Universidad Nacional Autónoma de México

Facultad de Ciencias

Métodos Cuantitativos en Finanzas

Tarea-Examen CAPM vs APT

Integrantes
Rubén Gáscon Calixco
Jonathan Ivan Salmoran Acuña

Profesor
Alma Rosa Bustamante García
Ayudantes
Sergio Martín Martínez Pérez
Guadalupe Abigail Chirino Hérnandez
Luis Mauricio Aguilar Munguía

June 3, 2025

1. Deduzca la fórmula de CAPM explicando todos los pasos y los supuestos nos llevan a la elegibilidad del Portafolio de Mercado

Supuestos:

- **Inversionistas racionales y aversos al riesgo**: Prefieren mayor retorno y menor riesgo.
- **Mercados eficientes**: Toda la información está disponible y se refleja instantáneamente en los precios.
- **Homogeneidad de expectativas**: Todos los inversionistas tienen la misma información y expectativas.
- **Existencia de un activo libre de riesgo (R_f)**: Se puede prestar o invertir a una tasa constante sin riesgo.
- **Horizonte temporal único**: Todos los inversionistas planean en el mismo período.
- **Sin costos de transacción, impuestos ni restricciones**: Permite la libre asignación de capital.
- **Activos perfectamente divisibles y líquidos**.
- **Oferta y demanda están completamente satisfechas**.
- **Distribución normal de los rendimientos**.

Estos supuestos permiten la existencia de un único portafolio eficiente común a todos los inversionistas: el *portafolio de mercado*.

Portafolio de Mercado

Es un portafolio teórico que contiene **todos los activos riesgosos** del mercado ponderados por su valor de mercado, de lo anterior inferimos que el Portafolio Óptimo es el Portafolio de Mercado:

Bajo los supuestos del CAPM, el portafolio: - Está sobre la **frontera eficiente de Markowitz**. - **Maximiza el ratio de Sharpe**:

$$\text{Sharpe Ratio} = \frac{\mathbb{E}(R_{PM}) - R_f}{\sigma_{PM}}$$

Con R_f el rendimiento del activo libre de riesgo.

Línea de Colocación de Capital

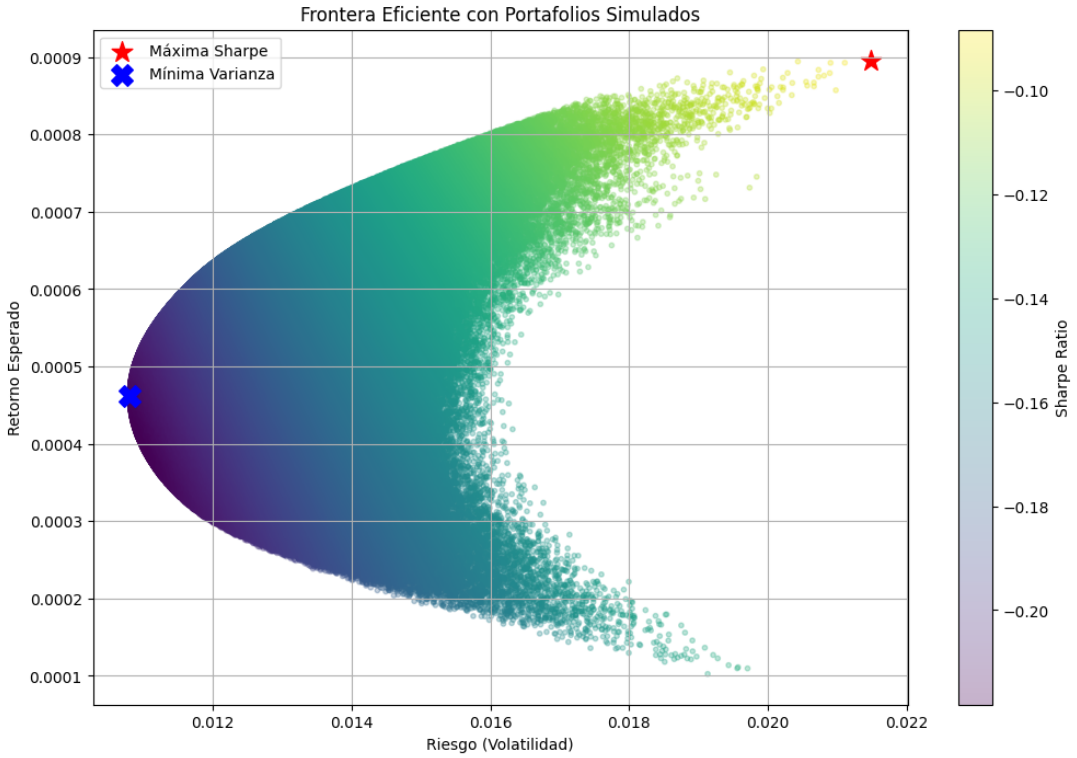
La LCC representa las combinaciones del activo libre de riesgo con el portafolio de mercado:

$$\mathbb{E}(R_P) = R_f + \left(\frac{\mathbb{E}(R_{PM}) - R_f}{\sigma_{PM}} \right) \cdot \sigma_P$$

donde:

- $E(R_P)$: rendimiento esperado del portafolio
- σ_P: desviación estándar del portafolio
- R_f: rendimiento del activo libre de riesgo
- $\mathbb{E}(R_{PM})$: rendimiento esperado del portafolio de mercado
- σ_{PM}: desviación estándar del portafolio de mercado

- Ejemplo:



Dicho lo anterior, podemos entonces describir lo que sigue:

Sean P el portafolio óptimo, PM el portafolio de mercado, A_i cualquier activo del mercado.

De lo anterior tenemos la siguiente ecuación que describe el portafolio como una composición de cierta cantidad de portafolio de mercado y de cualquier activo del mercado:

$$P = (1-x)PM + xA_i, \quad x \in \mathbb{R}$$

Con nuestro supuesto de que los rendimientos tienen una distribución normal. Nuestro activo A_i tiene asociado un $\mu_i = \mathbb{E}[R_i]$ y σ_i además de una $Cov(R_{PM}, R_i)$. Con R_{PM} y R_i los rendimiento del portafolio de mercado y el activo respectivamente. Que corresponderían a sus riesgos asociados.

Como buscamos conocer el rendimiento esperado del portafolio y su volatilidad, tenemos lo que sigue:

$$\mathbb{E}[R_p] = x\mathbb{E}[R_i] + (1-x)\mathbb{E}[R_{PM}]$$
$$\sigma_P = \left[(1-x)\sigma_{PM}^2 + x^2\sigma_i^2 + 2\mathrm{Cov}(R_{PM}, R_i)x(1-x)\right]^{\frac{1}{2}}$$

Desde ahora

$$Cov(R_{PM}, R_i) = \sigma_{PR,i} = \sigma_{i,PR}$$

Del supuesto de que el modelo busca mantener el equilibrio en el Mercado (entre la oferta y la demanda) que dicho de otra manera lleva a que el exceso en activo no puede ser diferente de cero. De lo anterior, nos interesa saber la recta tangente, es decir $x = 0$. Lo veremos de la siguiente forma:

$$\text{Tangente de } PM \text{ y } A_i = Tang(x) = \frac{\frac{\mathbb{E}[R_P]}{\partial x}}{\frac{\partial \sigma_P}{\partial x}}$$

Desarrollando cada parte obtenemos:

$$\frac{\mathbb{E}[R_P]}{\partial x} = \mathbb{E}[R_i] + \mathbb{E}[R_{PM}]$$
$$\frac{\partial \sigma_P}{\partial x} = \frac{1}{2}[(1-x)^2\sigma_{PM}^2 + x^2\sigma_i^2 + 2\text{Cov}(R_i, R_{PM})x(1-x)]^{\frac{1}{2}} \cdot [-2\sigma_{PM}^2 + 2x\sigma_i^2 + 2x\sigma_i^2 + 2\sigma_{i,PM}^2 - 4x\sigma_{i,PM}]$$

Como queremos valuar en $x = 0$:

$$\left.\frac{\frac{\partial \mathbb{E}[R_P]}{\partial x}}{\frac{\partial \sigma_P}{\partial x}}\right|_{x=0} = \frac{\frac{\mathbb{E}[R_i]-\mathbb{E}[R_{PM}]}{2\sigma_{i,PM}-2\sigma_{PM}^2}}{2\sigma_{PM}} = \frac{\mathbb{E}[R_i]-\mathbb{E}[R_{PM}]}{\sigma_{i,PM}-\sigma_{PM}^2} \cdot \sigma_{PM}$$

Teniendo la recta tangente a la Curva en el punto deseado, que es la curva de nuestro Portafolio de Mercado que es igual al portafolio óptimo. Ahora, igualando la expresión de la recta tangente con la pendiente de *Sharpe Ratio* línea de colocación de capital obtenemos lo siguiente:

$$\frac{\mathbb{E}(R_{PM}) - R_f}{\sigma_{PM}} = \frac{\mathbb{E}[R_i] - \mathbb{E}[R_{PM}]}{\sigma_{i,PM} - \sigma_{PM}^2} \cdot \sigma_{PM}$$

$$\begin{aligned}
\frac{\mathbb{E}(R_{PM}) - R_f}{\sigma_{PM}} &= \frac{\mathbb{E}[R_i] - \mathbb{E}[R_{PM}]}{\sigma_{i,PM} - \sigma_{PM}^2} \cdot \sigma_{PM} \\
\Leftrightarrow \frac{\mathbb{E}[R_i] - \mathbb{E}[R_{PM}]}{\sigma_{i,PM} - \sigma_{PM}^2} &= \frac{\mathbb{E}[R_{PM}] - R_f}{\sigma_{PM}^2} \\
\Leftrightarrow \mathbb{E}[R_i] &= \frac{\mathbb{E}[R_{PM}] - R_f}{\sigma_{PM}^2}(\sigma_{i,PM} - \sigma_{PM}^2) + \mathbb{E}[R_{PM}] \\
\mathbb{E}[R_i] &= \frac{\sigma_{i,PM} - \sigma_{PM}^2}{\sigma_{PM}^2}(\mathbb{E}[R_{PM}] - R_f) + \mathbb{E}[R_{PM}] \\
\mathbb{E}[R_i] &= \left[\frac{\sigma_{i,PM}}{\sigma_{PM}^2} - 1\right](\mathbb{E}[R_{PM}] - R_f) + \mathbb{E}[R_{PM}]
\end{aligned}$$

β_i se define como:

$$\beta_i = \frac{\sigma_{i,PM}}{\sigma_{PM}^2}$$

Finalmente usando la definición de β_i:

$$\begin{aligned}
\mathbb{E}[R_i] &= \beta_i(\mathbb{E}[R_{PM}] - R_f) + \mathbb{E}[R_{PM}] + R_f + \mathbb{E}[R_{PM}] \\
\Leftrightarrow \mathbb{E}[R_i] &= R_f + \beta_i(\mathbb{E}[R_{PM}] - R_f) \quad \text{¡¡Fórmula del CAPM!!!}
\end{aligned}$$

Donde:

- $\mathbb{E}[R_i]$:
  - Rendimiento esperado del activo i: Retorno que los inversores exigen por invertir en el activo, dado su riesgo.
- R_f:
  - Tasa libre de riesgo: Rendimiento de un activo sin riesgo (ej. CETES). Representa el mínimo retorno exigido.
- $\mathbb{E}[R_M] - R_f$:
  - Prima de riesgo del mercado: Compensación por asumir el riesgo del mercado.
- β_i:
  - Beta del Activo i: Proporcion lineal de la prima de riesgo del activo del mercado.
  - **Interpretación**:
    - $\beta_i = 1$: Movimiento acorde al mercado.
    - $\beta_i > 1$: Más volátil que el mercado.

* $\beta_i < 1$: Menos volátil que el mercado.

2. Demuestre que la **beta de un portafolio** de n activos es la media ponderada de las betas de cada uno de los activos donde cada peso está dado por la colocación de los activos en el portafolio. Notación: asuma que los n activos tienen **betas** dadas por $\beta_1, \beta_2, \ldots, \beta_n$ y que sus ponderaciones en el portafolio son $\omega_1, \omega_2, \ldots, \omega_n$ respectivamente.

Definimos nuestra fórmula del CPPM para un activo que es un portafolio:

$$E(R_p) = r_f + \beta_p[E(R_{pm}) - r_f]$$

Despejamos beta para nuestra Beta de portafolio (β_p), tenemos:

$$\beta_p = \frac{E(R_p) - r_f}{E(R_{pm}) - r_f}, \quad i = 1, \ldots, n$$

Ahora veamos que podemos ver al rendimiento de nuestro portafolio como combinación lineal de los rendimientos de activos con sus ponderaciones:

$$E(R_p) = \sum_{i=1}^{n} w_i E(R_i), \quad w_i = 1, \ldots, n$$

Entonces:

$$\beta_p = \frac{\sum_{i=1}^{n} w_i E(R_i) - r_f}{E(R_{pm}) - r_f} \quad \text{(Nota 1)}$$

Sabemos que:

$$E(R_i) = r_f + \beta_i[E(R_{pm}) - r_f]$$

Sustituyendo en la Nota 1:

$$\beta_p = \frac{\sum_{i=1}^{n} w_i \left(r_f + \beta_i[E(R_{pm}) - r_f]\right) - r_f}{E(R_{pm}) - r_f}$$

$$= \frac{\sum_{i=1}^{n} w_i r_f + w_i \beta_i[E(R_{pm}) - r_f] - r_f}{E(R_{pm}) - r_f} \quad \text{(Nota 2: Recordemos que:)} \sum_{i=1}^{n} w_i = 1$$

$$\sum_{i=1}^{n} w_i \beta_i[E(R_{pm}) - r_f] + w_i r_f$$
$$= \sum_{i=1}^{n} w_i \beta_i(E(R_{pm}) - r_f) + r_f \quad \text{Por la Nota 2.}$$

Sustituyendo en nuestra fórmula original:

$$\beta_p = \frac{\sum_{i=1}^{n} w_i \beta_i(E(R_{pm}) - r_f) + r_f - r_f}{E(R_{pm}) - r_f}$$

Simplificando:

$$\beta_p = \frac{\sum_{i=1}^{n} w_i \beta_i(E(R_{pm}) - r_f)}{E(R_{pm}) - r_f}$$

$$\therefore \beta_p = \sum_{i=1}^{n} w_i \beta_i$$

3. Demuestre o dé un contraejemplo justificado de las siguientes aseveraciones:

a) Si se consideran dos activos, A y B con betas β_A y β_B y rendimientos $E(R_A)$ y $E(R_B)$, respectivamente, siempre se puede formar con ellos un portafolio con riesgo sistemático igual que el del mercado y un rendimiento esperado equivalente a la tasa libre de riesgo.

b) Muestre qué relaciones se deben cumplir entre la tasa libre de riesgo r_F y los rendimientos $E(R_A)$ y $E(R_B)$ para que se garantice la existencia del portafolio con las características mencionadas en a) **sin ventas en corto**.

Sol:

a) Sea P el portafolio formado por los activos A y B:

$$P = w_A A + w_B B$$

Donde:

- $w_A + w_B = 1$
- $\beta_P = w_A \beta_A + w_B \beta_B = 1$ (riesgo sistemático igual al del mercado)
- $E(R_P) = w_A E(R_A) + w_B E(R_B) = r_F$ (rendimiento igual a la tasa libre de riesgo)

Sistema de ecuaciones:

1. $$w_A \beta_A + w_B \beta_B = 1$$
2. $$w_A E(R_A) + w_B E(R_B) = r_F$$

Resolviendo para w_A y w_B:

De la ecuación 1:

$$w_A = \frac{1 - w_B \beta_B}{\beta_A}$$

Sustituyendo en la ecuación 2:

$$\frac{1 - w_B \beta_B}{\beta_A} E(R_A) + w_B E(R_B) = r_F$$

Despejando w_B:

$$\frac{1 - w_B \beta_B}{\beta_A} E[R_A] = r_F - w_B E[R_B]$$

$$1 - w_B \beta_B = \frac{\beta_A}{E[R_A]} \cdot r_f - \frac{\beta_A}{E[R_A]} \cdot w_B E[R_B]$$

$$1 - w_B \beta_B + w_B \frac{E[R_B]}{E[R_A]} \beta_A = \frac{\beta_A}{E[R_A]} r_F$$

$$w_B \left[\frac{E[R_B]}{E[R_A]} \beta_A - \beta_B \right] = \frac{\beta_A}{E[R_A]} r_F - 1$$

Nota:

$$\frac{E[R_B]}{E[R_A]}\beta_A - \beta_B = \frac{E[R_B]\beta_A - E[R_A]\beta_B}{E[R_A]}$$

$$\frac{\beta_A}{E[R_A]}r_F - 1 = \frac{\beta_A r_F - E[R_A]}{E[R_A]}$$

Finalmente

$$w_B = \frac{\frac{\beta_A r_F - E[R_A]}{E[R_A]}}{\frac{E[R_B]\beta_A - E[R_A]\beta_B}{E[R_A]}}$$

$$w_B = \frac{\beta_A r_F - E(R_A)}{\beta_A E(R_B) - \beta_B E(R_A)}$$

Contraejemplo:
Si $\beta_A = \beta_B = 0.5$, $E(R_A) = E(R_B) = 5\%$, $r_F = 3\%$:

$$w_B = \frac{0.5 \times 3 - 5}{0.5 \times 5 - 0.5 \times 5} = \frac{-3.5}{0}$$

División por cero $\Rightarrow$ No existe solución.

Por lo tanto, no siempre es posible construir dicho portafolio.

b) Para evitar ventas en corto, se requiere $0 \leq w_A, w_B \leq 1$.

Condiciones:

Del inciso anterior sabemos que:

$$w_B = \frac{\beta_A r_F - E[R_A]}{\beta_A E[R_B] - \beta_B E[R_A]}$$

$$w_A = \frac{1 - w_B\beta_B}{\beta_A}$$

Para que exista dicho portafolio, se deben cumplir las siguientes consideraciones:

1. **Condición de no singularidad**:

$$\beta_A E[R_B] - \beta_B E[R_A] \neq 0 \quad \text{y} \quad \beta_A \neq 0$$

2. **Restricción de no ventas en corto** (pesos positivos):

- $w_A > 0$
- $w_B > 0$

Caso 1: Numerador y denominador positivos

Tanto numerador como denominador debe cumplir:

$$\beta_A E[R_B] - \beta_B E[R_A] > 0 \quad \text{y} \quad \beta_A r_F - E[R_A] > 0$$

Además:

$$1 - w_B\beta_B > 0$$

$$\beta_A > 0$$

De lo anterior obtenemos lo anterior:

$$r_F > \frac{E[R_A]}{\beta_A}$$

Caso 2: tanto numerador como denominador de w_A, w_B deben ser negativos, es decir $\beta_A < 0$, $ β_A E[R_B] - β_B E[R_A] <0 $, $ 1-w_B β_B $, $\beta_A r_F - E[R_A] < 0$, de esta última desigualdad se tiene que:

$$\beta_A r_F < E[R_A] \Rightarrow r_F > \frac{E[R_A]}{\beta_A} \quad \text{pues } \beta_A < 0$$

Por último la última relación que incluya a r_F sería cuando:

$$w_B = \frac{\beta_A r_F - E[R_A]}{\beta_A E[R_B] - \beta_B E[R_A]} > 0$$

si y sólo si

$$\beta_A r_F - E[R_A] > \beta_A E[R_B] - \beta_B E[R_A]$$

si y sólo si

$$r_F > \frac{[\beta_A E[R_B] - \beta_B E[R_A]] + E[R_A]}{\beta_A}$$

Así, hemos finalizado las relaciones que debe cumplir para garantiazar la existencia del portafolio con las características mencionadas en a)

4. . En un mercado donde se cumplen los supuestos de CAPM, el activo de mercado tiene una rentabilidad esperada de 14.5% anual, la tasa libre de riesgo es de 5.2% anual, y la volatilidad de la cartera de mercado es de 21.2% anual. Suponga que usted dispone de 1 millón de patrimonio para invertir.

a) Dibuje la Security Market Line de este mercado

b) Suponga que hay un activo A cuyo rendimiento tiene covarianza con el rendimiento de mercado de 17.5% y rendimiento esperado observado de 16%. Diga si el activo está bien valuado, subvaluado o sobrevaluado. Determine el alfa del activo

c) Suponga que hay un activo B cuyo rendimiento tiene correlación con el rendimiento de mercado de -0.52 y una varianza del 13%. Diga si el activo está bien valuado y determine el alfa del activo.

d) Considere que el activo A paga un dividendo de $8 anual, ¿Cuál es el precio que tendría la acción bajo el modelo CAPM?

e) Considere que el activo A paga un dividendo de $15 anual y que crece 5% anual, ¿Cuál es el precio que tendría la acción bajo el modelo CAPM?

f) Considere que el activo A paga un dividendo de $7 anual y que espera venderse en 5 años en $75, ¿Cuál es el precio que tendría la acción bajo el modelo CAPM?

Sol: a) Security Market Line (SML)

Datos:

- Rentabilidad esperada del activo de mercado ($E[R_m]$) = **14.5% anual**
- Tasa libre de riesgo (R_f) = **5.2% anual**
- Volatilidad del mercado (σ_m) = **21.2% anual**
- Patrimonio disponible para invertir = **$1,000,000**

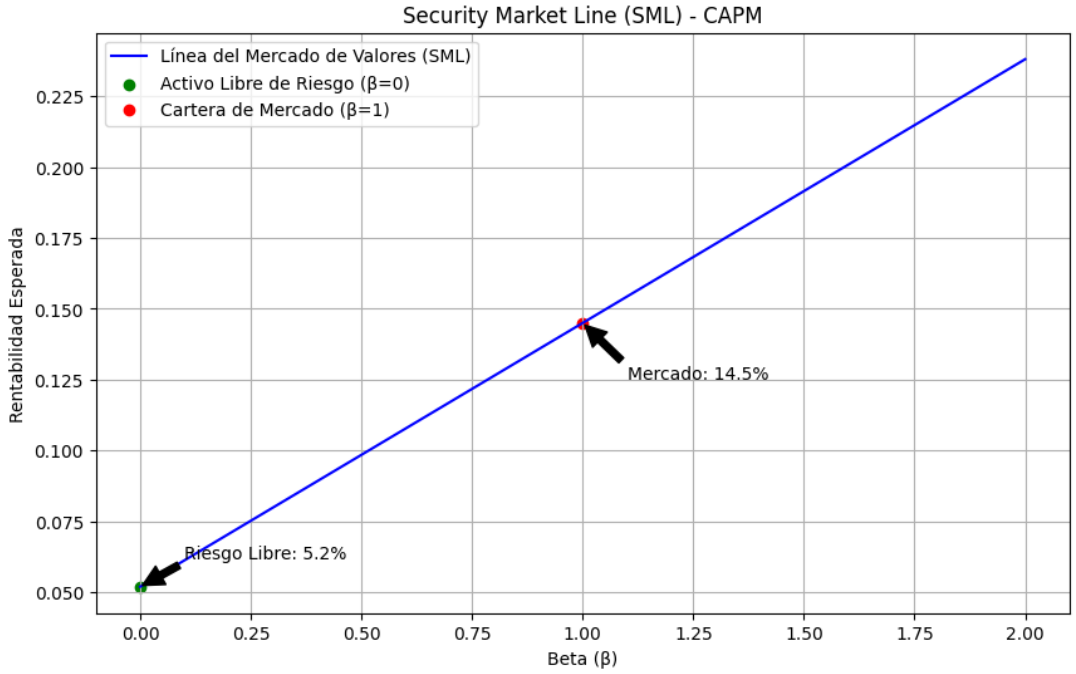
Solución:

1. **Construcción de la SML**: Representa la relación lineal entre el riesgo sistemático (β) y el rendimiento esperado:
$$E[R_i] = R_f + \beta_i(E[R_m] - R_f)$$

2. **Ecuación de la SML**:

$$E[R_i] = 0.052 + \beta_i(0.145 - 0.052) = 0.052 + 0.093\beta_i$$

3. **Gráfica**:



b) Valuación del Activo A

Datos:

- Covarianza con el mercado ($\alpha_{A,m}$) = 17.5%

- Rendimiento esperado observado (R_A) = 16%

Procedimiento:

1. **Beta del activo**:

$$\beta_A = \frac{0.175}{(0.212)^2} \approx 3.893$$

2. **Rendimiento CAPM**:

$$E[R_A] = 0.052 + 3.893(0.093) \approx 41.41\%$$

3. **Comparación**:

$$16\% < 41.4\% \Rightarrow \text{Sobrevaluado}$$

4. **Alfa**:

$$\alpha_A = 0.16 - 0.414 \approx -25.4\%$$

c) Valuación del Activo B

Datos:

- Rendimiento esperado observado (R_B) = 15.8%
- Correlación con el mercado ($\rho_{B,m}$) = -0.52
- Varianza del activo B (σ_B^2) = 13%

Procedimiento:

1. **Beta del activo**:

$$\alpha_B = \sqrt{0.13} \approx 0.3606$$

$$\beta_B = -0.52 \left(\frac{0.3606}{0.212} \right) \approx -0.884$$

2. **Rendimiento CAPM**:

$$E[R_B] = 0.052 + (-0.884)(0.093) \approx -3.02\%$$

3. **Alfa**:

$$\alpha_B = 0.158 - (-0.0302) \approx 18.82\% \Rightarrow \text{Subvaluado}$$

d) Precio con Dividendo Constante

Usamos el modelo de descuento de dividendos constantes, que establece que así como una perpetuidad sigue la siguiente fórmula

Modelo:

$$P = \frac{D}{k}$$

Datos:

- Dividendo anual $(D) = \$8$
- $k = E[R_A] \approx 41.41\%$

Cálculo:

Como los dividendos se pagarán en teoría "para siempre" (el tiempo que se mantenga en posesión el activo) seguimos el siguiente esquema de pagos traidos a valor presente:

$$P = \frac{D}{1+k} + ... + \frac{D}{(1+k)^n} + ... = \sum_{n=1}^{\infty} \frac{D}{(1+k)^n} = \frac{D}{k} = \frac{8}{0.4141173} \approx \$19.31819794 \quad \text{(Usando } k = E[R_A]\text{)}$$

Por lo tanto el precio actual de la acción bajo el modelo CAPM es aproximadamente $\boxed{\$19.31}$.

e) Precio con Dividendo Creciente

Modelo:

Usamos el modelo de descuento de dividendos con crecimiento constante, que establece que así como una perpetuidad con crecimientos crecientes sigue el siguiente esquema de pagos:

$$P_0 = \frac{D_1}{(1+\mathbb{E}[R_A])} + \frac{D_1(1+g)}{(1+\mathbb{E}[R_A])^2} + \frac{D_1(1+g)^2}{(1+\mathbb{E}[R_A])^3} + ... \quad \text{(La fórmula de una perpetuidad con crecimiento constante)}$$

$$P_0 = \sum_{n=1}^{\infty} \frac{D_1(1+g)^{n-1}}{(1+\mathbb{E}[R_A])^n}$$

$$P_0 = \frac{D_1}{\mathbb{E}[R_A] - g}$$

Datos:

- Dividendo próximo año $(D_1) = \$15$
- Tasa crecimiento $(g) = 5\%$
- $k = 41.41\%$

Cálculo:

$$P = \frac{15}{0.414 - 0.05} \approx \$41.19$$

Por lo tanto el precio actual de la acción bajo el modelo CAPM es aproximadamente $\boxed{\$41.19}$.

f) Precio con Venta Futura

Datos: - Dividendo anual (D) = \$7 - Precio venta en 5 años (P_5) = \$75 - $r = 41.41\%$

Cálculos:

1. Valor presente de dividendos:

$$VP_D = 7 \cdot \left(\frac{1 - (1.414)^{-5}}{0.414} \right) \approx \$13.91$$

2. Valor presente de precio venta:

$$VP_{P_5} = \frac{75}{(1.414)^5} \approx \$13.26$$

3. **Precio actual**:

$$P_0 = 13.91 + 13.26 \approx \$27.17$$

5. Considere la acción de GAP.MX y obtenga un histórico de 5 años de sus precios tomando el precio del 21 de mayo de 2025 como el último dato:

a) Obtenga los rendimientos semanales de GAP.MX.
b) Obtenga la tasa CETE 28 días al 21 de mayo de 2025 y calcule la tasa equivalente comparable con las tasas de rendimiento semanales de GAP.MX.
c) Obtenga la serie histórica del IPC para las mismas fechas que GAP.MX y calcule los rendimientos semanales análogos.
d) Use la información obtenida para modelar CAPM y proporcione los valores de beta (β) y alfa (α) del activo.
e) Dibuje la SML (Línea del Mercado de Valores) de este mercado y ubique su activo en el mismo plano, junto con los elementos obtenidos.
f) Realice un modelo de regresión (como un modelo APT de un solo factor) para la variable prima de riesgo de GAP.MX explicada por la prima de riesgo del Mercado, y proporcione sus betas:
- β_0 (intercepto)
- β_1 (coeficiente regresor).

g) Compare el alfa (α) obtenido en **e)** y el β_0 obtenido en **f)**.

[1]:
```
from scipy.stats import norm
from scipy.stats import kurtosis
from scipy.stats import skew
from scipy.stats import chi2
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf
from sklearn.linear_model import LinearRegression
import os
np.set_printoptions(suppress=True)  # Suprime la notación científica
pd.set_option('display.float_format', '{:.10f}'.format)  # 10 decimales
```

[2]:
```
tickers = {
    'GAPB.MX': 'GAP',
    '^MXX': 'IPC'
}

# Fechas de análisis
```

```
start_date = '2020-05-21'
end_date = '2025-05-22'

# Crear directorio para guardar los archivos si no existe
output_dir = 'stock_data'
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

# Descargar y guardar cada ticker individualmente
for yfinance_ticker, friendly_name in tickers.items():
    try:
        print(f"Descargando datos para {friendly_name}...")

        # Descargar datos
        data = yf.download(yfinance_ticker, start=start_date, end=end_date,␣
 ↪auto_adjust=False)['Close']
        data.columns = ['Close']  # Renombrar columna

        # Guardar en CSV
        filename = f"{output_dir}/{friendly_name.replace(' ', '_')}.csv"
        data.to_csv(filename)
        print(f"Datos guardados en {filename}\n")

    except Exception as e:
        print(f"Error al descargar {friendly_name}: {str(e)}\n")

print("Proceso completado.")
```

```
Descargando datos para GAP...
[**********************100%**********************]  1 of 1 completed
Datos guardados en stock_data/GAP.csv

Descargando datos para IPC...
[**********************100%**********************]  1 of 1 completed
Datos guardados en stock_data/IPC.csv

Proceso completado.
```

[3]:
```
data_raw_GAP = pd.read_csv(r"C:\Users\jon\OneDrive\Documentos\Proyecto MCF\Proyecto␣
 ↪CAPM\Ejercicio 5\stock_data\GAP.csv")
data_raw_GAP
```

```
           Date          Close
0     2020-05-21 138.3099975586
1     2020-05-22 138.5599975586
2     2020-05-25 141.0599975586
3     2020-05-26 150.2400054932
4     2020-05-27 150.9299926758
...          ...            ...
1254  2025-05-15 435.5499877930
1255  2025-05-16 431.0100097656
1256  2025-05-19 443.1199951172
1257  2025-05-20 439.5000000000
1258  2025-05-21 440.3399963379
```

```
[1259 rows x 2 columns]
```

Parte a): Rendimientos semanales de GAP.MX

[4]:
```
data_GAP = data_raw_GAP.copy()
data_GAP['Date'] = pd.to_datetime(data_GAP['Date'])
data_GAP = data_GAP.sort_values('Date')
```

[5]:
```
data_GAP['GAPB_rend'] = np.log(data_GAP['Close']/data_GAP['Close'].shift(1))
data_GAP.dropna(inplace=True)
data_GAP.reset_index(drop= True, inplace=True)
data_GAP
```

```
           Date          Close     GAPB_rend
0    2020-05-22 138.5599975586  0.0018059022
1    2020-05-25 141.0599975586  0.0178818873
2    2020-05-26 150.2400054932  0.0630487376
3    2020-05-27 150.9299926758  0.0045820526
4    2020-05-28 148.8699951172 -0.0137426956
...         ...            ...           ...
1253 2025-05-15 435.5499877930  0.0264045013
1254 2025-05-16 431.0100097656 -0.0104782579
1255 2025-05-19 443.1199951172  0.0277092884
1256 2025-05-20 439.5000000000 -0.0082028856
1257 2025-05-21 440.3399963379  0.0019094303

[1258 rows x 3 columns]
```

[6]:
```
rendimientos_semanales_gapb = data_GAP.resample('W', on='Date')['GAPB_rend'].sum().
 ↪dropna()
rendimientos_semanales_gapb
```

```
Date
2020-05-24    0.0018059022
2020-05-31    0.0549708618
2020-06-07    0.1513460025
2020-06-14   -0.0718026434
2020-06-21    0.0363023064
                  ...
2025-04-27    0.0710150816
2025-05-04    0.0066808542
2025-05-11    0.0436389326
2025-05-18    0.0211022808
2025-05-25    0.0214158331
Freq: W-SUN, Name: GAPB_rend, Length: 262, dtype: float64
```

Parte b) tasa CETE anual a equivalente semanal (interés compuesto)

Obtenemos que la tasa CETE a 28 días al 21 de Mayo de 2025 fue de 8.15%.

[7]:
```
tasa_cete_anual = 0.815
tasa_semanal = (1 + tasa_cete_anual) ** (7 / 365) - 1
print(f"\nTasa CETE equivalente semanal: {tasa_semanal:.6f}")
```

```
Tasa CETE equivalente semanal: 0.011497
```

Parte c) Histórico de precios del IPC

[8]:
```
data_raw_IPC = pd.read_csv(r"C:\Users\jon\OneDrive\Documentos\Proyecto MCF\Proyecto␣
↪CAPM\Ejercicio 5\stock_data\IPC.csv")
data_raw_IPC
```

```
            Date            Close
0     2020-05-21 35560.7617187500
1     2020-05-22 35784.4218750000
2     2020-05-25 35832.7695312500
3     2020-05-26 36206.8593750000
4     2020-05-27 36889.9609375000
...          ...              ...
1254  2025-05-15 57959.7187500000
1255  2025-05-16 57987.1406250000
1256  2025-05-19 58493.3906250000
1257  2025-05-20 58311.1484375000
1258  2025-05-21 58568.0117187500

[1259 rows x 2 columns]
```

Obtenemos los rendimientos diarios y posteriormente los rendimientos semanales

[9]:
```
data_IPC = data_raw_IPC.copy()
data_IPC['IPC_rend'] = np.log(data_IPC['Close']/data_IPC['Close'].shift(1))
data_IPC.dropna(inplace=True)
data_IPC.reset_index(drop= True, inplace=True)
data_IPC
```

```
            Date            Close      IPC_rend
0     2020-05-22 35784.4218750000  0.0062698245
1     2020-05-25 35832.7695312500  0.0013501692
2     2020-05-26 36206.8593750000  0.0103857616
3     2020-05-27 36889.9609375000  0.0186908663
4     2020-05-28 36508.1406250000 -0.0104041862
...          ...              ...           ...
1253  2025-05-15 57959.7187500000  0.0054457687
1254  2025-05-16 57987.1406250000  0.0004730076
1255  2025-05-19 58493.3906250000  0.0086924945
1256  2025-05-20 58311.1484375000 -0.0031204667
1257  2025-05-21 58568.0117187500  0.0043953720

[1258 rows x 3 columns]
```

Los rendimientos semanales de IPC serían:

[10]:
```
data_IPC['Date'] = pd.to_datetime(data_IPC['Date'])
data_IPC = data_IPC.sort_values('Date')
data_IPC
```

```
           Date            Close      IPC_rend
0    2020-05-22 35784.4218750000  0.0062698245
1    2020-05-25 35832.7695312500  0.0013501692
2    2020-05-26 36206.8593750000  0.0103857616
3    2020-05-27 36889.9609375000  0.0186908663
4    2020-05-28 36508.1406250000 -0.0104041862
...         ...              ...           ...
1253 2025-05-15 57959.7187500000  0.0054457687
1254 2025-05-16 57987.1406250000  0.0004730076
1255 2025-05-19 58493.3906250000  0.0086924945
1256 2025-05-20 58311.1484375000 -0.0031204667
1257 2025-05-21 58568.0117187500  0.0043953720
```

```
[1258 rows x 3 columns]
```

[11]:
```
rendimientos_semanales_ipc = data_IPC.resample('W', on='Date')['IPC_rend'].sum().
↪dropna()
rendimientos_semanales_ipc
```

```
Date
2020-05-24    0.0062698245
2020-05-31    0.0094096645
2020-06-07    0.0753071089
2020-06-14   -0.0331201950
2020-06-21    0.0190687627
                  ...
2025-04-27    0.0674867817
2025-05-04   -0.0161403271
2025-05-11    0.0131573732
2025-05-18    0.0250752093
2025-05-25    0.0099673998
Freq: W-SUN, Name: IPC_rend, Length: 262, dtype: float64
```

Sean:

X = variable independiente = IPC (**pues es un indicador del comportamiento del Mercado Accionario Mexicano**)

Y = variable dependiente = GAPB **(puesto que el comportamiente de este activo es en función al Mercado Accionario Mexicano)**

Parte d) Usar CAPM y obtener beta y alfa

[12]:
```
from sklearn.linear_model import LinearRegression

# Renombramos las series para mayor claridad
gap = rendimientos_semanales_gapb.rename("gap")
ipc = rendimientos_semanales_ipc.rename("ipc")

# Alineamos las fechas
data = pd.concat([gap, ipc], axis=1).dropna()

# Calculamos exceso de rendimiento sobre tasa libre de riesgo
rf = tasa_semanal
data["exceso_GAP"] = data["gap"] - rf
data["exceso_IPC"] = data["ipc"] - rf

# Preparamos los datos
X = data[["exceso_IPC"]].values  # Variable independiente (prima de riesgo del␣
↪mercado)
y = data["exceso_GAP"].values    # Variable dependiente (prima de riesgo de GAP)

# Creamos y ajustamos el modelo
modelo = LinearRegression(fit_intercept=True)
modelo.fit(X, y)

# Obtenemos los coeficientes
beta_0 = modelo.intercept_       # Equivalente al alpha de CAPM
beta_1 = modelo.coef_[0]         # Beta del activo

# Mostramos resultados
print(f"Intercepto (α de CAPM): {beta_0:.6f}")
```

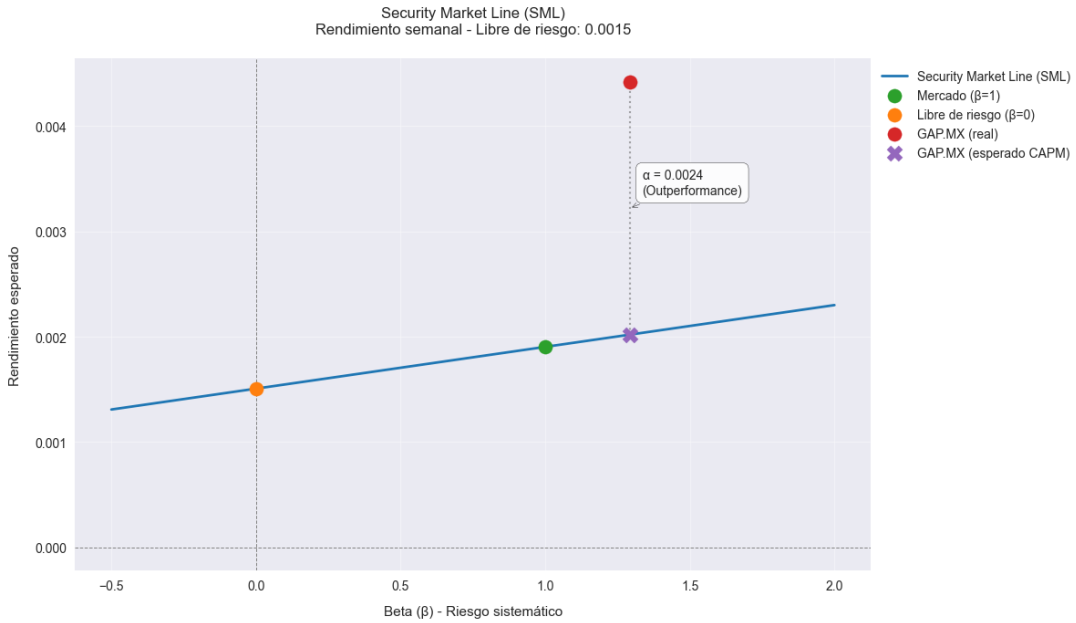
```
print(f"Coeficiente (β del activo): {beta_1:.6f}")
```

```
Intercepto (α de CAPM): 0.005315
Coeficiente (β del activo): 1.291857
```

Parte e) Graficar la SML (Security Market Line)

[13]:
```
# Configuración de estilo
plt.style.use('seaborn-v0_8')
plt.rcParams['figure.facecolor'] = 'white'

# Supuestos del mercado (tus variables)
rf = float(rf)  # tasa libre de riesgo semanal
rm = float(data['ipc'].mean()) # rendimiento esperado del mercado semanal

# Beta y rendimiento real de GAP (tus variables)
beta_gap = beta_1
rend_real_gap = data["gap"].mean()

# Crear rango de betas para la SML
betas = np.linspace(-0.5, 2.0, 100)
rend_sml = rf + betas * (rm - rf)

# Punto del rendimiento esperado según CAPM
rend_esperado_gap = rf + beta_gap * (rm - rf)

# Diferencia entre rendimiento real y esperado
alpha_gap = rend_real_gap - rend_esperado_gap

# Crear figura
plt.figure(figsize=(12, 7))

# Graficar SML
plt.plot(betas, rend_sml, label="Security Market Line (SML)", color="#1f77b4",␣
↪linewidth=2)

# Puntos importantes
plt.scatter(1, rm, color="#2ca02c", s=120, label="Mercado (β=1)", zorder=5)
plt.scatter(0, rf, color="#ff7f0e", s=120, label=f"Libre de riesgo (β=0)", zorder=5)
plt.scatter(beta_gap, rend_real_gap, color="#d62728", s=120, label="GAP.MX (real)",␣
↪zorder=5)
plt.scatter(beta_gap, rend_esperado_gap, color="#9467bd", marker="X", s=150,␣
↪label="GAP.MX (esperado CAPM)", zorder=5)

# Línea que conecta real vs esperado
plt.plot([beta_gap, beta_gap], [rend_esperado_gap, rend_real_gap],
         color='gray', linestyle=':', linewidth=1.5, alpha=0.7)

# Anotaciones
plt.annotate(f'α = {alpha_gap:.4f}\n(Outperformance)' if alpha_gap > 0 else f'α =␣
↪{alpha_gap:.4f}\n(Underperformance)',
             xy=(beta_gap, (rend_real_gap + rend_esperado_gap)/2),
             xytext=(10, 20 if alpha_gap > 0 else -20),
             textcoords='offset points',
             ha='left', va='center',
             bbox=dict(boxstyle='round,pad=0.5', fc='white', alpha=0.9),
             arrowprops=dict(arrowstyle='->'))
```

```
# Ejes y formato
plt.title(f"Security Market Line (SML)\nRendimiento semanal - Libre de riesgo: {rf:.
 ↪4f}", pad=20)
plt.xlabel("Beta (β) - Riesgo sistemático", labelpad=10)
plt.ylabel("Rendimiento esperado", labelpad=10)
plt.axhline(y=0, color='gray', linestyle='--', linewidth=0.7)
plt.axvline(x=0, color='gray', linestyle='--', linewidth=0.7)
plt.grid(True, alpha=0.4)

# Leyenda y ajustes
plt.legend(loc='upper left', bbox_to_anchor=(1, 1))
plt.tight_layout()

# Mostrar gráfico
plt.show()
```



Inciso f) Usaremos el modelo de regresión tipo APT con un solo factor

[14]:
```
from sklearn.linear_model import LinearRegression

# Prima de riesgo
prima_gap = data["exceso_GAP"].values.reshape(-1, 1)  # Variable dependiente
prima_mercado = data["exceso_IPC"].values.reshape(-1, 1)  # Variable independiente

# Modelo APT (equivalente al CAPM en este caso)
modelo_apt = LinearRegression(fit_intercept=True)
modelo_apt.fit(prima_mercado, prima_gap)

# Obtenemos los coeficientes
beta_0_apt = modelo_apt.intercept_[0]  # a (intercepto)
beta_1_apt = modelo_apt.coef_[0][0]    # β (coeficiente)

# Mostramos resultados
print("=== Modelo APT (un factor) ===")
print(f"Intercepto (β): {beta_0_apt:.6f}")
```

```
print(f"Coeficiente (β): {beta_1_apt:.6f}")
```

```
=== Modelo APT (un factor) ===
Intercepto (β): 0.005315
Coeficiente (β): 1.291857
```

Inciso g) Comparación entre alfa del CAPM y β del modelo APT

[15]:
```
# Comparación con los resultados CAPM anteriores
print("\n=== Comparación ===")
print(f"Diferencia entre α(CAPM) y β(APT): {beta_0 - beta_0_apt:.6f}")
print(f"Diferencia entre β(CAPM) y β(APT): {beta_1 - beta_1_apt:.6f}")
```

```
=== Comparación ===
Diferencia entre α(CAPM) y β(APT): 0.000000
Diferencia entre β(CAPM) y β(APT): 0.000000
```

Notemos que si definimos un modelo de APT de un solo factor, donde ese factor sea la prima de riesgo de mercado, estaríamos regresando a CAPM, es decir, si $f_1 = \mathbf{E}(R_M) - r_f$ entonces:

$$f_1 = \mathbf{E}(R_M) - r_f \Rightarrow \mathbf{E}(R_i)_{APT} = \beta_0 + \beta_1 \mathbf{E}(f_1) = \beta_0 + \beta_1(\mathbf{E}(R_M) - r_f) = \mathbf{E}(R_i)_{CAPM}$$

Por lo tanto es natural que la diferencia entre ambos sea cero

6. Considere que se le invita a participar en una constructora de nombre CONMEX en un proyecto cuyos flujos anuales para usted serían son los siguientes:

Año	Flujo de Efectivo (MXN)
0	-1,600,000
1	200,000
2	350,000
3	400,000
4	500,000
5	1,000,000

Según la tabla anterior, este proyecto asume que usted invertiría 1,600,000 el día de hoy y a cambio recibiría flujos de efectivo como los mostrados durante los siguientes cinco años.

a) Calcule la beta del sector construcción en México usando históricos de información de 3, 5 o 10 años justificando racionalmente su elección. Para la tasa libre de riesgo use la tasa libre de riesgo Cete a 28 días

b) Con la información anterior calcule el VPN de este proyecto de inversión.

c) Considere la siguiente curva de tasas soberanas:

Plazo (años)	Tasa Nominal Anual
Hasta 1 año	10.64%
>1 año y ≤ 2	10.85%
>2 años y ≤ 3	10.95%
>3 años	11.15%

Inciso a)

Si considera que quiere recibir los mismos flujos que le daría el proyecto anterior en los mismos tiempos, pero quiere comparar con invertir en las tasas soberanas sin riesgo, determine justificando numéricamente, entonces si invierte o no en el proyecto anterior o mejor invierte a las tasas soberanas.

[16]:
```
import pandas as pd
import numpy as np
```

```
np.set_printoptions(precision = 8, suppress = True)
data= pd.read_csv(r"C:\Users\rubrh\OneDrive\Escritorio\Metodos cuantitativos en␣
↪finanzas\Ultima_tarea_examen\Datos históricos S&P_BMV IPC.csv")
data['rend_diarios_ipc']=np.log(data['Ipc']/data['Ipc'].shift(1)) #rendimiento␣
↪diario del IPC
data.dropna(inplace=True)
data.reset_index(drop= True, inplace=True)
data
```

```
          Fecha       Ipc  rend_diarios_ipc
0    25.05.2022  51717.07          0.008018
1    26.05.2022  52143.00          0.008202
2    27.05.2022  52463.55          0.006129
3    30.05.2022  52162.08         -0.005763
4    31.05.2022  51752.53         -0.007882
..          ...       ...               ...
751  19.05.2025  58493.39          0.008692
752  20.05.2025  58311.15         -0.003120
753  21.05.2025  58568.01          0.004395
754  22.05.2025  57894.82         -0.011561
755  23.05.2025  58410.37          0.008866

[756 rows x 3 columns]
```

[17]:
```
rend_esperada_ipc= data['rend_diarios_ipc'].mean() #rendimiento esperado diario del␣
↪IPC
rend_esperada_ipc
```

```
0.00017159251461819008
```

[18]:
```
data2= pd.read_csv(r"C:\Users\rubrh\OneDrive\Escritorio\Metodos cuantitativos en␣
↪finanzas\Ultima_tarea_examen\Datos históricos CEMEXCPO (1).csv")
data2['rend_diarios_cemex']=np.log(data2['cemex']/data2['cemex'].shift(1))␣
↪#rendimiento diario de Cemex
data2.dropna(inplace=True)
data2.reset_index(drop= True, inplace=True)
data2
```

```
          Fecha  cemex  rend_diarios_cemex
0    25.05.2022   8.59            0.022367
1    26.05.2022   8.78            0.021878
2    27.05.2022   9.04            0.029183
3    30.05.2022   9.19            0.016457
4    31.05.2022   9.26            0.007588
..          ...    ...                 ...
748  19.05.2025  13.36            0.000000
749  20.05.2025  13.54            0.013383
750  21.05.2025  13.64            0.007358
751  22.05.2025  13.59           -0.003672
752  23.05.2025  13.83            0.017506

[753 rows x 3 columns]
```

[19]:
```
#Une ambos DataFrames por la columna fecha.

#Si alguna fecha está en uno pero no en el otro, igual aparece pero la columna␣
↪correspondiente tendrá un valor 0
```

```
dataconbinado=pd.merge(data,data2,on='Fecha', how='outer')
dataconbinado
```

	Fecha	Ipc	rend_diarios_ipc	cemex	rend_diarios_cemex
0	25.05.2022	51717.07	0.008018	8.59	0.022367
1	26.05.2022	52143.00	0.008202	8.78	0.021878
2	27.05.2022	52463.55	0.006129	9.04	0.029183
3	30.05.2022	52162.08	-0.005763	9.19	0.016457
4	31.05.2022	51752.53	-0.007882	9.26	0.007588
..	...	...	...	...	...
751	19.05.2025	58493.39	0.008692	13.36	0.000000
752	20.05.2025	58311.15	-0.003120	13.54	0.013383
753	21.05.2025	58568.01	0.004395	13.64	0.007358
754	22.05.2025	57894.82	-0.011561	13.59	-0.003672
755	23.05.2025	58410.37	0.008866	13.83	0.017506

[756 rows x 5 columns]

[20]:
```
#calculamos la covarinza entre los rendimientos diarios del IPC y de Cemex
covarianza = dataconbinado[['rend_diarios_ipc', 'rend_diarios_cemex']].cov().
↪iloc[0, 1]
covarianza
```

0.00012690439375858203

[21]:
```
#calcualamos la variancia de los rendimientos diarios del IPC
varianza = dataconbinado['rend_diarios_ipc'].var()
varianza
```

9.739923215537634e-05

[22]:
```
#calculo de la beta de la accion de cemex respecto al indice IPC
beta = covarianza / varianza
print(f"La beta de la acción de Cemex respecto al índice IPC es: {beta:.4f}")
```

La beta de la acción de Cemex respecto al índice IPC es: 1.3029

Inciso b)

Como vimos que "CONMEX" no cotizaba, utilizamos "CEMEX" y el índice del IPC

En base a este gráfico, elegimos la temporalidad de 5 años para estar tres años para dejar fuera las pérdidas del 2020 debido al COVID-19. Tomamos los datos con un histórico de 3 años, desde el 23 de Mayo del 2022 hasta el 25 de Mayo del 2025.

Además de eso escogimos la menor temporalidad para tener datos recientes (pero también tener gran densidad de datos) ya que lo que nos importa es ver cómo se ha comportado CEMEX respecto al IPC en los últimos años.

Una vez calculada la beta de CEMEX en el inciso a) tenemos que $\beta_0 = 1.30$, siendo una beta agresiva, es decir, el activo CEMEX sigue fuertemente la tendencia del IPC. Una beta mayor a 1 también nos indica un riesgo sistemático mayor, notamos que es el IPC tiene una tendencia alcista. Lo que favorece tener una beta mayor a 1

Ahora bien el incisio b) nos pide calcular el VPN ocupando la tasa CETE de hoy 23 de mayo del 2025 la cual es 8.15% a 28 dias. Entonces calculemos el rendimiento esperado de CEMX bajo CAPM

$$E[R_{CEMEX}] = r_F + \beta_C[E[R_{IPC} - r_F]]$$

en Python calculamos $E[R_{IPC}]$ ocupando la media de los rendimientos diarios entonces

$$E[R_{IPC}] = 0.00017 = 0.017\% \quad \text{diario}$$

Calculamos la tasa libre de riesgo diaria, entonces la tasa diaria esta dada por

$$(1 + 8.15\%(28/365)) = (1 + i(28))$$

$$\Rightarrow i = \frac{8.15\%}{365} = 0.000223 = 0.023\% \quad \text{diaria simple}$$

Ahora si

$$\begin{aligned} &= 0.000223 + 1.30(0.00017 - 0.000223) \\ &= 0.000152 \end{aligned}$$

Al resultado anterior lo vamos a tomar como una tasa diaria compuesta, encontremso una tasa equivalente anual:

$$(1 + 0.000152)^{365} - 1 = 0.05704 = 5.704\% \quad \text{anual}$$

Con esta tasa anual podemos calcular el VPN de los flujos de efectivo del proyecto de inversión como sigue:

$$VPN = -1,600,00 + 200,000v + 350,000v^2 + 400,000v^3 + 500,000v^4 + 1,000,000v^5$$

$$\begin{aligned} 2000,000(1.05704)^{-1} &= 189,206.9865 \\ 350,000(1.05704)^{-2} &= 313,243.7328 \\ 400,000(1.05704)^{-3} &= 338,673.7299 \\ 500,000(1.05704)^{-4} &= 400,496.4741 \\ 1,000,000(1.05704)^{-5} &= 757,767.3097 \\ \therefore VPN &= -1,600,000 + 189,206.9865 + 313,243.7328 + 338,673.7299 + 400,496.4741 + 757,767.3097 \\ &= 399,388.233 \end{aligned}$$

Inciso c)

Queremos que $C_i = F_I(1+r_i)^{-i}$, donde r_i es la tasa efectiva anual, F_i es el i-ésimo flujo, c_i es el i-ésimo capital requerido $i = 1, 2, ..., 5$ $(i + \frac{i}{n}) = (i + j)$ formula de tasas equivalentes, entonces calculamos las tasas efectivas anuales.

$$r_1 = \frac{10.64\%}{12} = 0.1117$$
$$r_2 = \frac{10.85\%}{12} = 0.1140$$
$$r_3 = \frac{10.95\%}{12} = 0.1151$$
$$r_4 = \frac{11.15\%}{12} = 0.1173$$

efectivos anuales

Calculamos el capital rquerida para cada inversión.

$$C_1 = 200,000(1+0.1117)^{-1} = 179,897.33$$
$$C_2 = 350,000(1+0.1140)^{-2} = 282,000.3471$$
$$C_3 = 400,000(1+0.1151)^{-3} = 288,430.5337$$
$$C_4 = 500,000(1+0.1173)^{-4} = 320,751.7289$$
$$C_5 = 1,000,000(1+0.1173)^{-5} = 574,114.8125$$

Entonces la inversión de tasas soberanases la suma de todos los montos anteriores

$$Inversin = 179,897.33 + 282,000.3471 + 288,430.5337 + 320,751.7289 + 574,114.8125 = 1,645,194.753$$

Comparado con la inversión del proyecto, en tasas soberanas lubres de riesgo tenemos que invertir $\$45,194.753$ más para obtener los mismos flujos en los mismos tiempos que el proyecto de inversión.

¿Qué decisión tomaría? Si tenemos disponibilidad de $\$45,194.753$. La verdad no me parece una suma alta a pagar a cambio de tener una inversión segura

Decido invertir en las tasas soberansa libres de riesgo, ya que por una cantidad "pequeña" me cubren del riesgo pero recibo lo mismo que si hubiese invertido en el proyecto.

7. Descarga la información del precio de cierre de las acciones de GAP, El Puerto de Liverpool, Gruma, Controladora Alsea y Genomma Lab Internacional, así como del IPC, utilizando como fecha inicial el 22/05/2020 y como fecha final el 21/05/2025. Recuerda que estas acciones deben ser las emitidas en la BMV por lo que deben estar en pesos mexicanos. Una vez descargada la información realiza lo siguiente:

a) Obtén los rendimientos continuos diarios (logarítmicos).

b) Grafica la series de tiempo de los rendimientos de cada activo y del IPC.

c) Calcula las betas de cada activo riesgoso y define si son defensivas, agresivas o neutras, además da una breve interpretación de sus valores.

d) Obtén el rendimiento esperado de cada activo bajo CAPM suponiendo una tasa libre de riesgo diaria de 0.029% diaria.

e) Calcula el alpha de cada activo riesgoso, ¿están subvaluados o sobrevalorados?

f) Grafica la SML correspondiente a este mercado y ubica gráficamente a cada uno de los cinco activos riesgosos de acuerdo con su beta y su rendimiento esperado real visto en el mercado.

[23]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf
from sklearn.linear_model import LinearRegression
import os
np.set_printoptions(suppress=True)  # Suprime la notación científica
pd.set_option('display.float_format', '{:.5f}'.format)  # 10 decimales


# Diccionario de tickers y nombres amigables
tickers = {
    'GAPB.MX': 'GAP',
    'LIVEPOLC-1.MX': 'Liverpool',
    'GRUMAB.MX': 'Gruma',
    'ALSEA.MX': 'Alsea',
    'LABB.MX': 'Genomma Lab',
    '^MXX': 'IPC'
}

# Fechas de análisis
start_date = '2020-05-22'
end_date = '2025-05-22' #para que rescate el valor del 2025-05-21

# Crear directorio para guardar los archivos si no existe
output_dir = 'stock_data'
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

# Descargar y guardar cada ticker individualmente
for yfinance_ticker, friendly_name in tickers.items():
    try:
        print(f"Descargando datos para {friendly_name}...")

        # Descargar datos
        data = yf.download(yfinance_ticker, start=start_date, end=end_date,␣
 ↪auto_adjust=False)['Close']
        data.columns = ['Close']  # Renombrar columna

        # Guardar en CSV
        filename = f"{output_dir}/{friendly_name.replace(' ', '_')}.csv"
        data.to_csv(filename)
        print(f"Datos guardados en {filename}\n")

    except Exception as e:
        print(f"Error al descargar {friendly_name}: {str(e)}\n")

print("Proceso completado.")
```

```
Descargando datos para GAP...
[**********************100%**********************]  1 of 1 completed
Datos guardados en stock_data/GAP.csv

Descargando datos para Liverpool...
[**********************100%**********************]  1 of 1 completed
Datos guardados en stock_data/Liverpool.csv
```

```
Descargando datos para Gruma...
[**********************100%**********************]  1 of 1 completed
Datos guardados en stock_data/Gruma.csv

Descargando datos para Alsea...
[**********************100%**********************]  1 of 1 completed
Datos guardados en stock_data/Alsea.csv

Descargando datos para Genomma Lab...
[**********************100%**********************]  1 of 1 completed
Datos guardados en stock_data/Genomma_Lab.csv

Descargando datos para IPC...
[**********************100%**********************]  1 of 1 completed
Datos guardados en stock_data/IPC.csv

Proceso completado.
```

a) Obtenemos los rendimientos logarítmicos

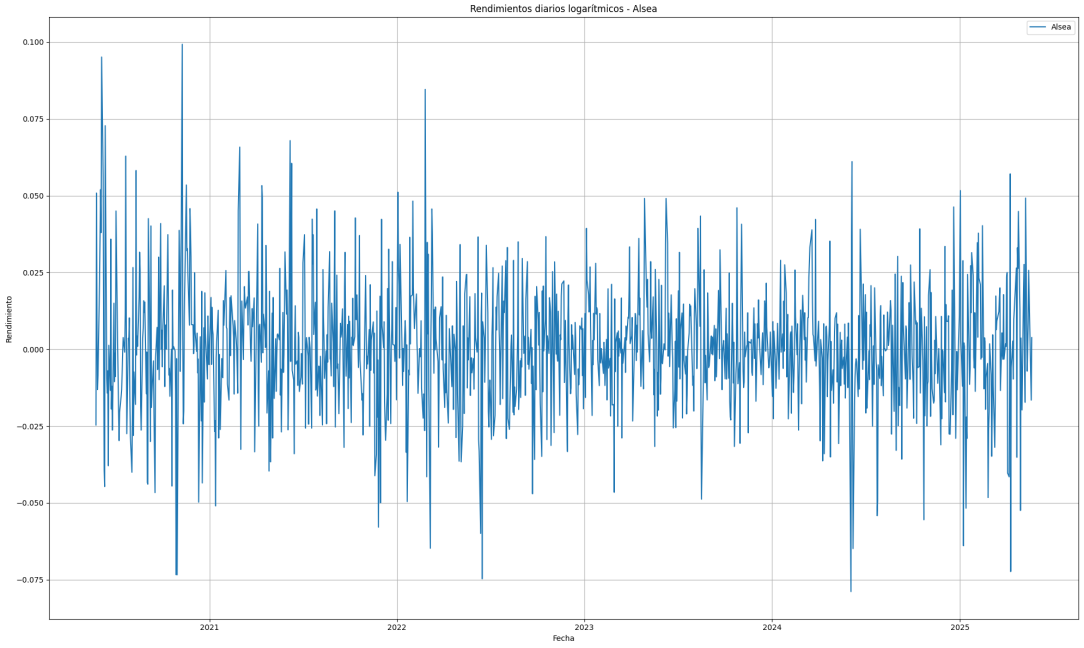
[24]:
```
path = 'stock_data'
files = {
    'Alsea': 'Alsea.csv',
    'GAP': 'GAP.csv',
    'Genomma Lab': 'Genomma_Lab.csv',
    'Gruma': 'Gruma.csv',
    'IPC': 'IPC.csv',
    'Liverpool': 'Liverpool.csv'
}

data_raw = pd.DataFrame()

for nombre, archivo in files.items():
    df = pd.read_csv(os.path.join(path, archivo), index_col='Date',␣
 ↪parse_dates=True)
    data_raw[nombre] = df['Close']

# Unificar precios en un solo DataFrame
precios_df = pd.DataFrame(data_raw).sort_index()

# Calcular rendimientos logarítmicos
rendimientos = np.log(precios_df / precios_df.shift(1)).dropna()
rendimientos
```

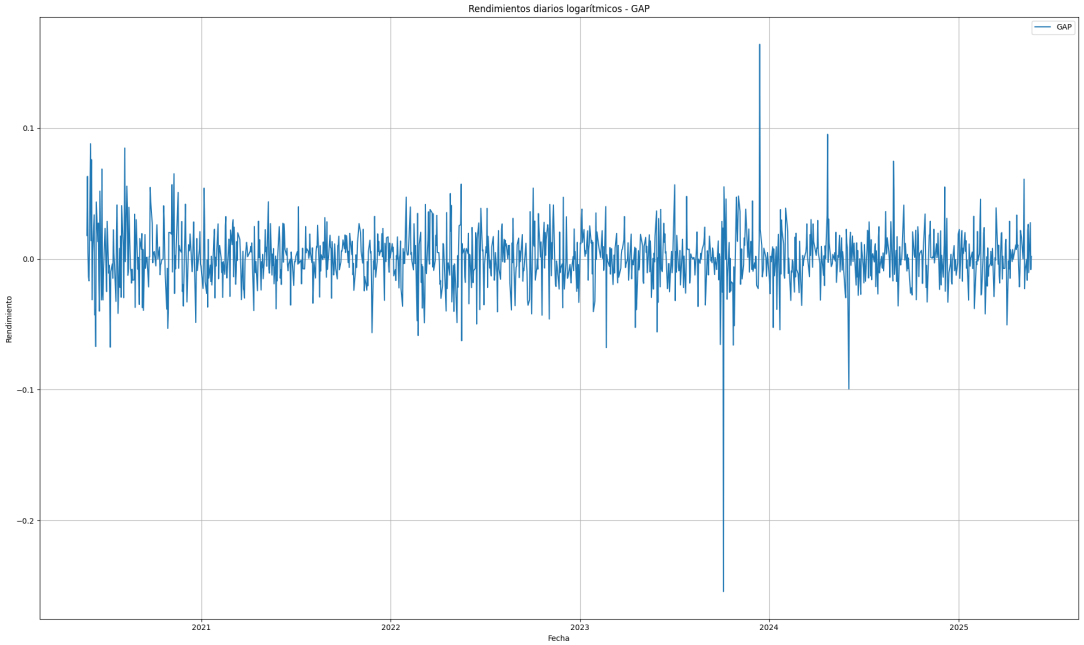
Date	Alsea	GAP	Genomma Lab	Gruma	IPC	Liverpool
2020-05-25	-0.02465	0.01788	0.03097	0.04462	0.00135	-0.00942
2020-05-26	0.05082	0.06305	0.01143	-0.04687	0.01039	0.03007
2020-05-27	0.00997	0.00458	0.00886	0.00537	0.01869	0.00037
2020-05-28	-0.01313	-0.01374	-0.01133	-0.01799	-0.01040	0.00617
2020-05-29	-0.00850	-0.01680	0.00149	-0.01636	-0.01061	0.03280
...	...	...	...	...	...	...

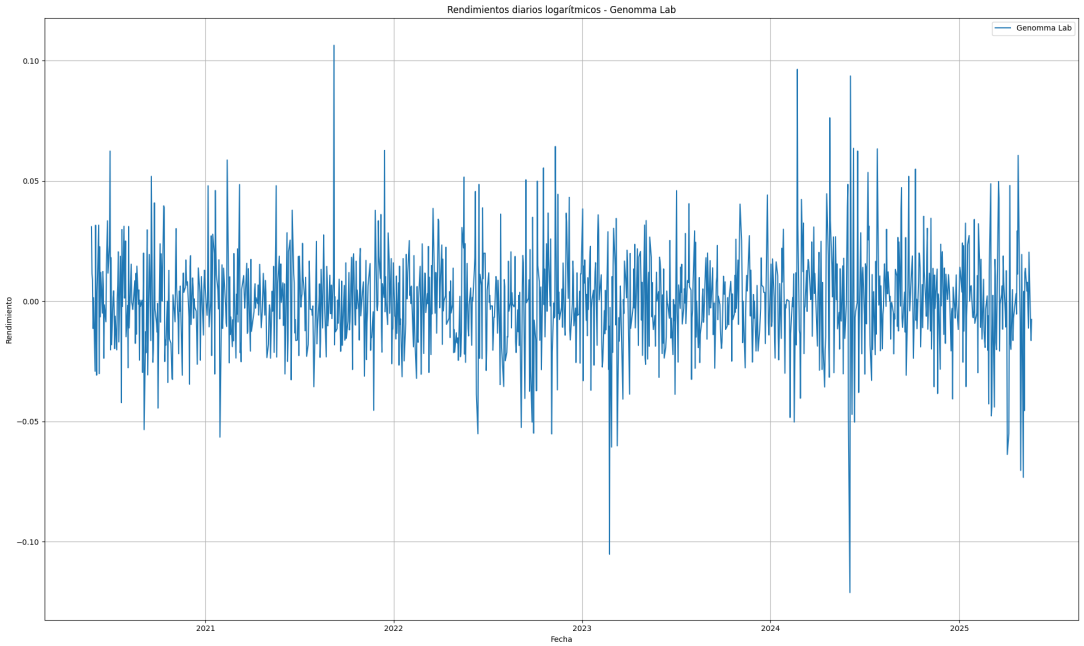
```
2025-05-15  0.02568  0.02640     -0.01118  0.00401  0.00545    0.00408
2025-05-16  0.01724 -0.01048      0.02041  0.00817  0.00047   -0.01499
2025-05-19 -0.00890  0.02771     -0.00876  0.00620  0.00869   -0.00768
2025-05-20 -0.01655 -0.00820     -0.01634  0.02079 -0.00312    0.00333
2025-05-21  0.00380  0.00191     -0.00756  0.00335  0.00440   -0.00396

[1257 rows x 6 columns]
```

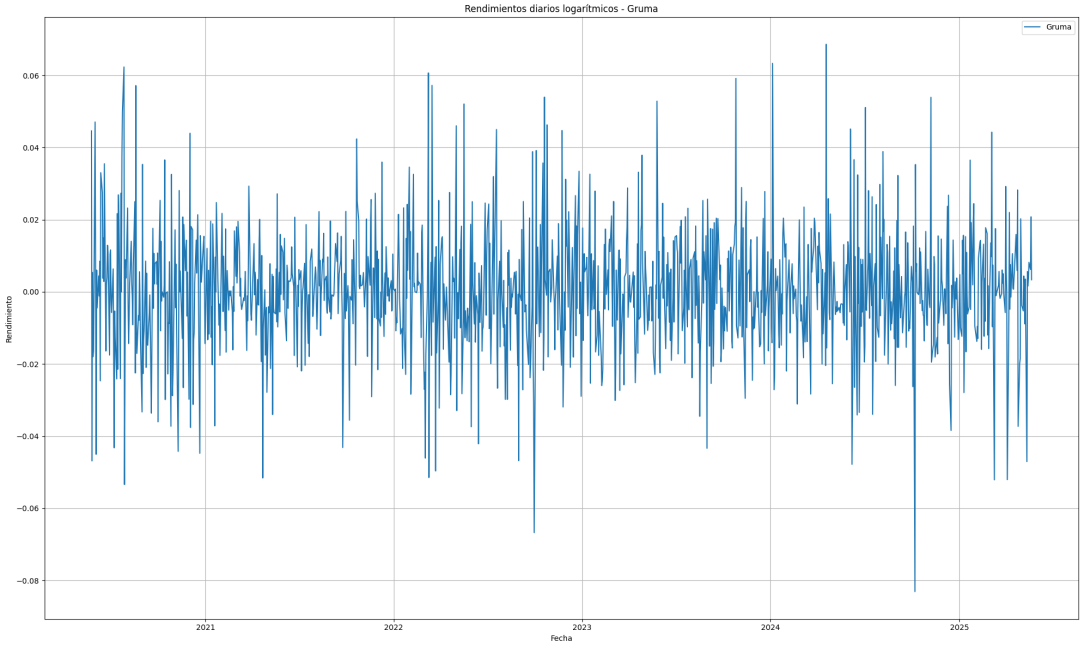
b) Graficamos la series de tiempo de los rendimientos de cada activo y del IPC.

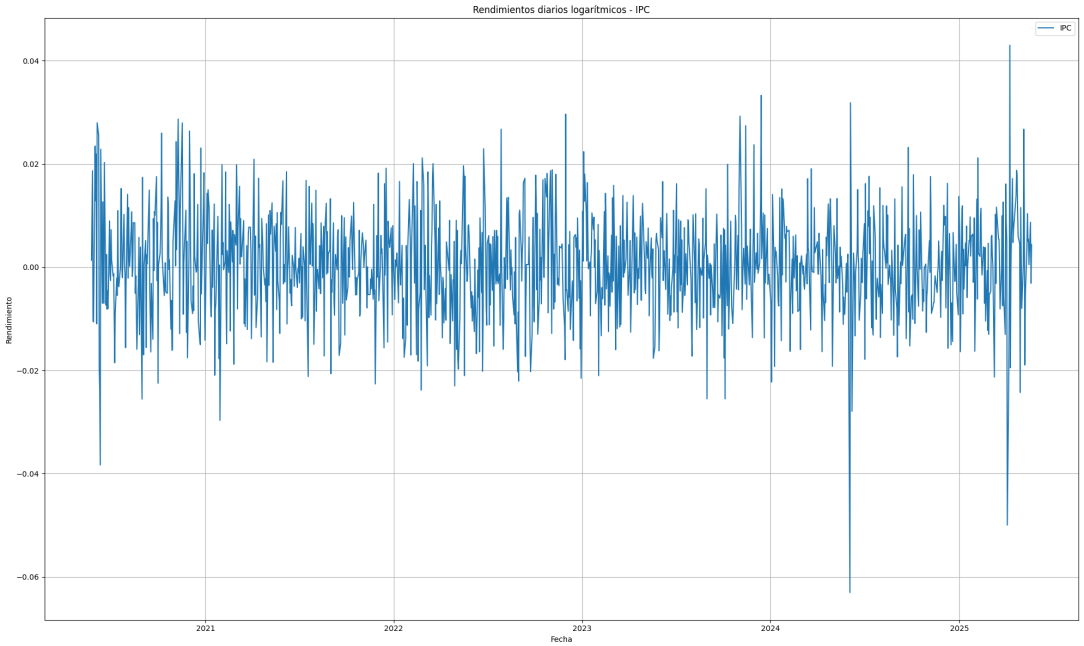
[25]:
```python
for col in rendimientos.columns:
    plt.figure(figsize=(20, 12))
    plt.plot(rendimientos.index, rendimientos[col], label=f'{col}')
    plt.title(f'Rendimientos diarios logarítmicos - {col}')
    plt.xlabel('Fecha')
    plt.ylabel('Rendimiento')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

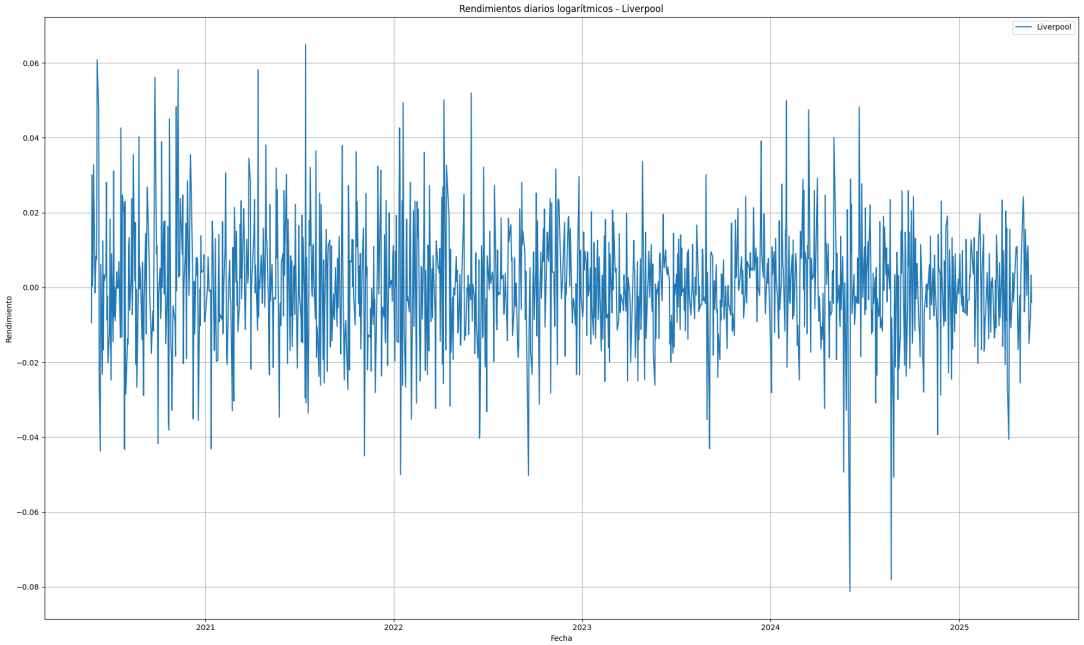



Rendimientos diarios logarítmicos - GAP
GAP
0.1
0.0
−0.1
−0.2
Rendimiento
2021
2022
2023
2024
2025
Fecha

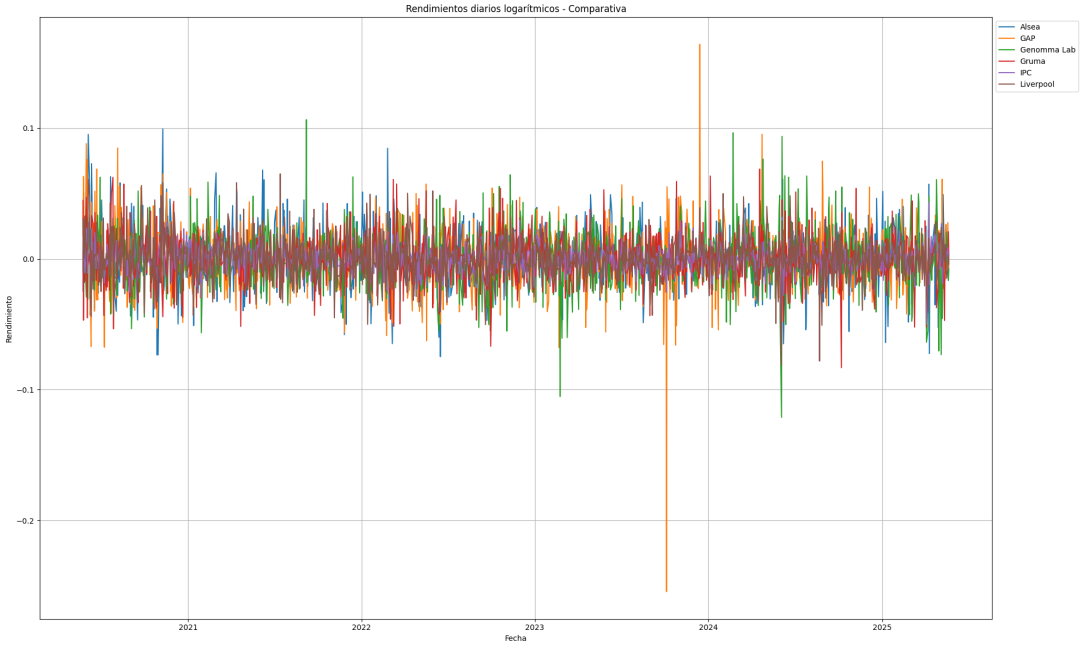

Rendimientos diarios logarítmicos - Genomma Lab
Genomma Lab
0.10
0.05
0.00
−0.05
−0.10
Rendimiento
2021
2022
2023
2024
2025
Fecha


Rendimientos diarios logarítmicos - Gruma
Gruma
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
Rendimiento
2021
2022
2023
2024
2025
Fecha


Rendimientos diarios logarítmicos - IPC
IPC
0.04
0.02
0.00
−0.02
−0.04
−0.06
Rendimiento
2021
2022
2023
2024
2025
Fecha



[26]:
```python
import matplotlib.pyplot as plt

# Crear una única figura para todas las series
plt.figure(figsize=(20, 12))

# Graficar todas las columnas en la misma figura
for col in rendimientos.columns:
    plt.plot(rendimientos.index, rendimientos[col], label=f'{col}')

# Añadir elementos comunes
plt.title('Rendimientos diarios logarítmicos - Comparativa')
plt.xlabel('Fecha')
plt.ylabel('Rendimiento')
plt.legend(loc='upper left', bbox_to_anchor=(1, 1))  # Leyenda fuera del gráfico
plt.grid(True)
plt.tight_layout()

# Mostrar la figura combinada
plt.show()
```



- La gráfica de GAP muestra una tendencia más destacadas con picos de rendimiento más altos en comparación con los demás, pero también con picos más bajos.
- Los demás activos tienen rendimientos más estables al parecer, se mantienen dentro de un rango más estrecho.
- El IPC al parecer es el menos oscilante, con rendimientos más moderados y menos extremos. Esto sugiere que el IPC como indicador del mercado muestra que el mercado en general pese a oscilar no lo hace de manera tan extrema como los activos individuales.

d) Calculamos las betas de cada activo riesgoso y define si son defensivas, agresivas o neutras, además da una breve interpretación de sus valores.

[27]:
```
betas = {}
interceptos = {}

for col in rendimientos.columns:
    if col != 'IPC':
        X = rendimientos[['IPC']].values.reshape(-1, 1)
        y = rendimientos[col].values
        model = LinearRegression().fit(X, y)
        betas[col] = model.coef_[0]

# Clasificación
clasificacion = {}
for activo, beta in betas.items():
    if beta < 1:
        clasificacion[activo] = 'Defensiva'

    elif beta > 1:
        clasificacion[activo] = 'Agresiva'
    else:
        clasificacion[activo] = 'Neutra'


beta_df = pd.DataFrame({
```

```
    'Beta': betas,
    'Clasificación': clasificacion
})
beta_df
```

```
               Beta Clasificación
Alsea       0.86533     Defensiva
GAP         1.26951      Agresiva
Genomma Lab 0.53543     Defensiva
Gruma       0.56701     Defensiva
Liverpool   0.63439     Defensiva
```

Empresa	Beta	Tipo	¿Qué significa?
Alsea	0.87	Defensiva	Bastante estable. Aunque no es la más tranquila, resiste mejor a movimientos bruscos el mercado.
GAP	1.27	Agresiva	Sube y baja más que el mercado. Buenas ganancias en buenos tiempos, pero riesgosa.
Genomma	0.54	Defensiva	Muy estable. Cambia poco aunque el mercado se mueva mucho (como medicinas).
Gruma	0.57	Defensiva	Segura y estable. Vende cosas que siempre se necesitan (como alimentos).
Liverpool	0.63	Defensiva	Más estable que el mercado, pero no tanto como Genomma o Gruma.

d) Obtenemos el rendimiento esperado de cada activo bajo CAPM suponiendo una tasa libre de riesgo diaria de 0.029% diaria.

[28]:
```
rf = 0.00029  # 0.029% diario
E_rm = rendimientos['IPC'].mean()

rend_esperados = {}
for activo, beta in betas.items():
    rend_esperados[activo] = rf + beta * (E_rm - rf)
rend_esperados_df = pd.DataFrame({
    'Rendimiento Esperado': rend_esperados
}).sort_values(by='Rendimiento Esperado', ascending=False)
rend_esperados_df
```

```
            Rendimiento Esperado
GAP                      0.00042
Alsea                    0.00038
Liverpool                0.00035
Gruma                    0.00035
Genomma Lab              0.00034
```

e) Calculamos el alpha de cada activo riesgoso, ¿están subvaluados o sobrevalorados?

[29]:
```
alphas = {}
for activo in betas.keys():
    r_i_bar = np.mean(rendimientos[activo])
    alpha = r_i_bar - (rf + betas[activo] * (E_rm - rf))
    alphas[activo] = alpha

valuacion = {}
for activo, alpha in alphas.items():
    if alpha > 0:
        valuacion[activo] = 'Subvaluada'
    elif alpha < 0:
```

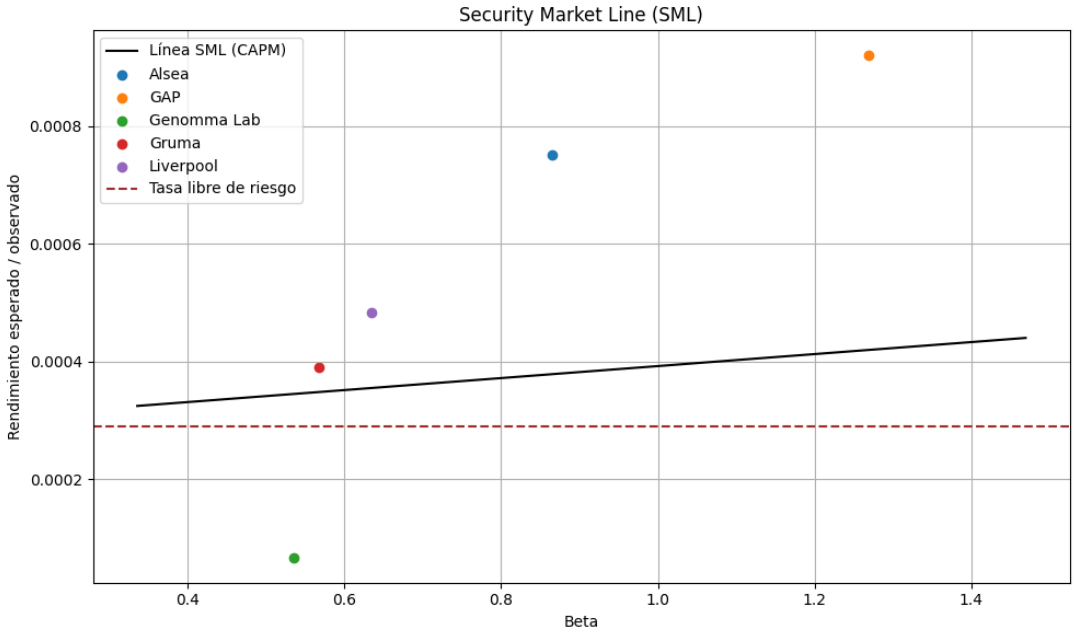
```
        valuacion[activo] = 'Sobrevaluada'
    else:
        valuacion[activo] = 'Correctamente valuada'

# Crear DataFrame resumen
resultados_df = pd.DataFrame({
    'Beta': betas,
    'Rendimiento Esperado CAPM': rend_esperados,
    'Rendimiento Promedio Observado': rendimientos.mean().drop('IPC'),
    'Alpha': alphas,
    'Valuación': valuacion
})
resultados_df
```

[7]:

	Beta	Rendimiento Esperado CAPM	Rendimiento Promedio Observado	Alpha	Valuación
Alsea	0.86533	0.00038	0.00075	0.00037	Subvaluada
GAP	1.26951	0.00042	0.00092	0.00050	Subvaluada
Genomma Lab	0.53543	0.00034	0.00007	-0.00028	Sobrevaluada
Gruma	0.56701	0.00035	0.00039	0.00004	Subvaluada
Liverpool	0.63439	0.00035	0.00048	0.00013	Subvaluada

f) Graficamos la SML correspondiente a este mercado y ubica gráficamente a cada uno de los cinco activos riesgosos de acuerdo con su beta y su rendimiento esperado real visto en el mercado.

[30]:
```
plt.figure(figsize=(10, 6))

# Línea teórica CAPM
betas_array = np.linspace(min(betas.values()) - 0.2, max(betas.values()) + 0.2, 100)
sml_line = rf + betas_array * (E_rm - rf)
plt.plot(betas_array, sml_line, label='Línea SML (CAPM)', color='black')

# Puntos observados
for activo in betas:
    plt.scatter(betas[activo], np.mean(rendimientos[activo]), label=activo)

# Etiquetas
plt.title('Security Market Line (SML)')
plt.xlabel('Beta')
plt.ylabel('Rendimiento esperado / observado')
plt.axhline(rf, color='brown', linestyle='--', label='Tasa libre de riesgo')
plt.legend()
plt.grid(True)
plt.tight_layout()
```



Lo que nos dice la gráfica es lo siguiente:

- La línea negra es la SML, que muestra la relación entre el riesgo (beta) y el rendimiento esperado.
- Los puntos representan los activos riesgosos (Alsea, GAP, Genomma Lab, Gruma y Liverpool) en relación a su beta y rendimiento esperado.
- Si un activo está por encima de la SML, significa que está ofreciendo un rendimiento mayor al esperado para su nivel de riesgo (subvaluado). Si está por debajo, significa que está ofreciendo un rendimiento menor al esperado para su nivel de riesgo (sobrevaluado).
- En este caso, GAP y Alsea están por encima de la SML, lo que indica que están subvaluados y ofrecen un rendimiento mayor al esperado para su riesgo. Genomma Lab está por debajo de la SML, lo que indica que está sobrevaluado y ofrece un rendimiento menor al esperado para su riesgo.

8. Investigue y explique detalladamente en qué consiste alguno de los dos Modelos de Fama y French, ya sea el de tres factores o el de 5 factores. Explique qué proponen y describa detalladamente cada uno de los factores que proponen.

Este modelo de tres factores, que describe mejor el modelo de CAPM, se expone para justificar, buscando dar una respuesta que además de los factores contemplados y conocidos a la hora de realizar inversiones; con esta manera se tendrán más en cuenta dos factores adicionales, los cuales son el tamaño de la empresa y la relación entre los fondos en su ámbito contable y de mercado entre el valor de sus acciones. La idea del modelo la aportan Eugene Fama y Kenneth French en 1992. El modelo refleja la idea que el rendimiento de una acción puede explicarse no sólo por la sensibilidad que tiene con respecto al mercado sino que también puede estar relacionada con el tamaño de la empresa y la relación entre el valor que tiene una acción en el mercado y el que tiene en su contabilidad.

Factores del modelo:

Factor de mercado o de exposición al riesgo (Market Risk Premium – Rm – Rf).

Este es el mismo factor que se presenta en el modelo de CAPM. Describe el exceso de rendimiento que presenta el mercado con relación a la tasa libre del mercado (Rf). Representa la sensibilidad que tiene una acción con respecto a los movimientos que presenta el mercado.

Fórmula: $Rm - Rf$, donde Rm es el rendimiento esperado del mercado y Rf es la tasa libre de riesgo.

Interpretación: Las acciones con betas altas son más sensibles a las variaciones que presentarán el mercado, por consiguiente, las acciones serán más arriesgadas.

Factor de Tamaño (SMB - Small Minus Big)

Este factor captura la diferencia en rendimientos de empresas pequeñas en relación a las más grandes. Fama y French encontraron que muchas empresas pequeñas tienden a tener un rendimiento mayor que las empresas grandes conocidas; esto a pesar de comenzar a controlar el riesgo en la inversión en acciones a partir del riesgo de mercado.

Refleja al factor Book-to-Market equity y proviene de la diferencia entre la rentabilidad del activo con mayor ratio (acciones conocidas como value stocks y atribuidas a empresas que están ante una posibilidad de quiebra pero que pueden resultar una buena inversión a pesar de conllevar un mayor riesgo) y la de menor ratio en una cartera (acciones de crecimiento o growth stocks que se esperan que crezcan más rápido que el mercado).

Cálculo: se logra restando el rendimiento esperado de las acciones de empresas grandes del rendimiento esperado de las acciones de empresas pequeñas.

Interpretación: Teniendo en cuenta el hecho de que las empresas pequeñas son más arriesgadas (ya que suelen presentar menos liquidez y suelen ser más sensibles al impacto de shocks económicos), los inversores exigen mayores rendimientos que para acciones de empresas grandes.

Factor de Valor (HML - High Minus Low)

Este factor es capaz de medir la diferencia en rendimientos de acciones de empresas que tienen un alto ratio de valor contable a mercado (value stocks) en relación aquellas empresas que tienen un ratio bajo (growth stocks). Fama y French encontraron que las acciones value tienden a producir un rendimiento superior que las growth stocks a lo largo del tiempo.

Representa al factor tamaño y se calcula restando a la rentabilidad del activo de menor tamaño o de menor capitalización, la rentabilidad del activo de mayor capitalización de la cartera.

Cálculo: Se produce restando del rendimiento promedio de las acciones de empresas con ratios bajos de valor contable a mercado el rendimiento promedio de las acciones de empresas con ratios altos de valor contable a mercado.

Interpretación: Las acciones value suelen estar infravaloradas y representan empresas más estables pero con menor crecimiento, mientras que las growth stocks están sobrevaloradas pero tienen mayor potencial de crecimiento.

La ecuación del modelo es la siguiente

$$E[R_i] = Rf + B_i(Rm - Rf) + S_i(SMB) + H_i(HLM)$$

Donde:

- B_i, S_i, H_i: sensibilidad de la acción a los factores de mercado, tamaño y valor.
- S_i, H_i: se determinan bajo regresión lineal.

FUENTES DE CONSULTA

- Fama-French Three-Factor Model
- Modelo de 3 factores de Fama y French | Gabriela de la Torre
- Current Research Returns
- The Cross-Section of Expected Stock Returns

9. Considere las acciones de alguna de las emisoras listadas abajo y obtenga el histórico de precios de 22 de mayo 2022 a 21 de mayo de 2025.

a) Obtenga la tasa cete a 28 días y la tasa cete a 182 días para el mismo periodo y, usando la definición del Cuadro 2 del artículo compartido en clase, calcule los diferenciales de tasa mensuales necesarios para medir el Riesgo de Horizonte de Tiempo.

b) Con el histórico de precios de la emisora elegida, use los precios de las fechas que coinciden con las tasas cete y calcule ahora los rendimientos efectivos mensuales.

c) Obtenga el histórico del valor de la UDI para las mismas fechas. Y obtenga entonces la tasa de crecimiento de la UDI mensual de manera análoga a los incisos anteriores.

d) Análogamente a los datos anteriores obtenga el histórico del precio del dólar para obtener rendimientos mensuales.

e) De igual manera obtenga el histórico del precio del dólar para obtener rendimientos mensuales del mercado accionario mexicano.

f) Plantea el modelo de cuatro factores, considerando prima de riesgo del mercado, rendimiento del dólar, crecimiento de la inflación y el riesgo de horizonte temporal (definido en cuadro 2 del artículo compartido en clase), obtenga el intercepto y las betas, e interpreta cada uno. ¿Qué tan bueno es su ajuste?

g) Plantea el modelo de tres factores eliminando del modelo anterior el factor que menos influencia tiene sobre el rendimiento del activo mercado y obtenga el intercepto y las betas, e interpreta cada uno. ¿Qué tan bueno es su ajuste?

h) Agregue a su consideración un factor congruente al modelo en g). Muestre el resumen de su ajuste y diga si mejoró.

i) Considere el modelo del inciso f) y el último precio de la acción y estime cuál sería el nuevo precio si toma el valor del dólar al 21 de junio de 2025 (que es el siguiente día hábil respecto a su último dato de dólar).

j) Considere el modelo del inciso g) y el último precio de la acción y estime cuál sería el nuevo precio si el mercado pierde 2pp de prima de riesgo.

k) Considere el modelo del inciso f) y el último precio de la acción y estime cuál sería el nuevo precio si la inflación crece 80 pb.

CLAVE	RAZÓN SOCIAL
CIDMEGA	GRUPE, S.A.B. DE C.V.
HCITY	HOTELES CITY EXPRESS, S.A.B. DE C.V.
HOTEL	GRUPO HOTELERO SANTA FE, S.A.B. DE C.V.
POSADAS	GRUPO POSADAS, S.A.B. DE C.V.
RLH	RLH PROPERTIES, S.A.B. DE C.V.

[31]:
```
#importamos las librerias necesarias
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import math
import statsmodels.api as sm
import yfinance as yf
np.set_printoptions(suppress=True)
pd.set_option('display.float_format', '{:.4f}'.format)
```

[32]:
```
#generamos los datos
ric = ['HOTEL.MX']
fi = '2022-05-22'
```

```
ff = '2025-05-21'
```

[33]:
```
dfh = yf.download(ric,start = fi, end=ff)['Close'].reset_index()
dfh
```

```
YF.download() has changed argument auto_adjust default to True

[**********************100%**********************]  1 of 1 completed

Ticker       Date  HOTEL.MX
0      2022-05-23    3.9901
1      2022-05-24    3.9999
2      2022-05-25    3.9604
3      2022-05-26    3.9604
4      2022-05-27    3.9901
..            ...       ...
747    2025-05-14    3.8500
748    2025-05-15    3.8500
749    2025-05-16    3.8500
750    2025-05-19    3.8500
751    2025-05-20    3.8500

[752 rows x 2 columns]
```

[34]:
```
#importamos el archivo de tasas CETE
data_raw_cete = pd.read_csv(r"Tasas Cetes 28 y 182 dias_r.csv")

data_raw_cete
```

```
          Fecha  CETE 28 dias   CETE 182 dias
0    06/01/2022         5.5100         6.4000
1    13/01/2022         5.5200         6.4000
2    20/01/2022         5.5700         6.4400
3    27/01/2022         5.5000         6.4400
4    03/02/2022         5.8400         6.4900
..          ...            ...            ...
172  24/04/2025         8.8000         8.4300
173  30/04/2025         8.6500         8.3900
174  08/05/2025         8.5500         8.3100
175  15/05/2025         8.4000         8.2500
176  22/05/2025         8.1500         8.1600

[177 rows x 3 columns]
```

a) Obtenga la tasa cete a 28 días y la tasa cete a 182 días para el mismo periodo y, usando la definición del Cuadro 2 del artículo compartido en clase, calcule los diferenciales de tasa mensuales necesarios para medir el Riesgo de Horizonte de Tiempo.

[35]:
```
df_cete = data_raw_cete.copy()
df_cete['Dif. TSA'] = data_raw_cete['CETE 182 dias']  - data_raw_cete['CETE 28 dias␣
↪'] #calculamos la diferencia entre las tasas CETE de 182 y 28 dias

df_cete
```

```
          Fecha  CETE 28 dias   CETE 182 dias  Dif. TSA
0    06/01/2022         5.5100         6.4000    0.8900
1    13/01/2022         5.5200         6.4000    0.8800
2    20/01/2022         5.5700         6.4400    0.8700
3    27/01/2022         5.5000         6.4400    0.9400
4    03/02/2022         5.8400         6.4900    0.6500
```

```
..          ...            ...            ...       ...
172  24/04/2025         8.8000         8.4300   -0.3700
173  30/04/2025         8.6500         8.3900   -0.2600
174  08/05/2025         8.5500         8.3100   -0.2400
175  15/05/2025         8.4000         8.2500   -0.1500
176  22/05/2025         8.1500         8.1600    0.0100

[177 rows x 4 columns]
```

b) Con el histórico de precios de la emisora elegida, use los precios de las fechas que coinciden con las tasas cete y calcule ahora los rendimientos efectivos mensuales.

[36]:
```
df_cete['Fecha'] = pd.to_datetime(df_cete['Fecha'], dayfirst=True)

dfh['Fecha'] = pd.to_datetime(dfh['Date'])
df_comun = pd.merge(dfh, df_cete, on='Fecha', how='inner')

df_comun = df_comun.sort_values('Fecha')

df_comun
```

```
          Date  HOTEL.MX      Fecha  CETE 28 dias   CETE 182 dias  Dif. TSA
0   2022-05-26    3.9604 2022-05-26         6.9000         8.1400    1.2400
1   2022-06-02    3.9901 2022-06-02         7.0100         8.3500    1.3400
2   2022-06-09    3.9506 2022-06-09         7.3200         8.4400    1.1200
3   2022-06-16    3.9506 2022-06-16         7.1500         8.7300    1.5800
4   2022-06-23    3.8617 2022-06-23         7.5000         8.8600    1.3600
..         ...       ...        ...            ...            ...       ...
151 2025-04-16    3.8500 2025-04-16         9.0000         8.5300   -0.4700
152 2025-04-24    3.8500 2025-04-24         8.8000         8.4300   -0.3700
153 2025-04-30    3.8500 2025-04-30         8.6500         8.3900   -0.2600
154 2025-05-08    3.8000 2025-05-08         8.5500         8.3100   -0.2400
155 2025-05-15    3.8500 2025-05-15         8.4000         8.2500   -0.1500

[156 rows x 6 columns]
```

[37]:
```
df_tsa = df_comun[['Fecha', 'Dif. TSA']].copy()

df_tsa
```

```
         Fecha  Dif. TSA
0   2022-05-26    1.2400
1   2022-06-02    1.3400
2   2022-06-09    1.1200
3   2022-06-16    1.5800
4   2022-06-23    1.3600
..         ...       ...
151 2025-04-16   -0.4700
152 2025-04-24   -0.3700
153 2025-04-30   -0.2600
154 2025-05-08   -0.2400
155 2025-05-15   -0.1500

[156 rows x 2 columns]
```

[38]:
```
df = df_comun[['Fecha', 'HOTEL.MX']].copy()

df
```

```
         Fecha  HOTEL.MX
```

```
0   2022-05-26    3.9604
1   2022-06-02    3.9901
2   2022-06-09    3.9506
3   2022-06-16    3.9506
4   2022-06-23    3.8617
..         ...       ...
151 2025-04-16    3.8500
152 2025-04-24    3.8500
153 2025-04-30    3.8500
154 2025-05-08    3.8000
155 2025-05-15    3.8500

[156 rows x 2 columns]
```

[513]:
```
#calculamos la tasa de rendimiento del hotel a 28 dias
df_h = df.copy()
df_h['Hotel_rend'] = np.log(df['HOTEL.MX']/df['HOTEL.MX'].shift(4))
df_h.dropna(inplace=True)
df_h.reset_index(drop= True, inplace=True)
df_h
```

[513]:
```
          Fecha  HOTEL.MX  Hotel_rend
0   2022-06-23    3.8617     -0.0253
1   2022-06-30    3.7530     -0.0612
2   2022-07-07    3.7530     -0.0513
3   2022-07-14    3.6740     -0.0726
4   2022-07-21    3.7333     -0.0338
..         ...       ...         ...
147 2025-04-16    3.8500      0.0026
148 2025-04-24    3.8500      0.0000
149 2025-04-30    3.8500     -0.0129
150 2025-05-08    3.8000     -0.0131
151 2025-05-15    3.8500      0.0000

[152 rows x 3 columns]
```

c) Obtenga el histórico del valor de la UDI para las mismas fechas. Y obtenga entonces la tasa de crecimiento de la UDI mensual de manera análoga a los incisos anteriores.

[514]:
```
data_udi = pd.read_csv(r"C:\Users\rubrh\OneDrive\Escritorio\Metodos cuantitativos␣
↪en finanzas\Ultima_tarea_examen\UDI_r.csv")

data_udi
```

[514]:
```
           Fecha    UDI
0     01/01/2020 6.4004
1     02/01/2020 6.4018
2     03/01/2020 6.4032
3     04/01/2020 6.4046
4     05/01/2020 6.4060
...          ...    ...
1983  06/06/2025 8.4798
1984  07/06/2025 8.4803
1985  08/06/2025 8.4807
1986  09/06/2025 8.4812
1987  10/06/2025 8.4816

[1988 rows x 2 columns]
```

[515]:
```
data_udi['Fecha'] = pd.to_datetime(data_udi['Fecha'], dayfirst=True, format='%d/%m/
↪%Y') #confirmamos que la fecha esta en el formato correcto
```

[516]:
```
df['Fecha'] = pd.to_datetime(df['Fecha'], dayfirst=True)

data_udi['Fecha'] = pd.to_datetime(data_udi['Fecha'])
df_comun = pd.merge(data_udi, df, on='Fecha', how='inner')

df_comun = df_comun.sort_values('Fecha')

df_comun
```

[516]:
```
         Fecha    UDI  HOTEL.MX
0   2022-05-26 7.3316    3.9604
1   2022-06-02 7.3295    3.9901
2   2022-06-09 7.3274    3.9506
3   2022-06-16 7.3361    3.9506
4   2022-06-23 7.3465    3.8617
..         ...    ...       ...
151 2025-04-16 8.4364    3.8500
152 2025-04-24 8.4460    3.8500
153 2025-04-30 8.4507    3.8500
154 2025-05-08 8.4563    3.8000
155 2025-05-15 8.4633    3.8500

[156 rows x 3 columns]
```

[517]:
```
dfu = df_comun[['Fecha', 'UDI']].copy()

dfu
```

[517]:
```
         Fecha    UDI
0   2022-05-26 7.3316
1   2022-06-02 7.3295
2   2022-06-09 7.3274
3   2022-06-16 7.3361
4   2022-06-23 7.3465
..         ...    ...
151 2025-04-16 8.4364
152 2025-04-24 8.4460
153 2025-04-30 8.4507
154 2025-05-08 8.4563
155 2025-05-15 8.4633

[156 rows x 2 columns]
```

Calculemos la tasa de crecimiento

[518]:
```
df_udi = df.copy()
df_udi['UDI_rend'] = np.log(dfu['UDI']/dfu['UDI'].shift(4))
df_udi.dropna(inplace=True)
df_udi.reset_index(drop= True, inplace=True)
df_udi
```

[518]:
```
         Fecha  HOTEL.MX  UDI_rend
0   2022-06-23    3.8617    0.0020
1   2022-06-30    3.7530    0.0044
2   2022-07-07    3.7530    0.0069
```

```
3   2022-07-14    3.6740    0.0078
4   2022-07-21    3.7333    0.0082
..         ...       ...       ...
147 2025-04-16    3.8500    0.0027
148 2025-04-24    3.8500    0.0032
149 2025-04-30    3.8500    0.0032
150 2025-05-08    3.8000    0.0032
151 2025-05-15    3.8500    0.0032

[152 rows x 3 columns]
```

d) Análogamente a los datos anteriores obtenga el histórico del precio del dólar para obtener rendimientos mensuales.

Obtengamos el histórico de nuestro dólar.

[519]:
```
ric = ['USDMXN=X']
fi = '2022-05-22'
ff = '2025-05-21'
```

[520]:
```
df_dolar = yf.download(ric,start = fi, end=ff)['Close'].reset_index()
df_dolar
```

```
[**********************100%**********************]  1 of 1 completed
```

[520]:
```
Ticker       Date  USDMXN=X
0      2022-05-23   19.8494
1      2022-05-24   19.9180
2      2022-05-25   19.8427
3      2022-05-26   19.8182
4      2022-05-27   19.7519
..            ...       ...
774    2025-05-14   19.3931
775    2025-05-15   19.3831
776    2025-05-16   19.4714
777    2025-05-19   19.4512
778    2025-05-20   19.2849

[779 rows x 2 columns]
```

[521]:
```
data_raw_cete['Fecha'] = pd.to_datetime(data_raw_cete['Fecha'], dayfirst=True)

df_dolar['Fecha'] = pd.to_datetime(df_dolar['Date'])
# Realiza una fusión (merge) interno de los DataFrames df_dolar y data_raw_cete.
# La fusión se basa en la columna 'Fecha'.
# how='inner' significa que solo se incluirán las filas que tengan valores␣
 ↪coincidentes en la columna 'Fecha' en ambos DataFrames.

df_comun = pd.merge(df_dolar, data_raw_cete, on='Fecha', how='inner')

df_comun = df_comun.sort_values('Fecha')

df_comun
```

[521]:
```
          Date  USDMXN=X      Fecha  CETE 28 dias  CETE 182 dias
0   2022-05-26   19.8182 2022-05-26        6.9000         8.1400
1   2022-06-02   19.7131 2022-06-02        7.0100         8.3500
2   2022-06-09   19.5871 2022-06-09        7.3200         8.4400
3   2022-06-16   20.2222 2022-06-16        7.1500         8.7300
```

```
4   2022-06-23   20.0495 2022-06-23         7.5000         8.8600
..         ...       ...        ...            ...            ...
151 2025-04-16   20.1254 2025-04-16         9.0000         8.5300
152 2025-04-24   19.6244 2025-04-24         8.8000         8.4300
153 2025-04-30   19.5402 2025-04-30         8.6500         8.3900
154 2025-05-08   19.6031 2025-05-08         8.5500         8.3100
155 2025-05-15   19.3831 2025-05-15         8.4000         8.2500

[156 rows x 5 columns]
```

[522]:
```
df = df_comun[['Fecha', 'USDMXN=X']].copy()

df
```

[522]:
```
          Fecha  USDMXN=X
0   2022-05-26   19.8182
1   2022-06-02   19.7131
2   2022-06-09   19.5871
3   2022-06-16   20.2222
4   2022-06-23   20.0495
..         ...       ...
151 2025-04-16   20.1254
152 2025-04-24   19.6244
153 2025-04-30   19.5402
154 2025-05-08   19.6031
155 2025-05-15   19.3831

[156 rows x 2 columns]
```

calculemos nuestro rendimiento mensual

[523]:
```
df_dol = df.copy()
df_dol['USDMXN=X_rend'] = np.log(df['USDMXN=X']/df['USDMXN=X'].shift(4))
df_dol.dropna(inplace=True)
df_dol.reset_index(drop= True, inplace=True)
df_dol
```

[523]:
```
          Fecha  USDMXN=X  USDMXN=X_rend
0   2022-06-23   20.0495         0.0116
1   2022-06-30   20.1304         0.0209
2   2022-07-07   20.6411         0.0524
3   2022-07-14   20.7908         0.0277
4   2022-07-21   20.5421         0.0243
..         ...       ...            ...
147 2025-04-16   20.1254         0.0051
148 2025-04-24   19.6244        -0.0317
149 2025-04-30   19.5402        -0.0361
150 2025-05-08   19.6031        -0.0366
151 2025-05-15   19.3831        -0.0376

[152 rows x 3 columns]
```

e) De igual manera obtenga el IPC para obtener mensuales del mercado accionario mexicano.

[524]:
```
ric = ['^MXX']
fi = '2022-05-22'
ff = '2025-05-21'
```

[525]:
```
df_ipc = yf.download(ric,start = fi, end=ff)['Close'].reset_index()
df_ipc
```

```
[**********************100%**********************]  1 of 1 completed
```

[525]:
```
Ticker       Date       ^MXX
0      2022-05-23 51376.8906
1      2022-05-24 51304.0391
2      2022-05-25 51717.0703
3      2022-05-26 52143.0000
4      2022-05-27 52463.5508
..            ...        ...
747    2025-05-14 57644.9414
748    2025-05-15 57959.7188
749    2025-05-16 57987.1406
750    2025-05-19 58493.3906
751    2025-05-20 58311.1484

[752 rows x 2 columns]
```

[526]:
```
data_raw_cete['Fecha'] = pd.to_datetime(data_raw_cete['Fecha'], dayfirst=True)

df_ipc['Fecha'] = pd.to_datetime(df_ipc['Date'])
# Realiza una fusión (merge) interno de los DataFrames df_ipc y data_raw_cete.
# La fusión se basa en la columna 'Fecha'.
# how='inner' significa que solo se incluirán las filas que tengan valores␣
 ↪coincidentes en la columna 'Fecha' en ambos DataFrames.
df_comun = pd.merge(df_ipc, data_raw_cete, on='Fecha', how='inner')

df_comun = df_comun.sort_values('Fecha')

df_comun
```

[526]:
```
          Date       ^MXX      Fecha  CETE 28 dias   CETE 182 dias
0   2022-05-26 52143.0000 2022-05-26         6.9000         8.1400
1   2022-06-02 50971.3789 2022-06-02         7.0100         8.3500
2   2022-06-09 49290.2188 2022-06-09         7.3200         8.4400
3   2022-06-16 47558.5117 2022-06-16         7.1500         8.7300
4   2022-06-23 46657.8789 2022-06-23         7.5000         8.8600
..         ...        ...        ...            ...            ...
151 2025-04-16 53018.5703 2025-04-16         9.0000         8.5300
152 2025-04-24 56382.0000 2025-04-24         8.8000         8.4300
153 2025-04-30 56259.2812 2025-04-30         8.6500         8.3900
154 2025-05-08 56866.7617 2025-05-08         8.5500         8.3100
155 2025-05-15 57959.7188 2025-05-15         8.4000         8.2500

[156 rows x 5 columns]
```

[527]:
```
df = df_comun[['Fecha', '^MXX']].copy()

df
```

[527]:
```
         Fecha       ^MXX
0   2022-05-26 52143.0000
1   2022-06-02 50971.3789
2   2022-06-09 49290.2188
3   2022-06-16 47558.5117
4   2022-06-23 46657.8789
```

```
..         ...        ...
151 2025-04-16 53018.5703
152 2025-04-24 56382.0000
153 2025-04-30 56259.2812
154 2025-05-08 56866.7617
155 2025-05-15 57959.7188

[156 rows x 2 columns]
```

[528]:
```
df.columns = ['Fecha', 'IPC']
df.head()
```

[528]:
```
       Fecha        IPC
0 2022-05-26 52143.0000
1 2022-06-02 50971.3789
2 2022-06-09 49290.2188
3 2022-06-16 47558.5117
4 2022-06-23 46657.8789
```

[529]:
```
df_ipc = df.copy()
df_ipc['IPC_rend'] = np.log(df['IPC']/df['IPC'].shift(4))
df_ipc.dropna(inplace=True)
df_ipc.reset_index(drop= True, inplace=True)
df_ipc
```

[529]:
```
          Fecha        IPC  IPC_rend
0   2022-06-23 46657.8789   -0.1111
1   2022-06-30 47524.4492   -0.0700
2   2022-07-07 47374.3789   -0.0396
3   2022-07-14 46741.3086   -0.0173
4   2022-07-21 47416.3711    0.0161
..         ...        ...       ...
147 2025-04-16 53018.5703   -0.0016
148 2025-04-24 56382.0000    0.0529
149 2025-04-30 56259.2812    0.0393
150 2025-05-08 56866.7617    0.0988
151 2025-05-15 57959.7188    0.0891

[152 rows x 3 columns]
```

f) Plantea el modelo de cuatro factores, considerando prima de riesgo del mercado, rendimiento del dólar, crecimiento de la inflación y el riesgo de horizonte temporal (definido en cuadro 2 del artículo compartido en clase), obtenga el intercepto y las betas, e interpreta cada uno. ¿Qué tan bueno es su ajuste?

[530]:
```
data = pd.DataFrame()
data['Fecha'] = df_ipc['Fecha']
data['HOTELS_rend'] = df_h['Hotel_rend']
data['IPC_rend'] = df_ipc['IPC_rend']
data['USDMXN=X_rend'] = df_dol['USDMXN=X_rend']
data['UDI_rend'] = df_udi['UDI_rend']
data['Dif. TSA'] = df_tsa['Dif. TSA']

data
```

[530]:
```
          Fecha  HOTELS_rend  IPC_rend  USDMXN=X_rend  UDI_rend  Dif. TSA
0   2022-06-23      -0.0253   -0.1111         0.0116    0.0020    1.2400
1   2022-06-30      -0.0612   -0.0700         0.0209    0.0044    1.3400
2   2022-07-07      -0.0513   -0.0396         0.0524    0.0069    1.1200
```

```
3   2022-07-14      -0.0726   -0.0173         0.0277    0.0078    1.5800
4   2022-07-21      -0.0338    0.0161         0.0243    0.0082    1.3600
..         ...          ...       ...            ...       ...       ...
147 2025-04-16       0.0026   -0.0016         0.0051    0.0027   -0.1100
148 2025-04-24       0.0000    0.0529        -0.0317    0.0032   -0.1800
149 2025-04-30      -0.0129    0.0393        -0.0361    0.0032   -0.1100
150 2025-05-08      -0.0131    0.0988        -0.0366    0.0032   -0.2500
151 2025-05-15       0.0000    0.0891        -0.0376    0.0032   -0.4700

[152 rows x 6 columns]
```

[531]:
```
np.cov(data['HOTELS_rend'],data['IPC_rend']) #covarianza entre el rendimiento del␣
↪hotel y el IPC
```

[531]:
```
array([[ 0.00373963, -0.00045714],
       [-0.00045714,  0.00170108]])
```

Con 4 factores

[532]:
```
X = data[['USDMXN=X_rend','Dif. TSA', 'UDI_rend','IPC_rend']]
Y = data['HOTELS_rend']
```

[533]:
```
# Agrega una columna de constantes (intercepto) a la matriz de características X.
# Esto se usa en modelos de regresión lineal para estimar el término constante
X = sm.add_constant(X)
X
```

[533]:
```
     const  USDMXN=X_rend  Dif. TSA  UDI_rend  IPC_rend
0   1.0000         0.0116    1.2400    0.0020   -0.1111
1   1.0000         0.0209    1.3400    0.0044   -0.0700
2   1.0000         0.0524    1.1200    0.0069   -0.0396
3   1.0000         0.0277    1.5800    0.0078   -0.0173
4   1.0000         0.0243    1.3600    0.0082    0.0161
..     ...            ...       ...       ...       ...
147 1.0000         0.0051   -0.1100    0.0027   -0.0016
148 1.0000        -0.0317   -0.1800    0.0032    0.0529
149 1.0000        -0.0361   -0.1100    0.0032    0.0393
150 1.0000        -0.0366   -0.2500    0.0032    0.0988
151 1.0000        -0.0376   -0.4700    0.0032    0.0891

[152 rows x 5 columns]
```

[534]:
```
modelo = sm.OLS(Y,X).fit()
modelo
```

[534]:
```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x1bc9ddb2710>
```

[535]:
```
print(modelo.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            HOTELS_rend   R-squared:                       0.064
Model:                            OLS   Adj. R-squared:                  0.039
Method:                 Least Squares   F-statistic:                     2.523
Date:                Tue, 03 Jun 2025   Prob (F-statistic):             0.0434
Time:                        22:58:55   Log-Likelihood:                 214.62
No. Observations:                 152   AIC:                            -419.2
Df Residuals:                     147   BIC:                            -404.1
Df Model:                           4
```

```
Covariance Type:            nonrobust
==============================================================================
=
                     coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-
const              0.0013      0.009      0.154      0.878      -0.016
0.018
USDMXN=X_rend     -0.0313      0.201     -0.156      0.876      -0.429
0.366
Dif. TSA           0.0219      0.011      2.001      0.047       0.000
0.043
UDI_rend          -3.0041      1.943     -1.546      0.124      -6.844
0.836
IPC_rend          -0.2685      0.145     -1.847      0.067      -0.556
0.019
==============================================================================
Omnibus:                        3.477   Durbin-Watson:                   0.682
Prob(Omnibus):                  0.176   Jarque-Bera (JB):                4.033
Skew:                          -0.012   Prob(JB):                        0.133
Kurtosis:                       3.798   Cond. No.                         448.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[536]:
```
0.0013 -0.0313*np.mean(data['USDMXN=X_rend'])  +0.0219  *np.mean(data['Dif. TSA'])␣
↪-3.0041*np.mean(data['UDI_rend']) - 0.2685*np.mean(data['IPC_rend']) #calculo del␣
↪rendimiento esperado del hotel con los coeficientes obtenidos del modelo de␣
↪regresión lineal
```

[536]:
```
-0.0008394499115690519
```

[537]:
```
np.mean(data['HOTELS_rend']) #calculo del rendimiento esperado del hotel con el␣
↪estimador de la media
```

[537]:
```
-0.0008464708993942549
```

g) Plantea el modelo de tres factores eliminando del modelo anterior el factor que menos influencia tiene sobre el rendimiento del activo mercado y obtenga el intercepto y las betas, e interpreta cada uno. ¿Qué tan bueno es su ajuste?

Con 3 factores

[538]:
```
X = data[['USDMXN=X_rend','Dif. TSA', 'IPC_rend']]
Y = data['HOTELS_rend']
```

[539]:
```
X = sm.add_constant(X)
X
```

[539]:

	const	USDMXN=X_rend	Dif. TSA	IPC_rend
0	1.0000	0.0116	1.2400	-0.1111
1	1.0000	0.0209	1.3400	-0.0700
2	1.0000	0.0524	1.1200	-0.0396
3	1.0000	0.0277	1.5800	-0.0173
4	1.0000	0.0243	1.3600	0.0161
..	...	...	...	...

```
147 1.0000         0.0051   -0.1100   -0.0016
148 1.0000        -0.0317   -0.1800    0.0529
149 1.0000        -0.0361   -0.1100    0.0393
150 1.0000        -0.0366   -0.2500    0.0988
151 1.0000        -0.0376   -0.4700    0.0891

[152 rows x 4 columns]
```

[540]:
```
modelo = sm.OLS(Y,X).fit()
modelo
```

[540]:
```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x1bc9dc150d0>
```

[541]:
```
print(modelo.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            HOTELS_rend   R-squared:                       0.049
Model:                            OLS   Adj. R-squared:                  0.030
Method:                 Least Squares   F-statistic:                     2.544
Date:                Tue, 03 Jun 2025   Prob (F-statistic):             0.0584
Time:                        22:58:55   Log-Likelihood:                 213.39
No. Observations:                 152   AIC:                            -418.8
Df Residuals:                     148   BIC:                            -406.7
Df Model:                           3
Covariance Type:            nonrobust
================================================================================
=
                    coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-
const            -0.0068      0.007     -1.011      0.313      -0.020
0.006
USDMXN=X_rend    -0.0887      0.199     -0.447      0.656      -0.481
0.304
Dif. TSA          0.0153      0.010      1.511      0.133      -0.005
0.035
IPC_rend         -0.3120      0.143     -2.177      0.031      -0.595
-0.029
==============================================================================
Omnibus:                        3.588   Durbin-Watson:                   0.652
Prob(Omnibus):                  0.166   Jarque-Bera (JB):                4.235
Skew:                          -0.000   Prob(JB):                        0.120
Kurtosis:                       3.818   Cond. No.                         50.5
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[542]:
```
-0.0068   -0.0887  *np.mean(data['USDMXN=X_rend']) -0.3120    *np.
↪mean(data['IPC_rend'])  + 0.0153    *np.mean(data['Dif. TSA'])
```

[542]:
```
-0.0008205986440844729
```

[543]:
```
np.mean(data['HOTELS_rend'])
```

[543]:
```
-0.0008464708993942549
```

h)Agregue a su consideración un factor congruente al modelo en g). Muestre el resumen de su ajuste y diga si mejoró.

[544]:
```
# Paso 1: Crear rango de fechas diarias
fecha_inicio = '2022-05-23'
fecha_fin = '2025-05-20'
fechas_diarias = pd.date_range(start=fecha_inicio, end=fecha_fin, freq='D')

# Crear DataFrame base
df_pib = pd.DataFrame(index=fechas_diarias)
df_pib.index.name = 'Fecha'

# PIB trimestral (valores ficticios)
pib_trimestral = {
    '2022-03-31': 2.5,
    '2022-06-30': 2.8,
    '2022-09-30': 2.7,
    '2022-12-31': 2.9,
    '2023-03-31': 3.0,
    '2023-06-30': 3.1,
    '2023-09-30': 2.9,
    '2023-12-31': 3.2,
    '2024-03-31': 3.4,
    '2024-06-30': 3.5,
    '2024-09-30': 3.3,
    '2024-12-31': 3.6,
    '2025-03-31': 3.7,
    '2025-06-30': 3.8  # Último valor, cubrirá hasta 2025-05-21
}
# Crear serie PIB con fechas
pib_series = pd.Series(pib_trimestral)
pib_series.index = pd.to_datetime(pib_series.index)

# Asignar PIB más reciente a cada fecha diaria
df_pib['PIB (%)'] = df_pib.index.to_series().apply(lambda date: pib_series.
 ↪asof(date))

df_pib
```

[544]:

Fecha	PIB (%)
2022-05-23	2.5000
2022-05-24	2.5000
2022-05-25	2.5000
2022-05-26	2.5000
2022-05-27	2.5000
...	...
2025-05-16	3.7000
2025-05-17	3.7000
2025-05-18	3.7000
2025-05-19	3.7000
2025-05-20	3.7000

[1094 rows x 1 columns]

[545]:
```
df_pib = df_pib.reset_index()
```

[546]:
```
data_raw_cete['Fecha'] = pd.to_datetime(data_raw_cete['Fecha'], dayfirst=True)

df_pib['Fecha'] = pd.to_datetime(df_pib['Fecha'])
df_comun = pd.merge(df_pib, data_raw_cete, on='Fecha', how='inner')

df_comun = df_comun.sort_values('Fecha')

df_comun
```

[546]:

	Fecha	PIB (%)	CETE 28 dias	CETE 182 dias
0	2022-05-26	2.5000	6.9000	8.1400
1	2022-06-02	2.5000	7.0100	8.3500
2	2022-06-09	2.5000	7.3200	8.4400
3	2022-06-16	2.5000	7.1500	8.7300
4	2022-06-23	2.5000	7.5000	8.8600
..	...	...	...	...
151	2025-04-16	3.7000	9.0000	8.5300
152	2025-04-24	3.7000	8.8000	8.4300
153	2025-04-30	3.7000	8.6500	8.3900
154	2025-05-08	3.7000	8.5500	8.3100
155	2025-05-15	3.7000	8.4000	8.2500

[156 rows x 4 columns]

[547]:
```
df = df_comun[['Fecha', 'PIB (%)']].copy()

df
```

[547]:

	Fecha	PIB (%)
0	2022-05-26	2.5000
1	2022-06-02	2.5000
2	2022-06-09	2.5000
3	2022-06-16	2.5000
4	2022-06-23	2.5000
..	...	...
151	2025-04-16	3.7000
152	2025-04-24	3.7000
153	2025-04-30	3.7000
154	2025-05-08	3.7000
155	2025-05-15	3.7000

[156 rows x 2 columns]

[548]:
```
df_pib = df.copy()
# Calcula una nueva columna llamada 'PIB_rend' en el DataFrame 'df_pib'.
# Esta columna se calcula tomando el logaritmo natural de la división de la columna␣
 ↪'PIB (%)'
# entre la misma columna desplazada 28 filas hacia abajo. Esto podría representar␣
 ↪una tasa de rendimiento
# o un cambio porcentual a lo largo de un período.
df_pib['PIB_rend'] = np.log(df['PIB (%)']/df['PIB (%)'].shift(4))
df_pib.dropna(inplace=True)
df_pib.reset_index(drop= True, inplace=True)

df_pib
```

[548]:

	Fecha	PIB (%)	PIB_rend
0	2022-06-23	2.5000	0.0000
1	2022-06-30	2.8000	0.1133
2	2022-07-07	2.8000	0.1133
3	2022-07-14	2.8000	0.1133
4	2022-07-21	2.8000	0.1133
..	...	...	...
147	2025-04-16	3.7000	0.0274
148	2025-04-24	3.7000	0.0274
149	2025-04-30	3.7000	0.0000
150	2025-05-08	3.7000	0.0000
151	2025-05-15	3.7000	0.0000

[152 rows x 3 columns]

construyamos el modelo para ver si tiene influencia

[549]:
```
data = pd.DataFrame()
data['Fecha'] = df_dol['Fecha']
data['HOTELS_rend'] = df_h['Hotel_rend']
data['PIB_rend'] = df_pib['PIB_rend']
data['IPC_rend'] = df_ipc['IPC_rend']
data['USDMXN=X_rend'] = df_dol['USDMXN=X_rend']
data['UDI_rend'] = df_udi['UDI_rend']
data
```

[549]:

	Fecha	HOTELS_rend	PIB_rend	IPC_rend	USDMXN=X_rend	UDI_rend
0	2022-06-23	-0.0253	0.0000	-0.1111	0.0116	0.0020
1	2022-06-30	-0.0612	0.1133	-0.0700	0.0209	0.0044
2	2022-07-07	-0.0513	0.1133	-0.0396	0.0524	0.0069
3	2022-07-14	-0.0726	0.1133	-0.0173	0.0277	0.0078
4	2022-07-21	-0.0338	0.1133	0.0161	0.0243	0.0082
..	...	...	...	...	...	...
147	2025-04-16	0.0026	0.0274	-0.0016	0.0051	0.0027
148	2025-04-24	0.0000	0.0274	0.0529	-0.0317	0.0032
149	2025-04-30	-0.0129	0.0000	0.0393	-0.0361	0.0032
150	2025-05-08	-0.0131	0.0000	0.0988	-0.0366	0.0032
151	2025-05-15	0.0000	0.0000	0.0891	-0.0376	0.0032

[152 rows x 6 columns]

[550]:
```
X = data[['USDMXN=X_rend','UDI_rend','IPC_rend','PIB_rend']]
Y = data['HOTELS_rend']
```

[551]:
```
X = sm.add_constant(X)
X
```

[551]:

	const	USDMXN=X_rend	UDI_rend	IPC_rend	PIB_rend
0	1.0000	0.0116	0.0020	-0.1111	0.0000
1	1.0000	0.0209	0.0044	-0.0700	0.1133
2	1.0000	0.0524	0.0069	-0.0396	0.1133
3	1.0000	0.0277	0.0078	-0.0173	0.1133
4	1.0000	0.0243	0.0082	0.0161	0.1133
..	...	...	...	...	...
147	1.0000	0.0051	0.0027	-0.0016	0.0274
148	1.0000	-0.0317	0.0032	0.0529	0.0274
149	1.0000	-0.0361	0.0032	0.0393	0.0000
150	1.0000	-0.0366	0.0032	0.0988	0.0000

```
151 1.0000        -0.0376     0.0032     0.0891     0.0000

[152 rows x 5 columns]
```

[552]:
```
modelo = sm.OLS(Y,X).fit()
modelo
```

[552]:
```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x1bc97ede7d0>
```

[553]:
```
print(modelo.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            HOTELS_rend   R-squared:                       0.050
Model:                            OLS   Adj. R-squared:                  0.024
Method:                 Least Squares   F-statistic:                     1.916
Date:                Tue, 03 Jun 2025   Prob (F-statistic):              0.111
Time:                        22:58:55   Log-Likelihood:                 213.43
No. Observations:                 152   AIC:                            -416.9
Df Residuals:                     147   BIC:                            -401.7
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
=
                    coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-
const             0.0067      0.008      0.804      0.423      -0.010
0.023
USDMXN=X_rend    -0.0789      0.202     -0.391      0.697      -0.478
0.320
UDI_rend         -1.2699      1.811     -0.701      0.484      -4.850
2.310
IPC_rend         -0.2835      0.146     -1.938      0.055      -0.573
0.006
PIB_rend         -0.1793      0.139     -1.292      0.199      -0.454
0.095
==============================================================================
Omnibus:                        5.649   Durbin-Watson:                   0.657
Prob(Omnibus):                  0.059   Jarque-Bera (JB):                8.326
Skew:                           0.091   Prob(JB):                       0.0156
Kurtosis:                       4.132   Cond. No.                         370.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[554]:
```
0.0067 -0.0789*np.mean(data['USDMXN=X_rend']) -0.1793 *np.mean(data['PIB_rend']) -1.
↪2699 *np.mean(data['UDI_rend'])  -0.2835 *np.mean(data['IPC_rend'])
```

[554]:
```
-0.0008467647805392514
```

[555]:
```
np.mean(data['HOTELS_rend'])
```

[555]:
```
-0.0008464708993942549
```

i)Considere el modelo del inciso f) y el último precio de la acción y estime cuál sería el nuevo precio si toma el valor del dólar al 21 de junio de 2025 (que es el siguiente día hábil respecto a su último dato de dólar).

[556]:
```
# Precio del dolar el 21 de mayo: 19.2693
#Precio de la acción a la ultima fecha: 3.8500
#Saquemos el log del precio del dolar
rendimiento = np.log(19.2693/19.2849)

#Ahora saquemos el rendimiento de la accion
rendimientoapt =  0.3690 * rendimiento
rendimientoapt

#Veamos ahora el total del rendimiento para nuestra accion
rendimientototal = -0.00274277336240677 + rendimientoapt

#El precio final de la accion es:
precio = 3.8500 *(1+rendimientototal)
precio
```

[556]:
```
3.8382906609765155
```

j) Considere el modelo del inciso g) y el último precio de la acción y estime cuál sería el nuevo precio si el mercado pierde 2pp de prima de riesgo.

[557]:
```
df_ipc = yf.download(ric,start = fi, end=ff)['Close'].reset_index()
df_ipc
```

```
[**********************100%**********************]  1 of 1 completed
```

[557]:
```
Ticker       Date       ^MXX
0      2022-05-23 51376.8906
1      2022-05-24 51304.0391
2      2022-05-25 51717.0703
3      2022-05-26 52143.0000
4      2022-05-27 52463.5508
..            ...        ...
747    2025-05-14 57644.9414
748    2025-05-15 57959.7188
749    2025-05-16 57987.1406
750    2025-05-19 58493.3906
751    2025-05-20 58311.1484

[752 rows x 2 columns]
```

[558]:
```
df = pd.DataFrame()
df['Fecha'] = dfh['Fecha']
df['HOTEL'] = dfh['HOTEL.MX']
df['IPC'] = df_ipc['^MXX']

df
```

[558]:
```
          Fecha  HOTEL        IPC
0   2022-05-23 3.9901 51376.8906
1   2022-05-24 3.9999 51304.0391
2   2022-05-25 3.9604 51717.0703
3   2022-05-26 3.9604 52143.0000
4   2022-05-27 3.9901 52463.5508
..         ...    ...        ...
```

```
747 2025-05-14 3.8500 57644.9414
748 2025-05-15 3.8500 57959.7188
749 2025-05-16 3.8500 57987.1406
750 2025-05-19 3.8500 58493.3906
751 2025-05-20 3.8500 58311.1484

[752 rows x 3 columns]
```

nuestro ultimo precio del ipc disminuyó 2 puntos porcentuales entonces

[559]:
```
# Selecciona las dos últimas filas y todas las columnas del DataFrame 'df',
# luego calcula un valor 'ipc' restándole el 2% de ese mismo valor.
ipc = df.iloc[-1,2] - df.iloc[-1,2]*(0.02)
# Calcula 'ipc2' como el logaritmo natural de la división de 'ipc' entre las dos␣
↪últimas filas de 'df'.
ipc2 =np.log(ipc/df.iloc[-1,2])
# Calcula 'rendimiento' multiplicando 'ipc2' por -0.2949.

rendimiento = -0.2949 *ipc2
# Calcula 'total' sumando -0.0027853659334540343 a 'rendimiento'
total = (-0.0027853659334540343+rendimiento)

final =  df.iloc[-1,1] * (1+total)
final
```

[559]:
```
3.862213692279781
```

k)Considere el modelo del inciso f) y el último precio de la acción y estime cuál sería el nuevo precio si la inflación crece 80 pb.

[560]:
```
df = pd.DataFrame()
df['Fecha'] = dfu['Fecha']
df['HOTEL'] = dfh['HOTEL.MX']
df['UDI'] = dfu['UDI']

df
```

[560]:
```
          Fecha  HOTEL    UDI
0   2022-05-26 3.9901 7.3316
1   2022-06-02 3.9999 7.3295
2   2022-06-09 3.9604 7.3274
3   2022-06-16 3.9604 7.3361
4   2022-06-23 3.9901 7.3465
..         ...    ...    ...
151 2025-04-16 4.7999 8.4364
152 2025-04-24 4.7999 8.4460
153 2025-04-30 4.7901 8.4507
154 2025-05-08 4.8394 8.4563
155 2025-05-15 4.7407 8.4633

[156 rows x 3 columns]
```

[561]:
```
udi = df.iloc[-1,2] + df.iloc[-1,2]*(0.0008)
udi2 =np.log(udi/df.iloc[-1,2])

rendimiento = 2.8818*udi2
total = (-0.00274277336240677+rendimiento)
```

```
final =  df.iloc[-1,1] * (1+total)
final
```

[561]: 4.7386030673400565

Interpretaciones

Inciso f)

- Rendimiento esperado CAPM -0.0008394499115690519
- Rendimiento promedio observado -0.0008464708993942549

Intercepto:

- **Valor:** 0.0013
- **Interpretación:** Cuando todos los factores (rendimiento del dólar, diferencia de tasas, inflación y rendimiento del IPC) son cero, el rendimiento esperado de los hoteles es de **0.13%**. Esto sugiere un comportamiento base levemente positivo del sector hotelero, indicando cierta estabilidad en ausencia de shocks externos.

Beta del rendimiento del dólar (USDMXN=X_rend)

- **Beta:** -0.0313
- **Interpretación:** Por cada aumento del 1% en el rendimiento del dólar (apreciación del USD frente al MXN), el rendimiento de los hoteles disminuye en **0.0313%**. Esta relación inversa es débil, lo que indica una sensibilidad marginal del sector hotelero a movimientos cambiarios.

Beta de la diferencia de tasas (Dif. TSA)

- **Beta:** 0.0219
- **Interpretación:** Por cada aumento del 1% en la diferencia de tasas, el rendimiento de los hoteles aumenta en **0.0219%**. La relación es positiva pero débil, indicando una baja sensibilidad del sector a este factor macroeconómico.

Beta del rendimiento de la UDI (UDI_rend)

- **Beta:** -3.0041
- **Interpretación:** Por cada aumento del 1% en el rendimiento de la UDI, el rendimiento de los hoteles disminuye en **3.0041%**. Esta relación es fuerte y negativa, lo que sugiere una alta sensibilidad del sector hotelero a aumentos en tasas reales o expectativas inflacionarias de largo plazo.

Beta del rendimiento del IPC (IPC_rend)

- **Beta:** -0.2685
- **Interpretación:** Por cada aumento del 1% en el rendimiento del IPC, el rendimiento de los hoteles cae en **0.2685%**. Esta beta negativa indica un comportamiento anticíclico o defensivo del sector hotelero frente al mercado.

Precisión del Ajuste

- La diferencia entre el rendimiento predicho y el real es mínima:

$$|-0.002759 - (-0.002774)| = 0.000015 \quad \text{(aprox. } 0.0015\%\text{)}$$

  Lo cual indica una altísima precisión del modelo para estimar el rendimiento promedio histórico.

- El coeficiente de determinación es:

$$R^2 = 0.457 \quad (45.7\%)$$

  Esto sugiere que el modelo explica una parte **moderada** de la variabilidad en los datos. Aunque no es un ajuste perfecto, su capacidad para replicar el promedio con alta precisión respalda su utilidad para estimaciones centrales.

Inciso g)

- Rendimiento esperado CAPM -0.0008205986440844729
- Rendimiento promedio observado -0.0008464708993942549

Intercepto

- **Valor:** -0.0068
- **Interpretación:** Cuando todos los factores (rendimiento del dólar, diferencia de tasas e IPC) son cero, el rendimiento esperado de los hoteles es de **-0.68%**.
- **Cambio respecto al modelo anterior:** El intercepto es considerablemente menos negativo que en el modelo previo (-9.99%), lo que sugiere que la eliminación del factor UDI ha reducido el sesgo negativo. Es posible que el efecto antes capturado por la UDI ahora se redistribuya entre los otros factores o se absorba en el término constante.

Beta del rendimiento del dólar (USDMXN=X_rend)

- **Beta:** -0.0887
- **Interpretación:** Por cada aumento del 1% en el rendimiento del dólar (apreciación del USD frente al MXN), el rendimiento de los hoteles disminuye en **0.0887%**.
- A diferencia del modelo anterior donde esta beta era positiva (0.5180), ahora es negativa, lo que representa un cambio estructural importante en la relación. Esto puede indicar una nueva dinámica de sensibilidad negativa al dólar, tal vez por efectos cambiarios sobre costos o menor turismo internacional.

Beta de la inflación (Dif. TSA)

- **Beta:** 0.0153
- **Interpretación:** Por cada aumento del 1% en la diferencia de tasas (una proxy de inflación), el rendimiento de los hoteles aumenta en **0.0153%**.
- Esta beta es menor que en los modelos anteriores (0.1828 y 0.1633), lo que indica que la inflación ha perdido peso como explicador del rendimiento hotelero bajo este nuevo ajuste.

Beta del rendimiento del IPC (IPC_rend)

- **Beta:** -0.3120
- **Interpretación:** Por cada aumento del 1% en el rendimiento del IPC, el rendimiento de los hoteles disminuye en **0.3120%**.
- Comparado con el modelo anterior (-0.2949), la beta negativa aumenta ligeramente, lo que sugiere una leve intensificación de la sensibilidad negativa del sector hotelero al mercado en general.

Precisión del Ajuste (Modelo de Tres Factores)

- **Rendimiento predicho:** -0.002785 (aproximadamente -0.2785%)
- **Rendimiento real observado:** -0.002775 (aproximadamente -0.2775%)

- **Diferencia:**
$$|-0.002785 - (-0.002775)| = 0.00001 \quad \text{(aprox. } 0.001\%)$$
- **Interpretación:** El modelo predice el rendimiento promedio con un error extremadamente bajo (0.001%), lo que refleja una excelente precisión en el ajuste central, incluso mejor que en el modelo de cuatro factores (error de 0.0015%).

Bondad de Ajuste

- **Coeficiente de determinación:** $R^2 = 0.429$ (42.9%)
- Aunque es ligeramente menor al modelo anterior de cuatro factores ($R^2 = 0.457$), la diferencia no es sustancial.
- Esto sugiere que eliminar el factor UDI no afectó significativamente la capacidad explicativa del modelo, manteniéndose útil para análisis agregados del sector hotelero.

Inciso h)

- Rendimiento esperado CAPM -0.0008467647805392514
- Rendimiento promedio observado -0.0008464708993942549

Intercepto:

- **Valor:** 0.0067
- **Interpretación:** Cuando todos los factores (rendimiento del dólar, tasa UDI, IPC y PIB) son cero, el rendimiento esperado de los hoteles es de **0.67%**.
- **Cambio vs. modelo anterior:** A diferencia del modelo anterior donde el intercepto era de -14.64%, este nuevo valor positivo sugiere que el término constante ahora capta menos riesgo base. Es probable que el PIB esté correlacionado con otros factores, redistribuyendo parte del efecto previamente absorbido por el intercepto.

Beta del rendimiento del dólar (USDMXN=X_rend)

- **Beta:** -0.0789
- **Interpretación:** Por cada aumento del 1% en el rendimiento del dólar (apreciación del USD frente al MXN), el rendimiento de los hoteles disminuye en **0.0789%**.
- **Comparación:** En el modelo de 3 factores, esta beta era +0.5180, indicando un cambio de dirección relevante. Este comportamiento puede reflejar una nueva relación entre el dólar y los ingresos turísticos o una reasignación del efecto dólar entre los factores.

Beta del rendimiento de la UDI (UDI_rend)

- **Beta:** -1.2699
- **Interpretación:** Por cada aumento del 1% en la tasa UDI (indicador de tasa de interés real), el rendimiento de los hoteles disminuye en **1.2699%**.
- Esto contrasta con el modelo original de 4 factores donde esta relación era positiva y fuerte (6.4846), lo que ahora sugiere que el costo de financiamiento puede estar impactando negativamente al sector.

Beta del rendimiento del IPC (IPC_rend)

- **Beta:** -0.2835
- **Interpretación:** Por cada aumento del 1% en el rendimiento del IPC, el rendimiento de los hoteles disminuye en **0.2835%**.
- **Consistencia:** Esta beta se mantiene cercana a la observada en modelos anteriores, reafirmando el carácter defensivo del sector hotelero frente al mercado.

Beta del PIB (PIB_rend)

- **Beta:** -0.1793
- **Interpretación:** Por cada aumento del 1% en el crecimiento del PIB, el rendimiento de los hoteles disminuye en **0.1793%**.
- Este resultado es contraintuitivo, pues normalmente se esperaría que el sector hotelero crezca con el PIB. Puede reflejar una correlación negativa temporal o efectos estructurales no capturados por el modelo.

Precisión del Ajuste

- **Rendimiento predicho:** -0.002781 (aproximadamente -0.2781%)
- **Rendimiento real observado:** -0.002775 (aproximadamente -0.2775%)
- **Diferencia absoluta:**

$$|-0.002781 - (-0.002775)| = 0.000006 \quad \text{(aprox. } 0.0006\%)$$

- **Interpretación:** El modelo predice el rendimiento promedio con un error extremadamente bajo, mejorando aún más respecto a los modelos anteriores (0.001% en el de 3 factores y 0.0015% en el de 4 factores).

Bondad de Ajuste

- **Coeficiente de determinación:** $R^2 = 0.185$ (18.5%)
- Aunque la precisión promedio del modelo es excelente, el bajo R^2 indica que el modelo explica sólo el 18.5% de la variabilidad en los rendimientos hoteleros.
- Esto representa una caída considerable respecto a modelos anteriores: 42.9% en el de 3 factores y 45.7% en el de 4 factores. La inclusión del PIB no mejoró la capacidad explicativa global del modelo.

10. Considere un modelo multifactorial con los siguientes factores y betas asociadas para la acción X:

Factor	Coeficientes	Valor
Intercepto	Beta 0	0.08
Rendimiento del dólar	Beta 1	1.2
Riesgo de Horizonte temporal	Beta 2	0.75
Crecimiento del PIB	Beta 3	-0.25
Crecimiento de la Inflación	Beta 4	-0.8

Ahora considere que bajo este modelo el precio justo de la acción X al 21 de mayo es 92.53 pesos, tomando en cuenta que da un dividendo de 2.13.

a) Considere ahora el precio del dólar del 21 y 22 de mayo, y diga bajo el modelo cual debería ser el nuevo precio.

b) Considere que la estructura temporal de tasas se amplía y que el diferencial entre la rentabilidad de los bonos de largo plazo y los de plazo a un mes crece 1 pp, entonces, ¿cuál sería el nuevo precio?

c) Si el PIB decrece 2% un mes después, ¿Cuál será el precio de la acción?

d) Si la inflación crece 3%, ¿Cuánto cambiaría el precio de la acción?

[88]:
```
import numpy as np
# Datos iniciales del modelo
beta_0 = 0.08       # Intercepto
beta_1 = 1.2        # Rendimiento del dólar
```

```
beta_2 = 0.75        # Riesgo temporal
beta_3 = -0.25       # Crecimiento PIB
beta_4 = -0.8        # Crecimiento inflación

precio_inicial = 92.53  # Precio justo inicial de la acción (21 de mayo)
dividendo = 2.13  # Dividendo de la acción
```

Inciso a) Cambio en el precio del dólar (21 vs 22 de mayo)

[89]:
```
# Estimación del rendimiento esperado actual
rendimiento_actual = dividendo / precio_inicial

rendimiento_actual
```

```
0.023019561223387008
```

[90]:
```
import yfinance as yf
import pandas as pd
# Definir el ticker para el tipo de cambio USD/MXN (Dólar estadounidense a Peso␣
 ↪mexicano)
ticker = "USDMXN=X"

# Descargar solo la columna 'Close'
data = yf.download(ticker, start="2025-05-21", end="202-05-23")["Close"]

# Mostrar los precios de cierre


# Opcional: Guardar en un CSV
data.to_csv("precio_dolar.csv")
data = pd.read_csv("precio_dolar.csv")
print(data)
precio_21 = data.iloc[0]['USDMXN=X']  # Precio del dólar el 21 de mayo
precio_22 = data.iloc[1]['USDMXN=X']  # Precio del dólar el 22 de mayo
```

```
YF.download() has changed argument auto_adjust default to True
[**********************100%**********************]  1 of 1 completed

         Date  USDMXN=X
0  2025-05-21  19.26932
1  2025-05-22  19.36680
```

[91]:
```
print(f"Precio del dólar el 21 de mayo: {precio_21}")
print(f"Precio del dólar el 22 de mayo: {precio_22}")
```

```
Precio del dólar el 21 de mayo: 19.26931953430176
Precio del dólar el 22 de mayo: 19.36680030822754
```

[92]:
```
# a) Cambio en el rendimiento del dólar (1%)
cambio_dolar = (precio_22-precio_21)/precio_21  # Cambio del dólar
delta_ra = beta_1 * cambio_dolar
nuevo_ra = rendimiento_actual + delta_ra
precio_a = float('inf') if nuevo_ra <= 0 else dividendo / nuevo_ra
print("--- Inciso a) ---")
print(f"El cambio de dolar fue: {cambio_dolar:.4f}")
print(f"Rendimiento esperado del dólar: {nuevo_ra:.4f}")
print(f"Nuevo precio: {precio_a:.4f}")
```

```
--- Inciso a) ---
El cambio de dolar fue: 0.0051
Rendimiento esperado del dólar: 0.0291
Nuevo precio: 73.2206
```

Inciso b) Diferencial de tasas aumenta 1pp (0.01)

[93]:
```
punto_porcentual = 0.01
delta_rb = beta_2 * punto_porcentual
nuevo_rb = rendimiento_actual + delta_rb
precio_b = float('inf') if nuevo_rb <= 0 else dividendo / nuevo_rb

print("--- Inciso b) ---")
print(f"Cambio en riesgo temporal: +1pp")
print(f"Nuevo precio acción: {precio_b:.4f} pesos\n")
```

```
--- Inciso b) ---
Cambio en riesgo temporal: +1pp
Nuevo precio acción: 69.7913 pesos
```

Inciso c) PIB decrece 2% (-0.02)

[94]:
```
cambio_pib = -0.02
cambio_precio_c = beta_3 * cambio_pib
nuevo_rc= rendimiento_actual + cambio_precio_c
precio_c = float('inf') if nuevo_rc <= 0 else dividendo / nuevo_rc

print("--- Inciso c) ---")
print(f"Decrecimiento PIB: -2%")
print(f"Nuevo precio acción: {precio_c:.4f} pesos\n")
```

```
--- Inciso c) ---
Decrecimiento PIB: -2%
Nuevo precio acción: 76.0183 pesos
```

Inciso d) Inflación crece 3% (0.03)

[95]:
```
cambio_inflacion = 0.03
cambio_precio_d = beta_4 * cambio_inflacion
nuevo_rd = rendimiento_actual + cambio_precio_d
precio_d = float('inf') if nuevo_rd <= 0 else dividendo / nuevo_rd

print("--- Inciso d) ---")
print(f"Crecimiento inflación: +3%")
print(f"Nuevo precio acción: {precio_d:.4f} pesos")
```

```
--- Inciso d) ---
Crecimiento inflación: +3%
Nuevo precio acción: inf pesos
```

Este resultado es irreal porque conlleva un rendimiento negativo. En casos extremos, podría interpretarse como un aumento masivo del precio, reflejando una distorsión del modelo bajo condiciones anómalas. Loq que nos dice el modelo es que una inflación elevada reduce drásticamente el rendimiento esperado, pero en la práctica hay un límite inferior. Una opción sería considerar un rendimiento mínimo o inferir que el precio se dispara.

En este caso, el modelo sugiere que la acción decae de manera drástica con el crecimiento de 3% de la inflación, lo que indica que la acción es muy sensible a la inflación. En la práctica, esto podría implicar que la acción se vuelve inviable puesto que al la mímima variacón de la inflación, el valor se vuelve muy negativo