Resumen Ejecutivo

Proyecto Final Herramientas de Programación

Nuestro proyecto es un programa en Python que tiene el objetivo de ayudar a personas interesadas en invertir en acciones. El usuario ingresa códigos de acciones (tickers), y el programa analiza cómo organizar un grupo de inversiones equilibrado. Utiliza el método Black-Litterman para mejorar las predicciones de ganancias, y luego hace una simulación Monte Carlo para ver posibles riesgos futuros. Adicionalmente, muestra gráficos y números claros para entender las decisiones a tomar.

El programa resuelve un problema muy común en las inversiones: cómo elegir acciones que den buenas ganancias sin mucho riesgo. En lugar de solo usar datos pasados, que pueden ser inestables, la utilización de Black-Litterman permite combinar ideas del mercado con opiniones simples del usuario, de forma que se crea un portafolio más confiable. Por su parte, la simulación de Monte Carlo imagina muchos escenarios futuros para mostrar qué podría pasar, como pérdidas posibles. Puede ser útil tanto para estudiantes o principiantes en finanzas, como para compañías de distintos sectores.

El código está organizado en partes claras. Primero se importan las librerías necesarias para crear la herramienta. Posteriormente se define una función de carga de datos, que recibe una lista de tickers, y utiliza yfinance para descargar precios de cierre de los últimos 5 años. Luego crea un diccionario con series de precios para cada ticker válido (omitiendo los que no tienen datos), y hace un dataframe con todos los precios limpios. Esta parte se ejecuta cuando se llama más adelante.

Luego se crea una clase llamada Activo, que es una plantilla para cada acción. Este prepara variables para precios, retornos, retorno esperado, riesgo, beta, y VaR. La función cargar extrae los precios del ticker, calcula retornos diarios, y anualiza las métricas de retorno esperado, riesgo, beta y VaR 95%.

La clase Portafolio es la central. En esta, se establece una función que toma tickers y una tasa libre de riesgo por defecto de 0.01, limpia los tickers, llama a cargar_datos_global para obtener precios, crea objetos Activo para cada uno válido, calculo pesos iguales iniciales, y prepara variables para covarianza, retornos ajustados, métricas óptimas y backtest. La función calcular_cov junta retornos de activos en un DataFrame, calcula la matriz de covarianza anualizada y devuelve True si hay al menos 2 activos. La función Black_litterman define matrices como P (identidad), Q (retornos históricos), Omega (incertidumbre), pesos de mercado iguales, y ajusta retornos esperados con tau=0.05. La función Optimizar usa mu_bl para minimizar el Sharpe negativo, calcula el peso óptimo de los tickers, las métricas de este portafolio, y el backtest. Generar_frontera simula 100 portafolios aleatorios con pesos distintos, calcula sus retorno, riesgo y Sharpe, y los guarda en un Dataframe. Resumen imprime un reporte con métricas, pesos y VaR promedio, une todo para el análisis principal.

La sección de gráficas define graficar, que usa el DataFrame de frontera para un scatter plot de riesgo vs retorno coloreado por Sharpe, con el óptimo en rojo. Es interactivo con un slider para tamaño de puntos. Metricas muestra retorno, beta y VaR para cada acción como descripción en la gráfica.

La sección de ejecución es donde se define a función obtener_tickers, que pide al usuario ingresar tickers uno por uno (hasta 30 o hasta introducir 'done'), o usa defaults. Luego, donde está el comentario de ejecutar todo, llama a obtener_tickers, crea el portafolio, corre black_litterman, optimizar, generar_frontera, resumen, graficar y metricas.

Después, hay una función grafico_pesos_profesional, que crea un gráfico de dona con los pesos ideales de los tickers. Finalmente, está la función montecarlo_simulación, que usa mu_bl, cov y pesos para simular 10,000 retornos normales, y dibuja un histograma con líneas para retorno óptimo y VaR 95%.