Consultas Machine Learning

Python

Análisis financiero para predicciones futuras en bolsa

```python
# 1. Importación de librerías
# ====================================
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
from statsmodels.tsa.seasonal import seasonal_decompose

# Configuración de gráficos
plt.style.use("seaborn-v0_8-darkgrid")  # Compatible con versiones nuevas
sns.set_theme(style="darkgrid")         # Activar estilo seaborn
plt.rcParams['figure.figsize'] = (12,6)
```

```python
# 2. Cargar CSV y vista inicial
# ====================================
df = pd.read_csv("Analista de datos AAPL_2006-01-01_to_2018-01-01_M30.csv")

print("Primeras filas:")
display(df.head())
print("\nInfo del DataFrame:")
display(df.info())
```

Primeras filas:

	Date	Open	High	Low	Close	Volume	Name
0	2006-01-03	10.34	10.68	10.32	10.68	201853036	AAPL
1	2006-01-04	10.73	10.85	10.64	10.71	155225609	AAPL
2	2006-01-05	10.69	10.70	10.54	10.63	112396081	AAPL
3	2006-01-06	10.75	10.96	10.65	10.90	176139334	AAPL
4	2006-01-09	10.96	11.03	10.82	10.86	168861224	AAPL

```
Info del DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3019 entries, 0 to 3018
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    3019 non-null   object
 1   Open    3019 non-null   float64
 2   High    3019 non-null   float64
 3   Low     3019 non-null   float64
 4   Close   3019 non-null   float64
 5   Volume  3019 non-null   int64
 6   Name    3019 non-null   object
dtypes: float64(4), int64(1), object(2)
memory usage: 165.2+ KB
None
```

```python
# 3. Limpieza de datos y conversión de fecha
# ==========================================
# Convertir columna de fecha a datetime
df['Date'] = pd.to_datetime(df['Date'])

# Ordenar por fecha
df = df.sort_values('Date')

# Revisar si hay valores nulos
print("\nValores nulos por columna:")
print(df.isnull().sum())

# Eliminar duplicados
df = df.drop_duplicates()
```

```
Valores nulos por columna:
Date      0
Open      0
High      0
Low       0
Close     0
Volume    0
Name      0
dtype: int64
```

```python
# 4. Establecer índice temporal
# ==========================================
df.set_index('Date', inplace=True)

# Vista rápida
display(df.head())
```

Date	Open	High	Low	Close	Volume	Name
2006-01-03	10.34	10.68	10.32	10.68	201853036	AAPL
2006-01-04	10.73	10.85	10.64	10.71	155225609	AAPL
2006-01-05	10.69	10.70	10.54	10.63	112396081	AAPL
2006-01-06	10.75	10.96	10.65	10.90	176139334	AAPL
2006-01-09	10.96	11.03	10.82	10.86	168861224	AAPL

```python
# 5. Rangos de datos
# ====================================
print("\nFecha mínima:", df.index.min())
print("Fecha máxima:", df.index.max())

# Filtrar todo el año 2010
df_2010 = df.loc['2010']
display(df_2010.head())
```

```
Fecha mínima: 2006-01-03 00:00:00
Fecha máxima: 2017-12-29 00:00:00
```

Date	Open	High	Low	Close	Volume	Name
2010-01-04	30.49	30.64	30.34	30.57	123432050	AAPL
2010-01-05	30.66	30.80	30.46	30.63	150476004	AAPL
2010-01-06	30.63	30.75	30.11	30.14	138039594	AAPL
2010-01-07	30.25	30.29	29.86	30.08	119282324	AAPL
2010-01-08	30.04	30.29	29.87	30.28	111969081	AAPL

```python
# 6. Shifting y Lags
# ====================================
df['Close_shift_1'] = df['Close'].shift(1)  # Desplazamiento 1 periodo
df['Close_lag_5'] = df['Close'].shift(5)    # Desplazamiento 5 periodos

display(df[['Close','Close_shift_1','Close_lag_5']].head(10))
```

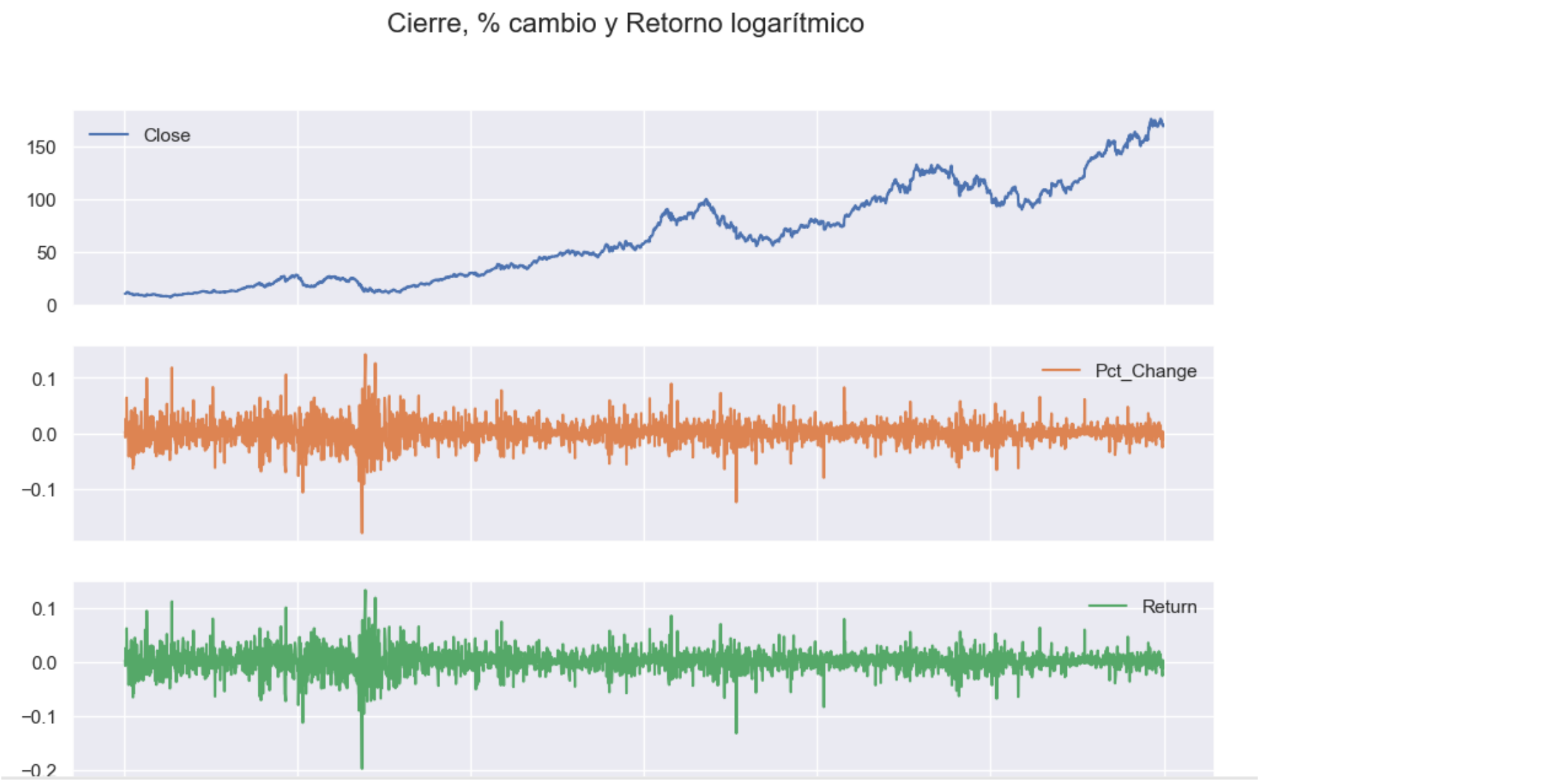
Date	Close	Close_shift_1	Close_lag_5
2006-01-03	10.68	NaN	NaN
2006-01-04	10.71	10.68	NaN
2006-01-05	10.63	10.71	NaN
2006-01-06	10.90	10.63	NaN
2006-01-09	10.86	10.90	NaN
2006-01-10	11.55	10.86	10.68
2006-01-11	11.99	11.55	10.71
2006-01-12	12.04	11.99	10.63
2006-01-13	12.23	12.04	10.90
2006-01-17	12.10	12.23	10.86

```python
# 7. Resampling (cambio de frecuencia) - corregido
df_resample = df.select_dtypes(include=[np.number]).resample('M').mean()
display(df_resample.head())
```

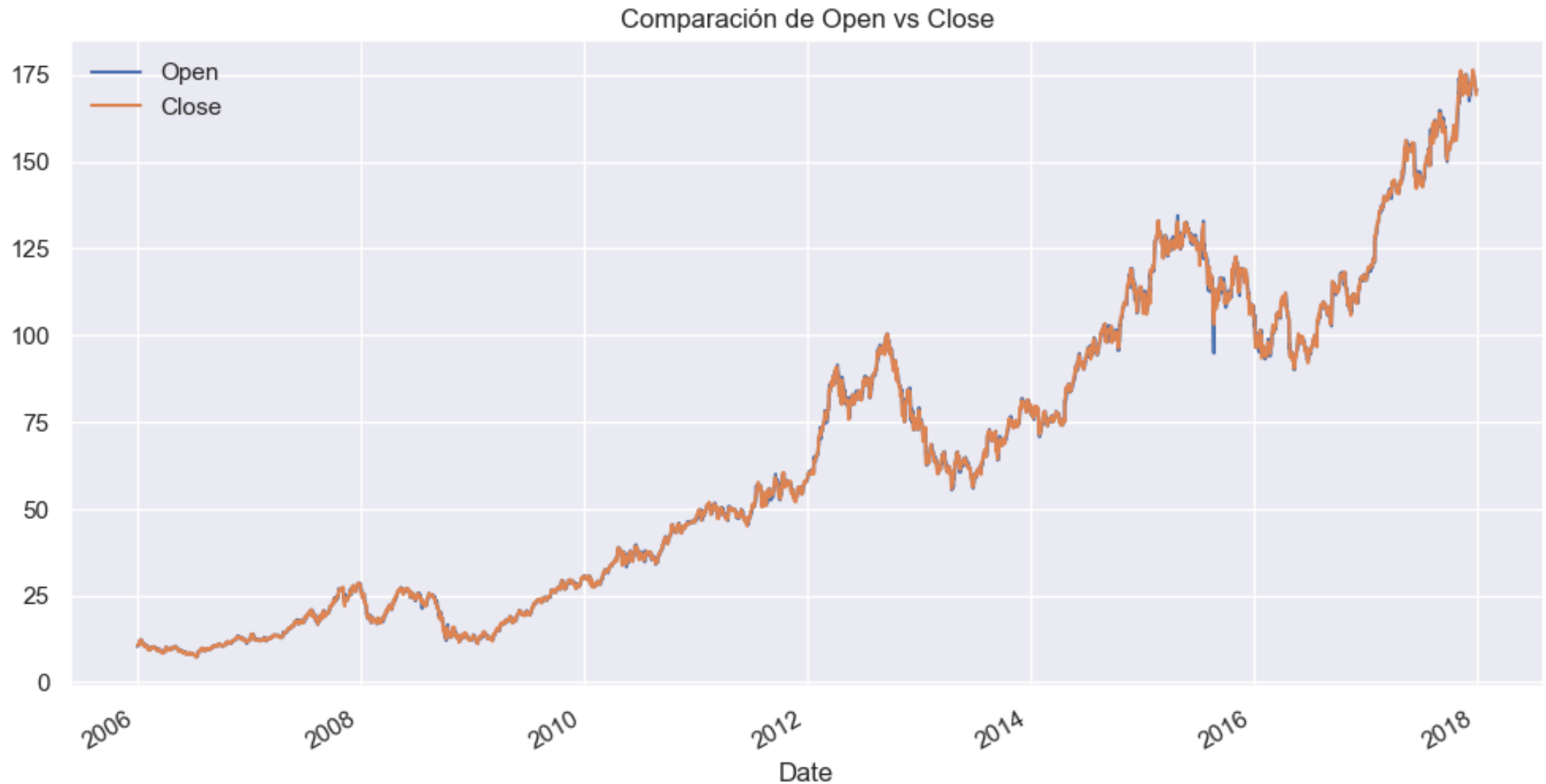
Date	Open	High	Low	Close	Volume	Close_shift_1	Close_lag_5
2006-01-31	11.144000	11.352000	10.931500	11.115500	2.737752e+08	11.132632	11.306000
2006-02-28	10.011579	10.164737	9.760000	9.930526	2.469824e+08	9.983684	10.043684
2006-03-31	9.229565	9.336522	9.034348	9.146957	2.521987e+08	9.182609	9.445217
2006-04-30	9.594211	9.733158	9.424211	9.572632	2.648837e+08	9.514737	9.317895
2006-05-31	9.642273	9.728636	9.435000	9.540455	1.775120e+08	9.609545	9.719545

```python
# 8. % Cambios y retornos
# ====================================
df['Pct_Change'] = df['Close'].pct_change()
df['Return'] = np.log(df['Close'] / df['Close'].shift(1))  # Retorno logarítmico

df[['Close','Pct_Change','Return']].plot(subplots=True, layout=(3,1), figsize=(12,8))
plt.suptitle("Cierre, % cambio y Retorno logarítmico", fontsize=16)
plt.show()
```



```
# 9. Comparación de series
# Ejemplo: Close vs Open
# ======================================
df[['Open','Close']].plot(figsize=(12,6), title="Comparación de Open vs Close")
plt.show()
```



```python
# 10. Gráficos OHLC y Candlestick con Plotly
# ========================================
fig_ohlc = go.Figure(data=go.Ohlc(
    x=df.index,
    open=df['Open'],
    high=df['High'],
    low=df['Low'],
    close=df['Close']
))
fig_ohlc.update_layout(title="Gráfico OHLC", xaxis_rangeslider_visible=False)
fig_ohlc.show()

fig_candle = go.Figure(data=[go.Candlestick(
    x=df.index,
    open=df['Open'],
    high=df['High'],
    low=df['Low'],
    close=df['Close']
)])
fig_candle.update_layout(title="Gráfico Candlestick", xaxis_rangeslider_visible=False)
fig_candle.show()
```

Gráfico OHLC

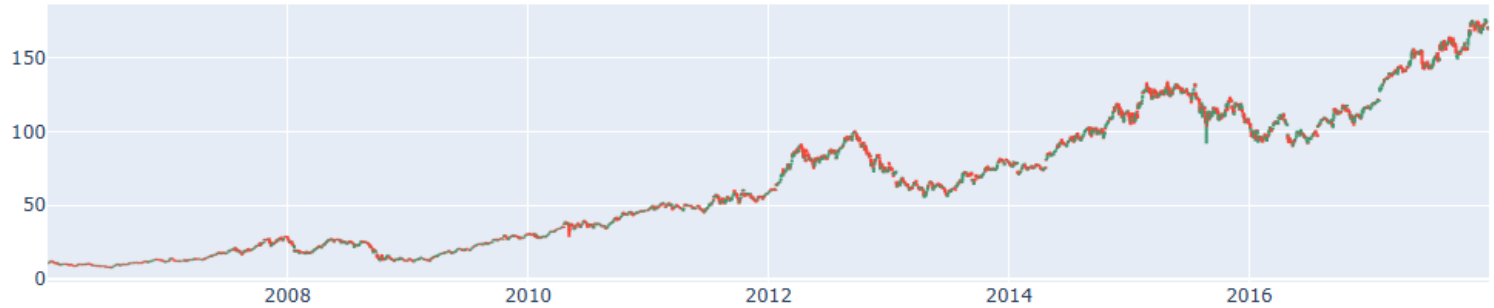
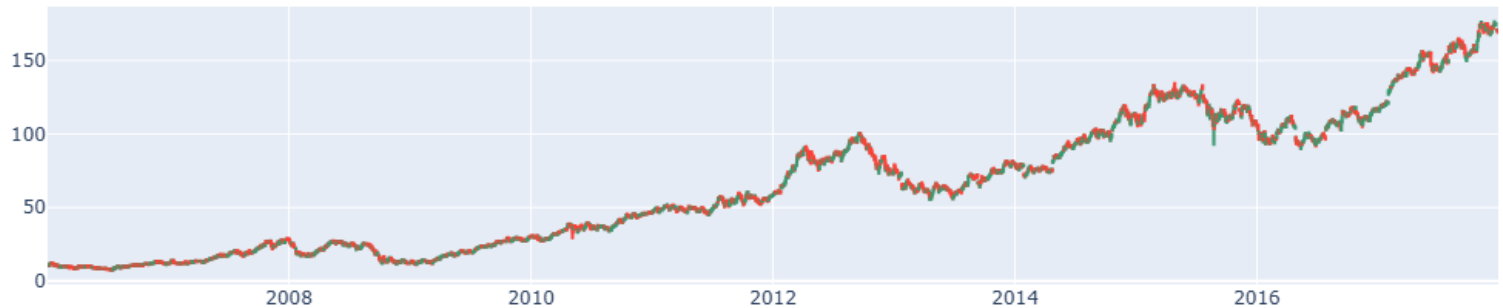


Gráfico Candlestick



```python
# 11. Descomposición de series de tiempo
# ======================================
df_monthly = df['Close'].resample('M').mean()

decomposition = seasonal_decompose(df_monthly, model='multiplicative')

fig = decomposition.plot()
fig.set_size_inches(12, 9)
plt.show()
```