PROYECTO DE OPTIMIZACIÓN MATEMÁTICO DE LA CARTERA DEL UPV Investment Club

In [1]:
```
import yfinance as yf
import numpy as np
import pandas as pd
```

Una vez importadas las librerías definimos las primeras funciones del código: Recoger datos desde la creación de la cartera y calcular los rendimientos logarítmicos diarios de los activos en cartera.

In [3]:
```
def obtener_datos(tickers, start_date):
    # Descargar datos históricos de precios de cierre ajustados
    data = yf.download(tickers, start=start_date)
    precios_cierre = data['Adj Close']
    return precios_cierre

def calcular_rendimientos_logaritmicos(precios_cierre):
    # Calcular los rendimientos logaritmicos diarios
    rendimientos_logaritmicos = np.log(precios_cierre / precios_cierre.shift(1)).dropna()
    return rendimientos_logaritmicos
def calcular_rendimiento_logaritmico(rendimientos_logaritmicos):
    rendimiento_logaritmico=np.mean(rendimientos_logaritmicos)
    return rendimiento_logaritmico
```

Definimos los tickers de nuestro portafolio y una fecha cercana a la creación de la cartera para ajustar el riesgo y rendimiento respecto a datos cercanos e influyentes para nosotros

In [4]:
```
# Definir los tickers de las acciones y la fecha de inicio
tickers = ['ULTA','TXN','SOM.L','POOL','PLUS.L','P8X.BE','MSFT', 'MC', 'EVO.ST','DHR','DNP.WA']
start_date = '2023-01-01'
```

In [5]:
```
# Obtener los datos de precios de cierre
precios_cierre = obtener_datos(tickers, start_date)

# Calcular los rendimientos logaritmicos diarios
rl= calcular_rendimientos_logaritmicos(precios_cierre)

# Calcular el rendimiento esperado diario
rendimiento_esperado_diario = calcular_rendimiento_logaritmico(rl)

# Calcular el rendimiento esperado anual
dias_negociacion_anuales = 252
rendimiento_esperado_anual = rendimiento_esperado_diario * dias_negociacion_anuales
```

```
[**********100%%**********]  11 of 11 completed
```

In [111…
```
# Mostrar los resultados
print("El rendimiento esperado anual de los activos es:\n", rendimiento_esperado_anual)
```

```
El rendimiento esperado anual de los activos es:
 DHR       0.041026
DNP.WA    -0.044361
EVO.ST     0.092589
MC         0.353549
MSFT       0.616716
P8X.BE     0.265510
PLUS.L     0.250875
POOL       0.232413
SOM.L     -0.175416
TXN        0.075269
ULTA       0.108784
dtype: float64
```

El uso de los cierres ajustados es lo que hace que tanto el rendimiento de PAX como el de PLUS500 se diferencie tanto del incremento del precio interanual, ya que tenemos en cuenta el ajuste debido a los dividendos pagados, en mayor parte, y demás ajustes por eventos corporativos de estas 2 empresas.

Una vez tenemos los rendimientos esperados podemos crear la matriz varianza-covarianza para obtener el riesgo de cada activo (varianza) y la covarianza entre ellos

In [112…
```
def calcular_matriz_covarianza(rendimientos_simples):
    # Calcular la matriz de covarianza de los rendimientos logaritmicos
    matriz_covarianza = rendimientos_simples.cov()
    return matriz_covarianza
```

In [113…
```
# Calcular la matriz de covarianza
matriz_covarianza = calcular_matriz_covarianza(rl)

print("\nMatriz de covarianza:")
print(matriz_covarianza)
```

```
Matriz de covarianza:
             DHR     DNP.WA    EVO.ST        MC      MSFT    P8X.BE    PLUS.L  \
DHR     0.000232  0.000058  0.000030  0.000109  0.000052  0.000016  0.000008
DNP.WA  0.000058  0.000459  0.000094  0.000049  0.000022  0.000095  0.000003
EVO.ST  0.000030  0.000094  0.000327  0.000076  0.000063  0.000042  0.000045
MC      0.000109  0.000049  0.000076  0.000422  0.000030  0.000042  0.000039
MSFT    0.000052  0.000022  0.000063  0.000030  0.000236  0.000025  0.000006
P8X.BE  0.000016  0.000095  0.000042  0.000042  0.000025  0.001241  0.000045
PLUS.L  0.000008  0.000003  0.000045  0.000039  0.000006  0.000045  0.000191
POOL    0.000133  0.000052  0.000097  0.000167  0.000070  0.000091  0.000042
SOM.L   0.000019  0.000013  0.000071  0.000015  0.000047 -0.000116  0.000020
TXN     0.000079  0.000021  0.000058  0.000115  0.000072  0.000050  0.000018
ULTA    0.000042  0.000028  0.000029  0.000077  0.000099  0.000080  0.000041

             POOL     SOM.L       TXN      ULTA
DHR     0.000133  0.000019  0.000079  0.000042
DNP.WA  0.000052  0.000013  0.000021  0.000028
EVO.ST  0.000097  0.000071  0.000058  0.000029
MC      0.000167  0.000015  0.000115  0.000077
MSFT    0.000070  0.000047  0.000072  0.000099
P8X.BE  0.000091 -0.000116  0.000050  0.000080
PLUS.L  0.000042  0.000020  0.000018  0.000041
POOL    0.000379  0.000031  0.000135  0.000091
SOM.L   0.000031  0.000512  0.000024 -0.000024
TXN     0.000135  0.000024  0.000219  0.000037
ULTA    0.000091 -0.000024  0.000037  0.000298
```

Queremos comprobar que la matriz es definida positiva, al ser la matriz de covarianza simétrica e invertible (no debería haber activos redundantes) y la varianza positiva por definición. Esto también asegura que el punto hallado sea un mínimo, que es lo buscado.

In [114…
```
# Calcula los eigenvalores de la matriz de covarianza
eigenvalores = np.linalg.eigvals(matriz_covarianza)

# Verifica si todos los eigenvalores son positivos
es_definida_positiva = np.all(eigenvalores > 0)

# Muestra el resultado
if es_definida_positiva:
    print("La matriz de covarianza es definida positiva.")
else:
    print("La matriz de covarianza NO es definida positiva.")
```

```
La matriz de covarianza es definida positiva.
```

Ahora para los siguientes cálculos nos hará falta la inversa de la matriz de covarianza

In [115…
```
# Calcula la inversa de la matriz de covarianza
matriz_covarianza_inversa = np.linalg.inv(matriz_covarianza)
```

Ahora vamos a proceder a calcular el vector de pesos (w) que minimiza el riesgo gracias a un problema general de optimización que hemos adaptado a nuestro caso, habiendo encontrado este resultado fruto de usar Multiplicadores de Lagrange con la condición de pesos, las hipótesis del problema general están satisfechas gracias a la estructura del problema, y que la matriz de covarianza sea definida positiva.

In [116…
```
# Calcula los valores de w
unos = np.ones(len(rendimiento_esperado_anual))
w_numerador = np.dot(matriz_covarianza_inversa, unos)
w_denominador = np.dot(np.dot(unos, matriz_covarianza_inversa), unos)
w =(w_numerador / w_denominador)

# Calcula el riesgo mínimo
riesgo_minimo = np.dot(np.dot(w, matriz_covarianza), w.T)

# Calcula el rendimiento
rendimiento = np.dot(rendimiento_esperado_anual, w.T)

# Muestra los resultados
print("Valores de w:", w)
print("Riesgo mínimo:", riesgo_minimo)
print("Rendimiento:", rendimiento)
```

```
Valores de w: [ 0.14737435  0.07952134  0.04383241  0.01674819  0.15187154  0.0263288
  0.25047962 -0.06769626  0.09094684  0.12802874  0.13256443]
Riesgo mínimo: 6.066493703746242e-05
Rendimiento: 0.16835979516564883
```

Podemos observar como para algún activo (POOL) da un peso negativo, esto implicaría estar en corto en ese activo, lo cual no va con nuestro estilo de inversión. Ahora surgen varias opciones de rebalanceo o incluir como segunda condición que w_i>=0.

También podemos observar como teóricamente el rendimiento esperado anual es del 16.8% y el riesgo del 0.006% a fecha 25/03/2024

%Realizado en Mathematica

In [117…
```
# Definición de los pesos w
w = {
    'w1': 0.127615,
    'w2': 0.0813164,
    'w3': 0.0355548,
    'w4': 0.00552349,
    'w5': 0.150369,
    'w6': 0.0244974,
    'w7': 0.249733,
    'w8': 7.45822e-6,
    'w9': 0.0916031,
    'w10': 0.108875,
    'w11': 0.124906
}

# Calcula el riesgo mínimo
riesgo_minimo = np.dot(np.dot(list(w.values()), matriz_covarianza), list(w.values()))

# Calcula el rendimiento
rendimiento = np.dot(rendimiento_esperado_anual, list(w.values()))

# Muestra los resultados
print("Valores de w:", w)
print("Riesgo mínimo:", riesgo_minimo)
print("Rendimiento:", rendimiento)
```

```
Valores de w: {'w1': 0.127615, 'w2': 0.0813164, 'w3': 0.0355548, 'w4': 0.00552349, 'w5': 0.150369, 'w6': 0.0244974, 'w7': 0.249733, 'w8': 7.45822e-06, 'w9': 0.0916031, 'w10': 0.108
875, 'w11': 0.124906}
Riesgo mínimo: 6.173668842425698e-05
Rendimiento: 0.17447971096170817
```

Con estos datos podemos observar un rendimiento esperado del 17.44% y un riesgo del 0.006% a fecha 25/03/2024, donde no tenemos ningun activo en corto a diferencia de la resolución anterior, pero nuestro riesgo es mayor.

La gran diferencia entre este rendimiento teórico y el de nuestra cartera se puede deber a varios factores: -Nuestra cartera no ha tenido todos los activos generando rendimientos positivos desde el 01/01/23, esta cartera sí - La cartera está optimizada respecto a datos pasados luego su comportamiento respecto a estos datos es el mejor, puede que si estos pesos se dejan un mes den un rendimiento muy buena, pero no tan espectacular como este *La optimización respecto a riesgos futuros sería el siguiente paso, así como su rebalanceo*

Como extensión podemos aplicar a nuestro análisis también la fórmula del modelo CAPM (para mero hecho comparativo), aunque este es un método que no se utiliza actualmente para optimizar, aun así es visto en las escuelas debido a su importancia y a la de todo el estudio de Markowitz los cuales fueron los predecesores.

In [118…
```
numerador_w2 = np.dot((rendimiento_esperado_anual - 0.0402 * unos),matriz_covarianza_inversa)
denominador_w2 = np.dot(np.dot((rendimiento_esperado_anual - 0.0402 * unos),matriz_covarianza_inversa),unos)
w2 = numerador_w2 / denominador_w2

# Muestra el resultado
print("El valor de w2 es:", w2)

# Calcula el riesgo mínimo
riesgo_minimo = np.dot(np.dot(w2, matriz_covarianza), w2.T)

# Calcula el rendimiento
rendimiento = np.dot(rendimiento_esperado_anual, w2.T)

# Muestra los resultados
print("Riesgo mínimo:", riesgo_minimo)
print("Rendimiento:", rendimiento)
```

```
El valor de w2 es: [-0.35529724 -0.10802558 -0.18583804  0.45298634  1.40388695  0.02985926
  0.50630134  0.12915896 -0.29832237 -0.52569936 -0.04901027]
Riesgo mínimo: 0.0005354373191413565
Rendimiento: 1.1713564883488548
```