Volatility Voyage

Assignment 4

Portfolio Analysis Using Indian Stock Market Data
+ Introductory Volatility forecasting

1.Objective:

The aim of this assignment is to build and evaluate a portfolio using real historical data from Indian stock markets. You will compute expected returns, standard deviations, portfolio variance, and Sharpe ratios from scratch, and visualize the Efficient Frontier to identify optimal portfolios.

Instructions:

1. Select Indian Stocks

Choose **any 3 actively traded Indian companies** from different sectors. For example:

- Reliance Industries (RELIANCE.NS)
- Infosys (INFY.NS)
- ICICI Bank (ICICIBANK.NS)

Use a **time period from January 2020 to May 2023** and retrieve monthly data.

2. Download Historical Data

Use Yahoo Finance or NSE India to download **adjusted closing prices**.

You may use the following sample Python code using the yfinance package:

```
import yfinance as yf
import pandas as pd
```

```
tickers = ['RELIANCE.NS', 'INFY.NS', 'ICICIBANK.NS']
data = yf.download(tickers, start='2020-01-01', end='2025-05-31', interval='1mo')['Adj Close']
returns = data.pct_change().dropna()
```

3. Compute Key Metrics

For each stock, calculate the following based on monthly returns:

- **Expected Return** (mean)
- **Standard Deviation** (volatility)
- **Covariance Matrix** (for portfolio risk evaluation)

Summarize these in a table.

4. Create Portfolios

Form at least **five portfolios** by changing weight combinations of the three stocks (weights should sum to 1). For each portfolio, compute:

- **Expected Portfolio Return**
- **Portfolio Standard Deviation**
- **Sharpe Ratio**

Assume a **monthly risk-free rate of 0.5%**

5. Plot the Efficient Frontier

- Use your calculated portfolio returns and standard deviations to plot the **Efficient Frontier** (Expected Return vs Risk).

- Identify:
    - **Minimum Variance Portfolio**
    - **Tangency Portfolio** (highest Sharpe Ratio) and the **Capital Market Line** (the line joining risk-free asset and tangency portfolio)

6. Report and Submission

Submit the following:

- A **Jupyter Notebook or Excel file** with all calculations and visualizations.
- A **short report** answering:
    - What effect did diversification have?
    - How did changes in weights affect risk and return?
    - Which portfolio would you recommend and why?

---

2. Now we move on to a slightly different approach to pick our stocks. What defines how user-friendly our model is? It is just about us incorporating the views of the investor too.

I might have the capacity to take more risk than a daily wage earner (not exactly, considering my current lifestyle :')) but I am surely not in a state right now to be able to take as much risk as someone like Mr Ambani himself. Should Mr Ambani and I, invest in the same set of stocks then? Surely not. What determines the kind of stock we invest in? VOLATILITY.

Someone more risk averse will invest in stocks that are not very volatile. We will use this very idea in this particular question.

Take the entire universe of stocks under NIFTY 50.

Calculate the volatility of all of them from 1st Jan 2022 to 1st September 2024.

Use the above data to forecast volatility of all the 50 stocks. (Relax if you can't retrieve data for all 50....afair, data for 48 stocks is accessible quite easily through yfinance).

How to forecast you ask? Here are three ways that I am giving you. These are the most basic methods for time series forecasting.

Firstly, calculate the rolling volatility, you will need this in all 3 methods.

```
data['returns'] = data['Close'].pct_change()

data['rolling_vol'] = data['returns'].rolling(window=20).std()
```

Methods:

1. Rolling Mean Forecast

This is the most straightforward method. It uses the average of the past N periods' volatilities to forecast the next period's volatility. It assumes that future volatility will be similar to the recent average. This method is stable and easy to compute but does not adapt quickly to sudden changes in market conditions.

```
data['vol_forecast_rm'] = data['rolling_vol'].rolling(window=5).mean()
```

(Adjust the window size as per what you think is necessary,

2. Exponentially Weighted Moving Average (EWMA)

EWMA gives more weight to recent observations and less weight to older data. This allows the model to respond more quickly to changes in volatility. A smoothing parameter (lambda) controls how much weight is given to recent data. Lower lambda values give more weight to recent observations. (Note, how this lamba_ is different from lambda function)

```
lambda_= 0.94

data['vol_squared'] = data['returns']**2

data['ewma_vol'] = data['vol_squared'].ewm(alpha=1 - lambda_).mean() ** 0.5
```

3. AR(1) Model on Volatility

This method treats volatility as a time series and fits an autoregressive model of order 1 (AR(1)) to it. It assumes that today's volatility can be explained by yesterday's volatility and some noise. This approach works well when volatility is autocorrelated.

$$\sigma(t)=\alpha+\beta \cdot \sigma(t-1)+\epsilon(t)$$

$\sigma(t)$ represents the forecasted volatility at time tt. In financial terms, this is the estimate of how much the price of an asset is expected to fluctuate in the current time period, based on past behavior.

α is a constant that captures the long-term average or base level of volatility. It ensures that the model doesn't collapse to zero and helps maintain a minimum level of expected volatility even when recent fluctuations are low.

β is the autoregressive coefficient that measures how strongly the previous period's volatility ($\sigma(t-1)$) influences the current period's volatility. A higher value of β implies that past volatility has a stronger impact, making the volatility more persistent over time.

$\epsilon(t)$ is the random shock or error term for the current time period. It captures new information or events that couldn't be predicted from past volatility, such as economic announcements or unexpected news

```
from statsmodels.tsa.ar_model import AutoReg

model = AutoReg(data['rolling_vol'].dropna(), lags=1).fit()

forecast = model.predict(start=len(data), end=len(data))
```

The given code uses an AutoRegressive (AR) model of order 1 to forecast volatility, specifically the rolling_vol series. By setting lags=1, the model assumes that today's volatility can be predicted using just yesterday's volatility. When we call .fit(), the model estimates parameters - an intercept (α) and a lag coefficient (β) - that best explain the observed relationship between past and current volatility. The model.predict(...) function then uses these estimated values to project the next time step's volatility. This method works well for volatility because volatility in financial markets tends to be **persistent** - meaning if the market is volatile today, it is likely to stay volatile in the near future. This persistence makes volatility predictable to some extent, especially over short horizons. By modeling this pattern through a simple AR(1) relationship, we can generate basic but often effective short-term forecasts of volatility.

Now match the results for your trained data and the forecasted volatility, pick the volatility forecasting method that gives you best outcome(judge from RMSE).

Once you've forecasted the volatility of all NIFTY 50 stocks using the three methods mentioned above, you now need to **incorporate the investor's view on risk**.

Let the investor provide a number between **0 and 1** that represents their **risk averseness**:

- A value close to **0** means the investor is highly risk-averse and prefers low-volatility stocks.
- A value closer to **1** means the investor is risk-seeking and is comfortable with more volatile stocks.

Step 1: Normalize Forecasted Volatility

Whichever forecasting method you use (Rolling Mean, EWMA, AR(1)), ensure that you **normalize the forecasted volatilities** between 0 and 1. You can do this using Min-Max normalization(something we discussed in our last class):

normalized_vol = (vol_forecast - vol_forecast.min()) / (vol_forecast.max() - vol_forecast.min())

Step 2: Match Stocks with Investor Profile

Calculate the **distance** between each stock's normalized forecasted volatility and the **investor's risk averseness** score:

distance = abs(normalized_vol - risk_aversion_score)

Select the **3 stocks with the minimum distance** — these are the stocks that align best with the investor's risk profile.

Step 3: Portfolio Construction

Assign **equal weights** to the selected 3 stocks initially, or if you're confident, assign weights **inversely proportional to their volatility** (lower volatility stocks get higher weight).

Use an **initial investment amount (₹10,000,00)** and calculate how much to allocate to each stock based on the weight.

Step 4: Backtest the Portfolio (from 1st Jan 2022 to 1st Jan 2025)

- Generate signals for each of the 3 selected stocks (you may use MACD+ATR strategy or any basic crossover strategy).
- Apply the signals to compute returns from each stock.
- Aggregate all three into a single **portfolio return stream** using the assigned weights.

Compute and report all metrics that you have included in your backtester.

Submit 2 separate ipynb files for the two different questions.
