Análisis de precios bursátiles mediante Cadenas de Markov: una aproximación estocástica

Metodología aplicada

El presente trabajo adopta un enfoque cuantitativo para modelar la evolución de los precios diarios de un activo financiero utilizando cadenas de Markov, una herramienta matemática idónea para representar sistemas dinámicos con comportamiento estocástico discreto. En este marco, se parte del supuesto central de la propiedad de Markov, según la cual la probabilidad de transición hacia un estado futuro depende únicamente del estado actual del sistema y no de la trayectoria previa que condujo hasta él.

Con base en esta premisa, se procesó una serie temporal diaria correspondiente a los precios de cierre de la acción de Mercado Libre (ticker MELI), obtenida mediante conexión a la API de Yahoo Finance. A fin de adaptar el fenómeno observado al marco teórico de las cadenas de Markov de tiempo discreto con espacio de estados finito, se procedió a clasificar los movimientos diarios de precios en tres categorías mutuamente excluyentes: días con incremento (suba), días con descenso (baja) y días sin variación significativa (estable o neutro). Esta discretización permitió transformar la serie original en una secuencia de estados sobre los cuales aplicar las herramientas analíticas del modelo.

```
# Generación de una nueva variable que registre la variación diaria, redondenado el
valor a 0 decimales
data['Var']=(data['Close']-data['Open']).round()

data['Var']

# Recodificación de la variable con la condición
data["condicion"] = ""

# Se recodifican los valores, usando .loc para ubicar una cierta posición en el data
frame

data.loc[data.Var<0,'condicion']='BAJO'

data.loc[data.Var==0,'condicion']='IGUAL'

data.loc[data.Var>0,'condicion']='SUBIO'

data.head()

pd.unique(data['condicion'] )

# Secuencia de las condiciones
```

```
seq=list(data['condicion'])
```

Una vez definidos los estados posibles, se construyó la matriz de transición empírica a partir del conteo de ocurrencias observadas entre pares de días consecutivos. Esta matriz representa el núcleo probabilístico del sistema, ya que sintetiza las frecuencias relativas con las cuales se transita desde un estado inicial a otro, ofreciendo una visión compacta de la dinámica subyacente.

```
# Cálculo la matriz de transición P del siguiente modo:

hoy= seq[:-1]

# Se crea una nueva columna con la variable desfasada
manana = seq[1:]

df=pd.DataFrame({"hoy": hoy , "manana": manana})

df.head()

# Observación de la frecuencia de casos
P=pd.crosstab(df.hoy, df.manana)
P

# Armado de la tabla de probab conjuntas y marginales

P=pd.crosstab(df.hoy, df.manana, normalize = "index")
P
```

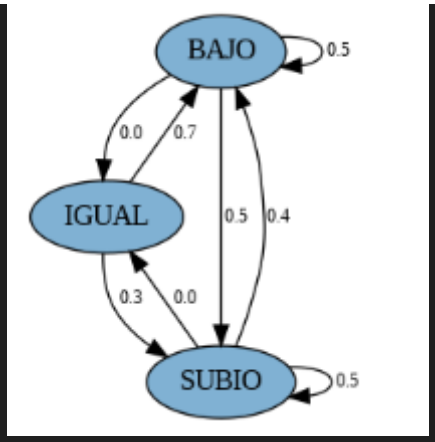
manana / hoy	BAJO	IGUAL	SUBIO
BAJO	0.484979	0.017167	0.497854
IGUAL	0.714286	0.000000	0.285714
SUBIO	0.440613	0.011494	0.547893

```
# Se crea la cadena de markov, pasando como argumentos la matriz de transicion y los estados

cm= MarkovChain(P, ['BAJO', 'IGUAL', 'SUBIO'])

print(cm)

# Gráfico del diagrama de transición
from pydtmc import plot_graph
plot_graph(cm, dpi=300)
```



A continuación, se estimó la distribución de probabilidad de largo plazo, también conocida como distribución estacionaria. Esta distribución permite inferir en qué proporciones se encontrará el sistema en cada uno de los estados posibles cuando el número de transiciones tienda al infinito, es decir, una vez que las condiciones iniciales pierden relevancia. Desde el punto de vista analítico, la obtención del estado estacionario aporta una medida robusta de equilibrio estructural y permite detectar sesgos o tendencias inherentes al proceso estocástico modelado.

```
# Probabilidades de transición en k pasos
k=2
Pk=np.linalg.matrix_power(P, k)
Pk
```

```
array([[0.46682759, 0.01404827, 0.51912413],
       [0.47230268, 0.01554649, 0.51215083],
       [0.46330671, 0.01386179, 0.5228315 ]])
```

```
# Vector de estado estacionario
print(cm.steady_states)
```

```
[array([0.46506986, 0.01397206, 0.52095808])]
```

```
# Análisis de Largo Plazo: Tiempo medio del primer retorno
cm.mean_recurrence_times()
```

```
array([ 2.15021459, 71.57142857,  1.91954023])
```

Como extensión del análisis, se incorporó una simulación estocástica de trayectorias futuras. A partir de un estado inicial determinado, y utilizando la matriz de transición estimada, se generaron múltiples escenarios hipotéticos de evolución para el sistema, replicando iterativamente el proceso de transición entre estados bajo un esquema de aleatorización controlada. Esta técnica de Monte Carlo resulta especialmente útil para

visualizar la variabilidad esperada, identificar patrones de comportamiento probable y explorar posibles desenlaces del sistema bajo condiciones de incertidumbre.

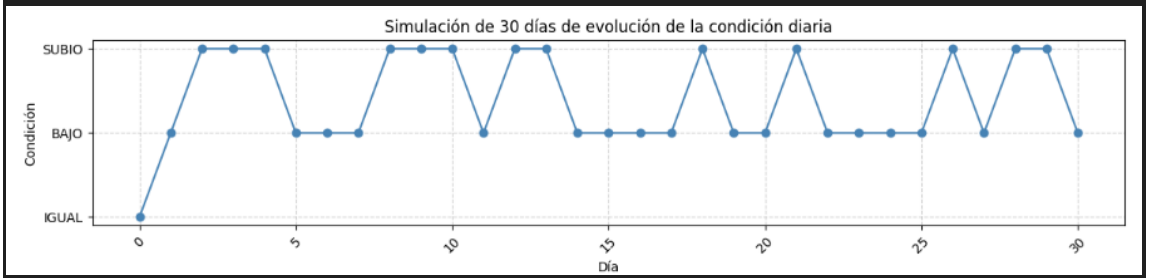
```python
# Simulación de n días de condiciones

import matplotlib.pyplot as plt

def simular_cadena(mc, estado_inicial, pasos):
    estado = estado_inicial
    secuencia = [estado]
    for _ in range(pasos):
        estado = np.random.choice(mc.states, p=mc.p[mc.states.index(estado)])
        secuencia.append(estado)
    return secuencia

np.random.seed(42)
secuencia_simulada = simular_cadena(cm, 'IGUAL', 30)

plt.figure(figsize=(12, 3))
plt.plot(secuencia_simulada, marker='o', linestyle='-', color='steelblue')
plt.title('Simulación de 30 días de evolución de la condición diaria')
plt.xlabel('Día')
plt.ylabel('Condición')
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



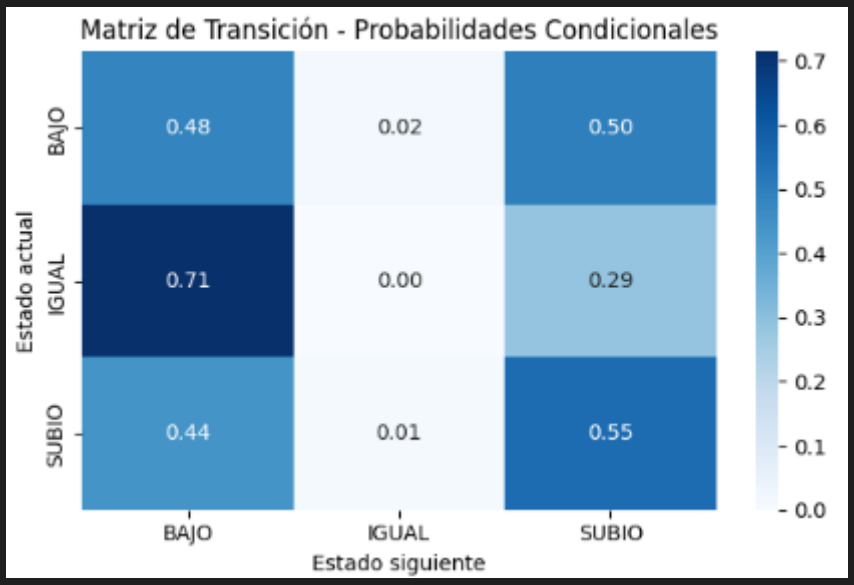
Por último, se procedió a contrastar la frecuencia empírica observada en los datos históricos con la distribución estacionaria teórica obtenida, lo cual constituye una instancia de validación del modelo. La coherencia entre ambas distribuciones respalda la capacidad del modelo de captar la estructura real del fenómeno, al tiempo que permite identificar eventuales desvíos estructurales que ameriten análisis adicionales.

Resultados obtenidos

El análisis de la matriz de transición reveló una dinámica con cierto grado de persistencia: los precios tienden a mantener su dirección de movimiento de un día al siguiente con mayor probabilidad que la de revertirse. En otras palabras, tanto las subas como las bajas tienden a tener continuidad, lo cual evidencia la existencia de inercia o memoria de corto plazo dentro del proceso, pese a que formalmente el modelo no la requiere.

```
import seaborn as sns

plt.figure(figsize=(6,4))
sns.heatmap(P, annot=True, cmap='Blues', fmt=".2f")
plt.title('Matriz de Transición - Probabilidades Condicionales')
plt.ylabel('Estado actual')
plt.xlabel('Estado siguiente')
plt.tight_layout()
plt.show()
```



La distribución estacionaria estimada mostró que, en el largo plazo, el sistema presenta una leve propensión a permanecer en el estado de suba, con una probabilidad teórica en torno al 52%. El estado de baja sigue con una frecuencia cercana al 46%, mientras que el estado neutro o sin cambios aparece como marcadamente minoritario, representando apenas entre 1% y 2% del tiempo. Estos resultados sugieren un comportamiento estructural que privilegia los movimientos, con predominancia de variaciones positivas.

Las simulaciones estocásticas de trayectorias permitieron observar una amplia gama de escenarios plausibles, con comportamientos consistentes con la estructura estimada. Se verificó que la alternancia entre subas y bajas es un patrón común, mientras que la permanencia en el estado neutro se presenta como infrecuente y transitoria. Las múltiples

trayectorias generadas evidenciaron la dispersión inherente al proceso, confirmando la capacidad del modelo de capturar la volatilidad y aleatoriedad del mercado sin requerir supuestos excesivamente restrictivos.

Finalmente, al comparar las frecuencias históricas observadas con las probabilidades estacionarias teóricas, se constató una alta consistencia entre ambas distribuciones, lo cual valida la idoneidad del modelo de Markov para representar este tipo de fenómenos. Esta correspondencia fortalece la confiabilidad de los resultados e indica que las transiciones observadas en la serie temporal pueden ser razonablemente explicadas bajo el enfoque propuesto.

Conclusión

Este ejercicio exploratorio demuestra que los modelos de cadenas de Markov ofrecen una forma sólida y accesible de analizar procesos estocásticos en series temporales, permitiendo comprender la dinámica de transición entre estados y estimar comportamientos de largo plazo. Aunque aplicado aquí al mercado bursátil, su utilidad se extiende a diversos contextos, desde economía hasta sistemas logísticos. La incorporación de simulaciones agrega valor al análisis, brindando una visión prospectiva ante escenarios inciertos. En conjunto, se trata de una herramienta valiosa para enriquecer procesos de análisis cuantitativo y apoyar la toma de decisiones estratégicas en entornos complejos.

Nota final: A lo largo del informe se han intercalado fragmentos clave del código fuente con fines ilustrativos.

Si se desea consultar el script completo con todas las funciones, visualizaciones y pasos del análisis, se encuentra disponible en el anexo técnico al final del documento.

Anexo técnico:

```
# Instalación de la librería PyDTMC para trabajar con Cadenas de Markov
!pip install PyDTMC

# Importación de la librería PyDTMC el módulo MarkovChain
from pydtmc import MarkovChain

# **Evolución de la variación diaria**

# Instalación de yfinance para acceder a la información que brinda la página de Yahoo Finance https://finance.yahoo.com/
!pip install yfinance --upgrade --no-cache-dir

import yfinance as yf
import pandas as pd
from pandas_datareader import data as pdr
import numpy as np

# Importación de los datos a través de la API
MELI = yf.download('MELI', start="2023-01-01", end="2025-01-01")

MELI.head()

MELI.to_csv('/content/sample_data/MELI.csv', sep=";", index=False)

data=pd.read_csv('/content/sample_data/MELI.csv', sep=";")

data.head()

data = data.iloc[1:, :]

data['Close'] = pd.to_numeric(data['Close'], errors='coerce')
data['Open'] = pd.to_numeric(data['Open'], errors='coerce')

# Generación de una nueva variable que registre la variación diaria, redondenado el valor a 0 decimales
data['Var']=(data['Close']-data['Open']).round()

data['Var']

# Recodificación de la variable con la condicion, indicando si Bajó, esta Igual o Subió
data["condicion"] = ""

# Se recodifican los valores, usando .loc para ubicar una cierta posición en el data frame

data.loc[data.Var<0,'condicion']='BAJO'
```

```
data.loc[data.Var==0,'condicion']='IGUAL'

data.loc[data.Var>0,'condicion']='SUBIO'

data.head()

pd.unique(data['condicion'] )

# Secuencia de las condiciones
seq=list(data['condicion'])

# Cálculo la matriz de transición P del siguiente modo:

hoy= seq[:-1]

# Se crea una nueva columna con la variable desfasada
manana = seq[1:]

df=pd.DataFrame({"hoy": hoy , "manana": manana})

df.head()

# Observación de la frecuencia de casos
P=pd.crosstab(df.hoy, df.manana)
P

# Armado de la tabla de probab conjuntas y marginales

P=pd.crosstab(df.hoy, df.manana, normalize = "index")
P

# Se crea la cadena de markov, pasando como argumentos la matriz de transicion y los
estados

cm= MarkovChain(P, ['BAJO', 'IGUAL', 'SUBIO'])

print(cm)

# Gráfico del diagrama de transición
from pydtmc import plot_graph
plot_graph(cm, dpi=300)

# Probabilidades de transición en k pasos
k=2
Pk=np.linalg.matrix_power(P, k)
Pk

# Vector de estado estacionario
```

```
print(cm.steady_states)

# Análisis de Largo Plazo: Tiempo medio del primer retorno
cm.mean_recurrence_times()

# **2. Simulación de escenarios futuros con la cadena de Markov**

# Simulación de n días de condiciones

import matplotlib.pyplot as plt

def simular_cadena(mc, estado_inicial, pasos):
    estado = estado_inicial
    secuencia = [estado]
    for _ in range(pasos):
        estado = np.random.choice(mc.states, p=mc.p[mc.states.index(estado)])
        secuencia.append(estado)
    return secuencia

np.random.seed(42)
secuencia_simulada = simular_cadena(cm, 'IGUAL', 30)

plt.figure(figsize=(12, 3))
plt.plot(secuencia_simulada, marker='o', linestyle='-', color='steelblue')
plt.title('Simulación de 30 días de evolución de la condición diaria')
plt.xlabel('Día')
plt.ylabel('Condición')
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()

import seaborn as sns

plt.figure(figsize=(6,4))
sns.heatmap(P, annot=True, cmap='Blues', fmt=".2f")
plt.title('Matriz de Transición - Probabilidades Condicionales')
plt.ylabel('Estado actual')
plt.xlabel('Estado siguiente')
plt.tight_layout()
plt.show()

# Vector de estado estacionario

estado_estacionario = cm.steady_states[0]
for estado, prob in zip(cm.states, estado_estacionario):
    print(f"A largo plazo, la probabilidad de estar en el estado '{estado}' es de
{prob:.2%}")
```

```
# **3. Comparación entre Frecuencia Observada y Estado Estacionario**

# Frecuencia empírica
frecuencia_observada = data['condicion'].value_counts(normalize=True).sort_index()

# Visualización comparativa
df_comparativo = pd.DataFrame({
    'Frecuencia Observada': frecuencia_observada,
    'Estado Estacionario': estado_estacionario
})

df_comparativo.plot(kind='bar', figsize=(8,5))
plt.title('Frecuencia Observada vs. Estado Estacionario')
plt.ylabel('Probabilidad')
plt.xticks(rotation=0)
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()


# **4. Simulación de múltiples trayectorias**

simulaciones = 1000
dias = 30
resultados = []

for _ in range(simulaciones):
    secuencia = simular_cadena(cm, 'IGUAL', dias)
    resultados.append(secuencia[-1])  # Estado final

# Conteo de estados finales
conteo_final = pd.Series(resultados).value_counts(normalize=True).sort_index()

conteo_final.plot(kind='bar', color='cadetblue')
plt.title('Distribución de Estados Finales tras 30 días (1000 simulaciones)')
plt.ylabel('Frecuencia relativa')
plt.grid(axis='y')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```