Reporte de Servicio Social.

Ángel Edmundo Hernández Martínez

1. Ecuación logistica y la complejidad

Para familiarizarnos con el estudio de la complejidad y el caos tratamos el caso de la Ecuación Logística. La ecuación logística es un modelo de crecimiento poblacional publicado por Pierre Verhulst. Que se expresa con la siguiente ecuación iterativa;

$$x_{t+1} = r \cdot x_t(1 - x_t), \text{ donde } 0 \leq x_0 \leq 1$$

```
import math
import statistics
import numpy as np
import matplotlib.pyplot as plt
from random import randint
import pandas as pd

import seaborn as sns
from functools import cache
import plotly.figure_factory as ff
import plotly.express as px
from collections import Counter


@cache
def logistic (R, x0, N):
    x = x0
    x_list = [x0]
    for i in range(N-1):
        x = R * x * (1.0 - x)
        x_list.append(x)

    return x_list, R, x0

def graficar(x_list, R, x0):
    plt.style.use('seaborn-whitegrid')
    plt.figure(figsize=(16, 6), facecolor='lightgray')
    plt.xlabel('N mero de iteraciones')
    plt.ylabel('Valor de x')
    plt.title(f'\nEcuaci n log stica\n\nR={R}  |  x0={x0}\n')
    plt.plot(x_list, 'o:r')
```

```
plt.show()
```

Donde

2. Creación de una base de datos

2.1. Extracción de datos.

Para extraer los datos de los distintos índices económicos se empleó la API de YahooFinance, conocida como Yfinance. Se tomó esta decisión porque la mayoría de las API's con las mismas funciones eran de paga, y, por otro lado, era más fácil obtener los datos de esta manera que extrayendo los datos de forma directa aplicando técnicas de Web Scrapping.

El uso de Yfinance es bastante simple, aún así creo esclarecedor una breve explicación de su funcionamiento. Yfinance se basa en un sistema de Tickets que son la forma de comunicarnos con el sitio Web de Yahoo Finance. Para emplear los tickets necesitamos de la clave con la que están designados los índices económicos, la cual puede ser consultada desde la página de YahooFinance. Por ejemplo, la clave del IPC MEXICO es "ˆMXX".

Con el ticket podemos consultar información del índice económico como sus acciones, sus dividendos, sus ganancias, etcétera. A continuación presentamos un ejemplo de código para consultar la información relacionada a un índice económico.

```
import yfinance as yf
IPC = yf.Ticker("^MXX")
# Muestra acciones (dividendos, splits)
IPC.actions
# Muestra dividendos
IPC.dividens
# Muestra splits
IPC.splits
# Existen mas metodos, que pueden ser consultados en la documentacion
```

Ahora con la posibilidad de obtener la información, necesitamos guardarla de algún modo. Y queríamos que fuera lo antes posible porque los datos con granularidad menor a un día en las series de tiempo se eliminaban después de 7 días.

La solución fue que mientras se diseñaba la base de datos, los datos se almacenaran en formato ".csv". Para ello hice un script que almacenaba los datos en la computadora y que se activaba de forma automática todos los días. Ahora comentaré las partes de dicho código, y explicaré la forma en que lo automaticé.

La primera parte del código es la importación de las funciones y librerias que ocuparemos, a saber;

```
import yfinance as yf
from pathlib import Path
from datetime import date, timedelta, datetime
import time
```

El módulo `pathlib` nos permite manipular las rutas de los archivos del sistema, de forma muy similar a `os.path`, pero con la diferencia de `pathlib` lo puede lograr a alto nivel y que su interface es mucho más conveniente. Por otro lado, los módulos `datetime` y `time` nos permite trabajar con fechas y tiempos, para ello emplea una clase conocida como datetime.

La siguiente parte del código consiste en la enumeración de los tickets de los distintos índices económicos de los que queremos extraer la información. Como explicamos antes, para poder usar los tickets necesitamos el TAG de los ínidces económicos que aparecen en la página de YahooFinance. Así, por ejemplo, tenemos que el TAG del NASDAQ es "ˆ NDX "ó del DOWJONES es "ˆDJI".

```
SP500 = yf.Ticker("^GSPC") #S&P 500
Nasdaq =yf.Ticker("^NDX") #Nasdaq 100
Nasdaqcomp = yf.Ticker("^IXIC") #Nasdaq Composite
DWJ = yf.Ticker("^DJI") #Dow Jones Industrial Average
IPC = yf.Ticker('^MXX') #IPC (BMV)
STOXX50E = yf.Ticker('^STOXX50E') #STOXX50
Nikkei = yf.Ticker('^N225') #Nikkei
CSI300 = yf.Ticker('000300.SS') #CS1300
```

La siguiente parte del código comienza guardando en listas los tickets y el nombre de los mismos, esto lo hacemos así para usarlos en los ciclos tipo **for**.

En el primer **for** establecemos las fechas usando el módulo `datetime`. El método `.today()` sirve, como se podría adivinar, para devolver la fecha del día de hoy en formato datetime. Por otro lado, el método `.replace()` sirve para modificar los valores de los datos tipo datetime. Y, la función `timedelta()` se usa para restar la cantidad de tiempo que se especifique.

Así lo que estamos haciendo es obtener la fecha de hoy hasta siete días antes, en un intervalo de 23 horas. Estos serán los días que obtendremos la información de los tickets.

En el siguiente **for**, que recorre la longitud de la lista de tickets, con el método `.history` obtenemos el historial de los tickets con intervalos de un minuto. Luego, con la función `Path` del módulo `pathlib` creamos con una ruta de acceso para los archivos .csv de acuerdo con el nombre del ticket y la fecha a la que pertenece, para ello usamos el método **.format**.

Por último, con el método `.to_csv` convertimos el historiald de los tickets en .csv, y especificamos que deseamos los encabezados y los índices, de igual modo, especificamos la ruta en la que se guardará con el "filepath"que habíamos crado anteriormente.

```
tickets = [SP500, Nasdaq, Nasdaqcomp, DWJ, IPC, STOXX50E, Nikkei, CSI300]
ticketsname = ["SP500", "Nasdaq", "Nasdaqcomp", "DWJ", "IPC", "STOXX50", "Nikkei", "CSI300"]

for i in range(7):
    start = datetime.today().replace(hour = 0, minute = 0, second = 0 ) — timedelta(days= i +1)
    end = datetime.today().replace(hour = 23) — timedelta(days = i +1)

    for i in range(len(tickets)):
        historial = tickets[i].history(start=start, end=end , interval="1m")
        filepath = Path('Info/{almacenamiento}/{name}_{date}.csv'
                .format(almacenamiento = ticketsname[i] ,name = ticketsname[i], date = start.strftime('%Y—%m—%d')))
        filepath.parent.mkdir(parents=True, exist_ok=True)
        historial.to_csv(filepath, header=True, index=True)
```

Por otro lado para la automatización del script ocupé el administrador de tareas de windows.

2.2. Diseño de base de datos

El diseño de una base de datos

2.3. Implementación y automatización de la base de datos

El plan original era usar PostgresSQL como el sistema de gestión de base de datos, pero por cuestiones técnicas terminé usando MySQL.

3. Análisis de distintos índices

3.1. Análisis general de los índices económicos

3.2. Análisis sobre complejidad de los índices económicos

3.2.1. Approximate entropy

El objetivo de la *approximate entropy* (ApEn) es estimar la

3.2.2. Sample entropy

4. Bibliografía

https://algotrading101.com/learn/yfinance-guide/
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7515030/