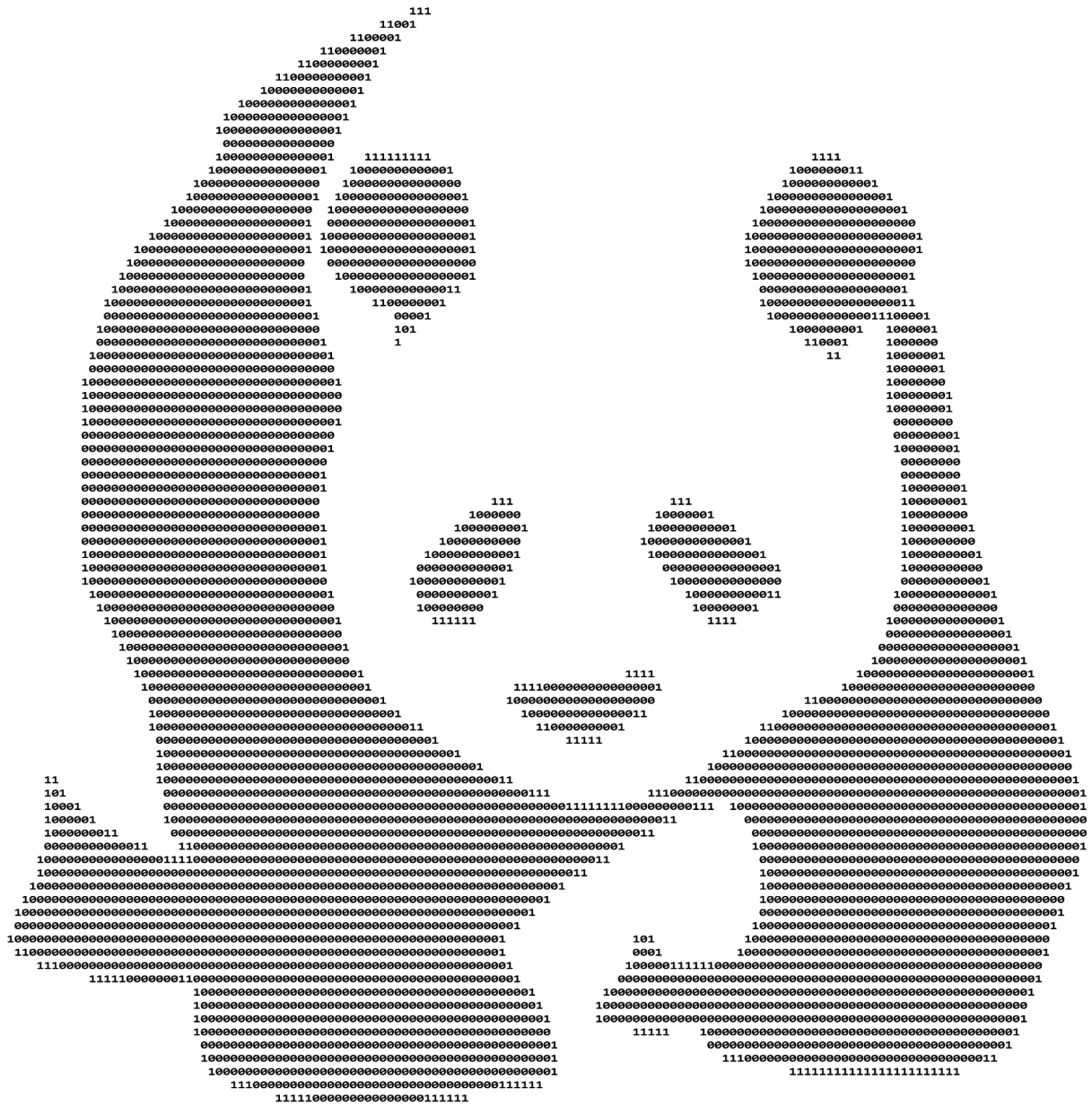
Pandas Projects



Objetivo del Documento

El propósito del presente documento es examinar el uso del módulo *pandas* del lenguaje de programación Python, a través de la implementación de proyectos prácticos. Esta metodología tiene como objetivo facilitar una comprensión profunda tanto de la documentación técnica como del uso adecuado de dicha herramienta, promoviendo simultáneamente el desarrollo de aplicaciones con valor funcional.

Los proyectos propuestos estarán principalmente enfocados en temáticas vinculadas a la economía y al trading algorítmico, con la finalidad de aportar soluciones prácticas y relevantes dentro de estos campos de estudio.

Proyecto I: Análisis de precios históricos de las acciones en el mercado.

Introducción

El presente proyecto se centra en el análisis de precios históricos de acciones cotizadas en el mercado financiero, utilizando para ello herramientas del ecosistema Python. En particular, se hará uso de la biblioteca yfinance, que permite acceder a datos bursátiles históricos, y de pandas, una librería fundamental para el procesamiento y análisis de datos estructurados.

El enfoque del proyecto estará orientado tanto a la aplicación técnica de estas herramientas como a la comprensión de los fundamentos del trading algorítmico, campo en el cual dichas bibliotecas cumplen un rol esencial. Se pondrá especial énfasis en la utilización de pandas para manipular y analizar la información obtenida, con el fin de obtener resultados precisos y útiles que deriven en la creación de un programa capaz de proporcionar información relevante sobre el comportamiento de los activos analizados.

Asimismo, se buscará integrar conocimientos teóricos propios de la ciencia de datos, con el objetivo de desarrollar salidas analíticas que respondan de manera eficaz a las necesidades planteadas por el entorno del trading algorítmico.

Parte I: Obtención de datos

La biblioteca pandas en Python constituye una herramienta fundamental para el manejo eficiente de grandes volúmenes de datos estructurados. No obstante, surge una pregunta esencial: ¿qué tipo de datos analizaremos y de dónde los obtendremos?

Con el fin de mantener una coherencia temática con el enfoque del trading algorítmico, se utilizará la biblioteca yfinance, la cual permite acceder y descargar información histórica de precios de activos financieros cotizados en el mercado de valores. Además de datos de precios (como apertura, cierre, máximos, mínimos y volumen), yfinance también proporciona información financiera relevante sobre las empresas, incluyendo balances generales, estados de resultados y flujos de efectivo (cash flow).

Si bien estos datos tienen un valor intrínseco, su análisis directo puede resultar complejo sin las herramientas adecuadas. En este contexto, pandas facilita la tarea al ofrecer estructuras de datos potentes y funcionalidades específicas para la manipulación, transformación y análisis eficiente de la información, convirtiéndola en un recurso accesible y útil para la toma de decisiones en entornos financieros.

1.1 Módulos a importar

Importamos tres librerías fundamentales para el análisis de datos financieros. pandas (abreviado como pd) permite manipular datos en forma de tablas, lo cual facilita enormemente el análisis y la transformación de la información. yfinance (abreviado como yf) se utiliza para descargar datos bursátiles históricos desde Yahoo Finance de manera sencilla y automatizada. Por último, datetime (abreviado como dt) sirve para gestionar fechas y horas dentro de Python, permitiendo calcular períodos de tiempo de forma precisa.

```
import pandas as pd
import yfinance as yf
import datetime as dt
```

1.2 Descarga de información

Posteriormente, se define una lista llamada tickers que contiene los símbolos de los activos financieros de interés: "YPFD.BA" (acción de YPF en la bolsa argentina), "AAPL" (Apple en la bolsa estadounidense) y "ALUA.BA" (Aluar en la bolsa argentina). A su vez, se establecen dos variables de fecha: start representa la fecha actual menos siete días, es decir, calcula una x cantidad de tiempo atrás desde hoy; mientras que end almacena la fecha y hora actuales. Estas variables serán utilizadas para establecer el rango temporal de descarga de los datos.

```
tickers = ["YPFD.BA","AAPL","ALUA.BA"]

start = dt.datetime.today() - dt.timedelta(7)
end = dt.datetime.today()
```

A continuación, se implementa un bucle for que recorre cada uno de los tickers de la lista. Dentro de este bucle, se utiliza la función **yf.download()** para descargar la información histórica de precios para cada activo. Esta descarga se realiza con un intervalo de 15 minutos **(interval="15m")** y cubriendo el rango de fechas comprendido entre hace siete días y hoy. La opción **auto_adjust=False** indica que los precios no deben ser ajustados automáticamente por eventos como dividendos o splits de acciones, preservando los valores originales.

Una vez descargados los datos, estos se convierten en un **DataFrame** de pandas llamado data. Se renombran las columnas del DataFrame para que tengan nombres específicos: "Adj Close", "Close", "High", "Low", "Open" y "Vol". Luego, se agregan dos nuevas columnas: una llamada Ticker, que guarda el nombre del activo correspondiente para identificar a qué empresa pertenece cada fila de datos, y otra llamada date, que toma el índice del DataFrame (que representa la fecha y hora de cada registro).

```
tickers = ["YPFD.BA","AAPL","ALUA.BA"]

start = dt.datetime.today() - dt.timedelta(7)
end = dt.datetime.today()

for ticker in tickers:
    yfinance_information = yf.download(ticker, start=start, end=end, interval="15m", auto_adjust=False)

  data = pd.DataFrame(yfinance_information)

  data.columns = ["Adj Close",'Close', 'High', 'Low', 'Open', 'Vol']
  data['Ticker'] = ticker
  data["date"] = data.index

  data.info()
```

Finalmente, el código ejecuta data.info(), una función que muestra un resumen general del DataFrame. Esta salida permite visualizar cuántas filas y columnas tiene el DataFrame, qué

tipo de datos contiene cada columna, y cuántos valores nulos existen, facilitando así una rápida verificación de que la información se haya descargado y organizado correctamente.

1.2.1 Datos en pandas

Pandas maneja principalmente tipos de datos estructurados en dos objetos principales: Series y DataFrames. Una Series es una estructura unidimensional que puede almacenar cualquier tipo de dato (números, cadenas de texto, valores booleanos, fechas, etc.), acompañada de un índice que identifica cada elemento. Se puede pensar en una Series como una columna de Excel: un conjunto de datos alineados uno debajo del otro, cada uno con su propio nombre o posición.

Por otro lado, un DataFrame es una estructura bidimensional, parecida a una hoja de cálculo o una tabla de base de datos, donde cada columna es en realidad una Series. Esto significa que cada columna de un DataFrame puede contener un tipo de dato diferente. Por ejemplo, una columna podría contener números enteros, otra cadenas de texto, otra fechas y otra valores booleanos (True/False), todo en el mismo DataFrame. Esto da a pandas una gran flexibilidad para trabajar con diferentes tipos de información al mismo tiempo.

En cuanto a los tipos de datos más comunes que pandas maneja dentro de las Series y DataFrames, podemos mencionar varios:

- Enteros (int): Son números enteros como 1, 5, 100 o -23. Dentro de pandas, suelen aparecer como int64, indicando que cada número ocupa 64 bits de memoria.
- Números decimales o flotantes (float): Representan números con parte decimal, como 3.14 o -0.01. Normalmente, pandas los almacena como float64.
- Cadenas de texto (object): Aunque parece extraño, pandas guarda las columnas de texto bajo el tipo object, porque puede contener cualquier tipo de objeto de Python, no solo texto. Desde 2020 en adelante, pandas también tiene un tipo más especializado llamado string, pero object sigue siendo muy usado para datos de texto.
- Valores booleanos (bool): Son valores True o False, y pandas los maneja como tipo bool, ocupando muy poca memoria.
- Fechas y tiempos (datetime64): Cuando trabajamos con fechas o tiempos, pandas utiliza el tipo datetime64[ns], donde ns significa nanosegundos, indicando la precisión de las marcas de tiempo.
- Datos categóricos (category): Este tipo especial se usa para columnas que tienen un número limitado de valores posibles (por ejemplo, colores, países, estados civiles). Guardar datos como category es más eficiente en memoria y puede acelerar algunas operaciones.

Finalmente, hay que mencionar que pandas también es capaz de trabajar con valores nulos, que representan datos faltantes. Estos pueden aparecer como NaN (Not a Number) en datos numéricos o como None en datos de texto o mezclados. Manejar correctamente los nulos es esencial para hacer análisis de datos confiables.