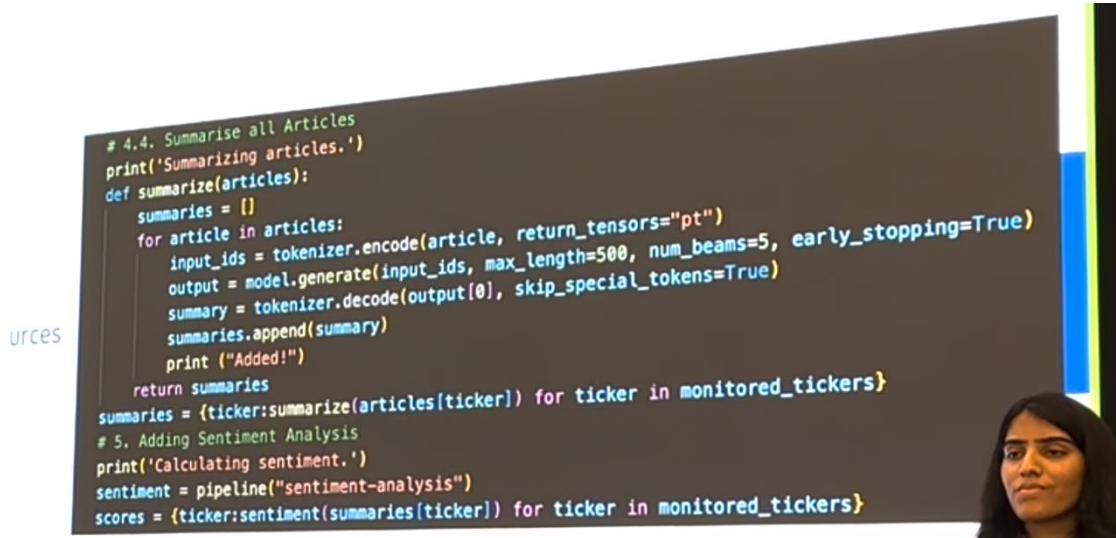


1. Sentiment:

By using the AI deep learning sector, AI could be used to analyse humans' natural language in news, reports, and articles related to the macroeconomic situation.

Sample code from the India team:



```
# 4.4. Summarise all Articles
print('Summarizing articles.')
def summarize(articles):
    summaries = []
    for article in articles:
        input_ids = tokenizer.encode(article, return_tensors="pt")
        output = model.generate(input_ids, max_length=500, num_beams=5, early_stopping=True)
        summary = tokenizer.decode(output[0], skip_special_tokens=True)
        summaries.append(summary)
        print ("Added!")
    return summaries
summaries = {ticker:summarize(articles[ticker]) for ticker in monitored_tickers}

# 5. Adding Sentiment Analysis
print('Calculating sentiment.')
sentiment = pipeline("sentiment-analysis")
scores = {ticker:sentiment(summaries[ticker]) for ticker in monitored_tickers}
```

First, we need to gather articles under each category: news about the company, news about the macroeconomic situation, reports from analysts about the stock, etc.

Then, we can rate each article's positivity on a scale from -10 to 10, with 0 indicating neutrality. Noted that for the macroeconomics situation news, we need to relate the content of the news to its impact on the stock market to rate the positivity. Each sector should have a final score for each stock.

SNP Score:

By inputting several indices that reflect the quantitative and qualitative situation of the stock, we could get a score out of 10 for each stock and choose stocks with the top scores. We need to find the weight of each input by referencing past three years historical data to determine their relative importance and hence have a weighting. Then, we need to input the current data into the program to get a score for each stock and select stocks with the top scores. The program should allow us to adjust the input index.

Sample code from the India team:

```

sector = input("Enter the sector the company operates in: ")
company = input("Enter the company: ")
pe_ratio = float(input("Enter Price-to-Earnings Ratio: "))
pb_ratio = float(input("Enter Price-to-Book Ratio: "))
eps_growth = float(input("Enter Earnings Per Share (EPS) Growth Rate: "))
revenue_growth = float(input("Enter Revenue Growth Rate: "))
roe = float(input("Enter Return on Equity (ROE): "))
debt_to_equity = float(input("Enter Debt-to-Equity Ratio: "))
fcf_yield = float(input("Enter Free Cash Flow Yield: "))
ps_ratio = float(input("Enter Price-to-sales: "))
piotroski_fscore = float(input("Enter Piotroski F-score: "))
alpha = float(input("Enter Alpha: "))
beta = float(input("Enter Beta: "))
sharpe = float(input("Enter Sharpe Ratio: "))
treynor = float(input("Enter Treynor Ratio: "))
benish_m_score = float(input("Enter Benish M Score: "))
std_dev = float(input("Enter Standard Deviation: "))
current_ratio = float(input("Enter Current Ratio: "))
quick_ratio = float(input("Enter Quick Ratio: "))

```

2. Diversity of the Portfolio

First, we need to build a covariance matrix for the portfolio using the stocks we selected in the second SNP score step to assess their correlations with each other.

I am not very sure about the best way to calculate the covariance matrix, but I found one way to do so and I will write it here. If there is any better way to build the matrix, feel free to change the method.

The method I found is:

Collect asset return data, and select historical asset returns for the same time period (daily, weekly, or monthly). Ensure that the data length is consistent (e.g., monthly data for all assets for the last three years).

Calculate the average return rate for each stock using the formula:

$$\mu_i = \frac{1}{T} \sum_{t=1}^T r_{i,t}$$

This can be done by using Python, and the sample code is:

```
import numpy as np
returns = np.array([[0.012, -0.005, 0.021], [-0.008, 0.013, 0.005], ...]) # 输入数据
mean_returns = np.mean(returns, axis=0)
```

Then, we can calculate the covariance matrix by using the

$$\Sigma_{ij} = \frac{1}{T-1} \sum_{t=1}^T (r_{i,t} - \mu_i)(r_{j,t} - \mu_j)$$

formula:

This can also be done by Python, and the sample code is:

```
cov_matrix = np.cov(returns, rowvar=False, ddof=1) # rowvar=False表示每列为一个变量
```

The values on the diagonal line of the matrix are variances, and the values in other positions are covariances. We need a graph illustration for our covariance matrix.

There is a graph help you to understand the naming rule and which value represent what in the covariance matrix:

对于一个包含 n 种资产的协方差矩阵 Σ , 其元素命名如下:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_n^2 \end{bmatrix}$$

- 对角线元素 (红色):

σ_i^2 表示第 i 种资产的方差 (自身波动率平方), 例如:

- σ_1^2 : 资产1的方差。
- σ_2^2 : 资产2的方差。

- 非对角线元素:

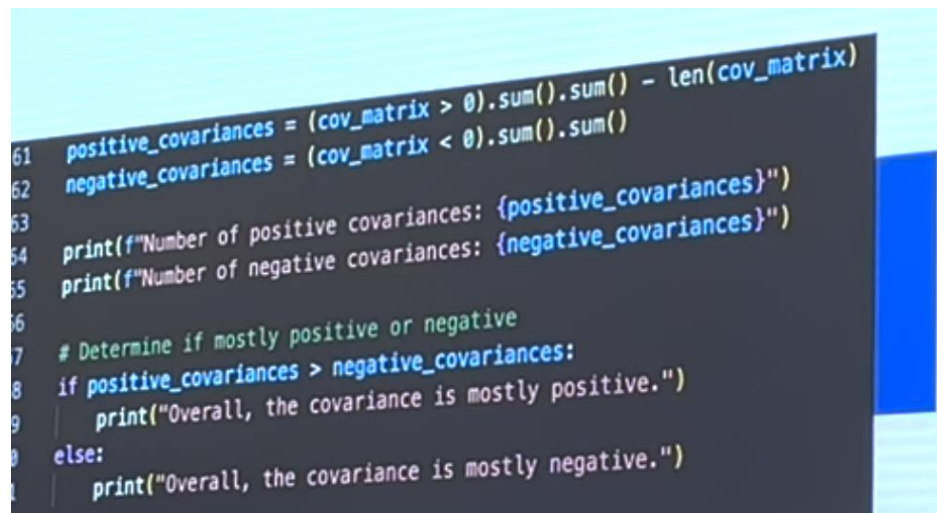
σ_{ij} ($i \neq j$) 表示资产 i 和资产 j 的协方差, 例如:

- σ_{12} : 资产1与资产2的协方差。
- σ_{23} : 资产2与资产3的协方差。

By using the covariance matrix, we can see the correlation between each two stocks in the portfolio, and a negative covariance value means that two stocks are not highly

related and are less likely to be impacted by the same change. In the matrix, the values in the diagonal line are variance, which show the fluctuation risk of the stock itself, and the covariance values are values not on the diagonal line. They show the linear relationship between two stocks and hence can show whether a change in one of them will also impact the other one. If the covariance is positive, those two assets move in the same direction (such as stocks and stock market indices). If it is negative, the two assets move in the opposite direction. If it is 0, then means there is no linear relationship but maybe a non-linear relationship. By using Python programming, we can use the positivity of all those covariance values to determine the diversity of the overall portfolio. Here, we cannot only consider the number of positive values and negative values, but we also need to consider the exact value itself to reflect their level of negativity or positivity.

Sample code from India group (but they only count the number of positive and negative covariance value):



```
61 positive_covariances = (cov_matrix > 0).sum().sum() - len(cov_matrix)
62 negative_covariances = (cov_matrix < 0).sum().sum()
63
64 print(f"Number of positive covariances: {positive_covariances}")
65 print(f"Number of negative covariances: {negative_covariances}")
66
67 # Determine if mostly positive or negative
68 if positive_covariances > negative_covariances:
69     print("Overall, the covariance is mostly positive.")
70 else:
71     print("Overall, the covariance is mostly negative.")
```

3. Efficient Frontier:

First, we need to use Python to randomly generate different combinations of stocks in different weightings for our portfolio.

We can calculate the portfolio return under different combinations of stocks:

$$\mu_p = \mathbf{w}^T \boldsymbol{\mu}$$

- \mathbf{w} : 资产权重向量 (如 $[0.3, 0.7]$ 表示30%资产A + 70%资产B)
- $\boldsymbol{\mu}$: 资产预期收益向量 (如 $[0.08, 0.12]$)

We can also calculate the risk of the portfolio under different stock combinations using the formula:

$$\sigma_p^2 = \mathbf{w}^T \Sigma \mathbf{w}$$

- Σ : 资产收益的协方差矩阵（对角线为方差，非对角线为协方差）

This could be calculated by using Jupiter notebook, Mathematica, or Python. Here is a sample python code:

```
import numpy as np

# 定义权重和协方差矩阵
w = np.array([0.6, 0.4])
Sigma = np.array([[0.04, -0.02], [-0.02, 0.09]])

# 计算组合方差
portfolio_variance = w.T @ Sigma @ w
print(f"组合方差: {portfolio_variance:.4f}") # 输出: 0.0192
```

Then we need to draw a graph with data points representing different combinations' risk and revenue on it. We can use Python to do so, and the sample code from India group is:

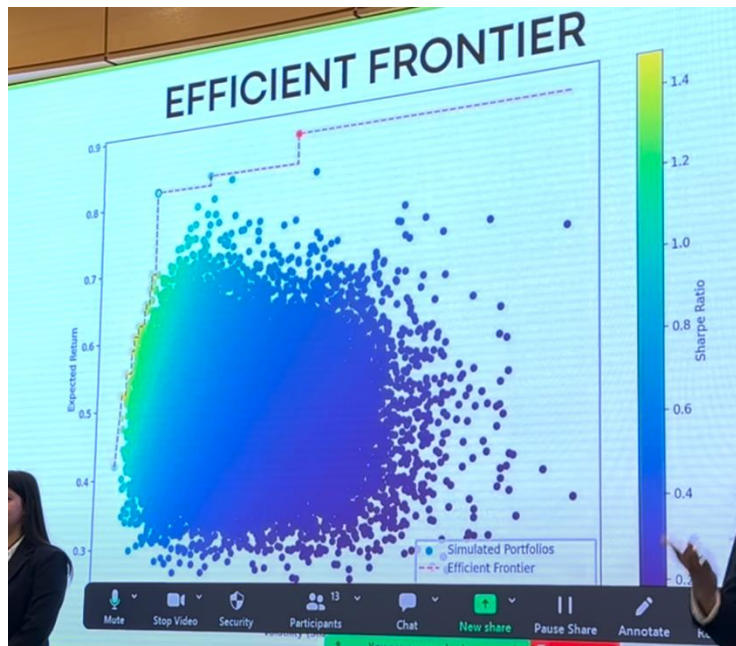
```
import matplotlib.pyplot as plt

# 绘制所有随机组合
plt.scatter(volatilities, returns, c=returns/volatilities, alpha=0.3)
plt.colorbar(label='Sharpe Ratio')

# 标记有效前沿
efficient_returns = returns[efficient_indices]
efficient_volatilities = volatilities[efficient_indices]
plt.plot(efficient_volatilities, efficient_returns, 'r-', lw=2, label='Efficient Frontier')

plt.xlabel('Volatility (σ)')
plt.ylabel('Expected Return (μ)')
plt.legend()
plt.show()
```

The graph should look like:



Then we just need to find the most left upper side points, they will be the efficient frontiers. Based on different consumer needs, we can then choose based on risk level or return level. For example, the case for the India group is that their consumer wants high returns and does not mind relatively high risk, so they choose the red point with the highest return among all efficient frontiers. Noted that the expected return is plotted on the y-axis, while risk is plotted on the x-axis.

4. Monte Carlo Simulation and Time Series

We want to use time series methods to forecast the portfolio return in the future five years and use the Monte Carlo Simulation to predict the portfolio return in the future fifteen years.

For the time series method, we use the past 1 year's historical price data.

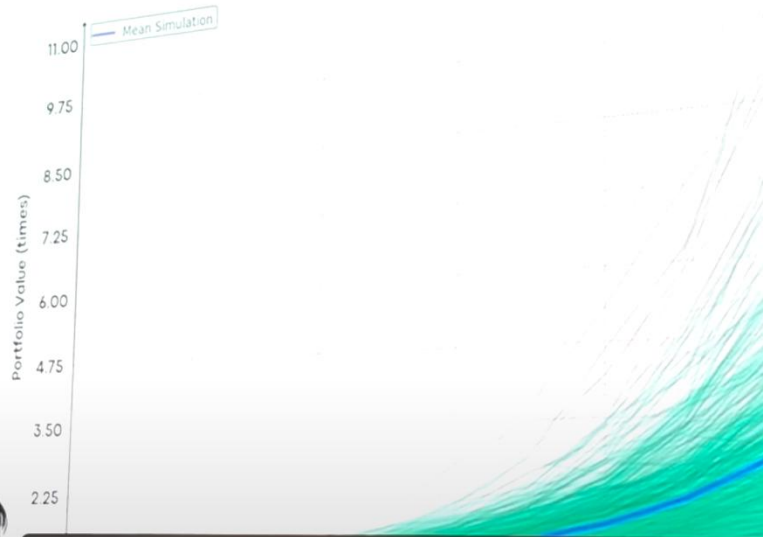
In the Monte Carlo Simulation, we need to set a probability distribution for uncertain factors (we do not know what factors yet, but they will act as input in the simulation).

Then, use a Python program to generate random values in the distribution to simulate different scenarios to simulate the portfolio's future performance. Then, again, use Python to draw the graph of hundreds of simulations and find the mean of the future performance.

The sample graph from the India group (They put portfolio value in the y-axis, and year number in the x-axis):

MONTE CARLO SIMULATION

Monte Carlo Simulation of Investment Over 15 Years



Mute Stop Video Security Participants 12 Chat New share Pause Share Annotate Remote control