Data Science and Advanced Programming 2025

Estimating future volatility for personal Portfolio

Final Project Report

Loris Bertschi
loris.bertschi.1@unil.ch
Student ID: 20556429

December 26, 2025

This project aims to help investors who manage investments and portfolios across different platforms, making it difficult to observe real returns and estimate risk. The goal is to develop a Python-based system that allows the user to centralize holdings, create a complete portfolio, compute key performance indicators, and ultimately provide short-term risk forecasts through machine-learning techniques.

The system allows the user to input holdings manually and automatically fetches historical market prices using the external financial data source Yahoo Finance. Portfolio performance metrics, such as returns, volatility, and drawdown, are calculated from these data to provide combined KPIs to the user. To further analyze the data, the system provides a forecast of the volatility of the whole portfolio based on engineered features derived from historical returns and rolling volatility measures. Three different models are evaluated: *Naïve baseline, Linear Regression, and Random Forest.*

The performance of each model is assessed using standard regression indicators: Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). Empirical results of the system show that Random Forest consistently outperforms the other models, a result that makes sense in financial time series, as non-linear modeling is essential. The main contribution of this project is the integration of portfolio analysis coupled with machine-learning-based risk prediction.

Keywords: data science, Python, machine learning, data science, portfolio, volatility, return

Contents

1 Introduction

Predicting the future is the ultimate objective for any investor looking for consistent positive returns and risk management. However, it remains the one capability that no investor truly possesses. Although trading and investment decisions are heavily based on historical data, past performance almost never guaranties future outcomes with certainty.

As a personal investor, one recurring challenge is understanding the true level of risk associated with building a “safe” or near-risk-free portfolio. Determining a reliable estimate of the risk I may face over the coming weeks or months is often complex, unclear, and difficult to approximate. This difficulty motivates the development of tools that bring structure, transparency, and predictability to portfolio risk analysis.

The problem addressed in this data science project is the lack of an accessible and systematic tool that allows individual investors to create a portfolio, track its past and current performance, and estimate its short-term risk using reliable machine learning models. More specifically, the project aims to answer:

How can we design a user-friendly system that imports or collects portfolio holdings, fetches current and historical market prices, and computes key metrics—such as returns, volatility, and value adjusted to estimated risk to help investors better understand and anticipate the risks they face in the next few days?

The objective of this project is to develop a practical tool that gives non-professional individual investors the ability to better understand the structure, performance, and short-term risk of their portfolios and holdings. The project aims to create a system capable of collecting portfolio holdings either through manual input or CSV import and transforming these data into a clean, structured dataset. Then it gathers, processes, and organizes historical and current market data so that meaningful statistical analyzes can be performed.

The main goal is then to compute the essential metrics required to evaluate a portfolio, such as daily returns, aggregated portfolio performance, volatility, and finally, deliver an estimated short-term risk indicators based on expectation provided by three *Machine Learning* models. The project further intends to present these results in a clear and interpretable way, allowing users to gain a transparent view of their exposure without requiring advanced financial expertise.

This report is organized to clearly present the development and outcomes of the project. First, it describes the methodology, including the characteristics of the data used, the analytical approach adopted, and the way these elements were implemented within the system. The following section presents the results, detailing the experimental setup, performance evaluation, and visualizations that support the analysis. The next discussion interprets the findings, highlighting what worked well, what proved challenging, and how the outcomes compare with initial expectations. Finally, the conclusion summarizes the main results and suggest potential future directions for extending or improving the project.

2 Literature Review / Related Work

Although financial markets have been studied since their inception, and many theories have been debated or disproven over time, one widely accepted conclusion remains: predicting market movements is extremely difficult. This challenge is well documented in the academic literature, particularly through the Efficient Market Hypothesis (EMH). Formalized by Fama (1970), the

EMH states that any patterns found in historical returns cannot be reliably used to predict future price movements. According to this hypothesis, market prices already reflect all publicly available information, making reliable prediction based on past data extremely challenging.

Despite this limitation, modern financial analysis relies heavily on historical returns to estimate portfolio risk. The foundations of such methods stem from Modern Portfolio Theory, introduced by Markowitz (1952), which defines risk through variance and covariance and highlights the importance of diversification, being the single most important rule in investing. Beyond the EMH, a large amount of work in financial econometrics has examined the statistical behavior of asset returns and volatility. Tsay (2010) highlights empirical properties of financial time series that complicate predictive modeling. These characteristics imply that financial returns almost never follow simple or stable statistical patterns, making long-term forecasting unreliable. However, Tsay also shows that short-term risk and uncertainty can be quantified by analyzing the historical distribution and volatility of returns, thus making this project relevant.

This perspective aligns closely with the goals of the present project. Rather than attempting to forecast future prices, the focus is on estimating the range of possible short-term fluctuations a portfolio may experience. By analyzing historical returns and computing measures such as volatility, the project provides investors with a practical approximation of the uncertainty of their holdings. While it does not overcome the fundamental unpredictability highlighted by Fama and Tsay, it offers a realistic and informative way to assess short-term risk based on empirical data.

3 Methodology

3.1 Data Description

The project is based on two distinct datasets: the portfolio information provided by the user, and the historical market data retrieved for each asset in the portfolio. The first dataset, the holding of the user is collected directly at the launch of the code, where the user inputs their holdings manually by entering the asset tickers, the quantities held, and the acquisition cost. This dataset is typically small, often fewer than a dozen entries, but it is essential for determining portfolio weights and computing performance measures.

The second dataset consists of historical price series collected through the yfinance API. For each asset in the user's portfolio, the system downloads daily adjusted closing prices over a predefined historical window. Once downloaded, the individual price series are merged into a single time-indexed matrix, stored in the data/ folder of the code, in which each column represents an asset and each row corresponds to a trading day. This dataset forms the basis for all following calculations, including returns, drawdown, and volatility for different time window. Unlike most data-science projects where a fixed dataset is collected once and reused every time, this project relies on a dynamic data-collection process. Each time the program is executed, the system collects the most recent market data available, meaning that the underlying dataset can change even when the user provides the same portfolio input.

Although the yfinance data source is widely used and generally reliable, certain data quality issues arise. Missing values may appear due to market holidays, asynchronous trading days, or incomplete historical coverage for specific assets. Additionally, newly listed securities or low-liquidity assets may exhibit short or noisy price series, which can reduce the stability of volatility estimates.

3.2 Approach

Several features and variables are then calculated from the raw price data. Daily returns are computed using percentage change, providing the fundamental data used for time-series modeling. After computing daily portfolio returns, they are used to create several rolling features that help the model understand recent market behavior. The features include realized volatility over the past 5, 10, and 20 days, as well as cumulative returns over the last 1, 5, and 10 days. These indicators capture short-term movements and the persistence of volatility in financial time series.

These features are used to predict the target variable, which is the next-week realized volatility, calculated as the standard deviation of returns over the following 5-day period. The dataset is constructed so that only past information is used to predict future volatility, ensuring that the model does not look ahead in time. Once the dataset is constructed, the next step consist of training and comparing three different forecasting models. The first, and most basic one is a *Naïve* model, it simply repeat the most recently observed volatility value to predict the next-week volatility of the portfolio. The second is a *Linear Regression* model that attempts to find and learn a linear relationship between the engineered features and future volatility. The third model, a *Random Forest* regressor, is an ensemble of decision trees trained on samples of the data with randomized features selections. This model is particularly well suited for noisy financial time series as it its non-linear architecture models complex interaction.

To ensure a fair evaluation of the forecasting models, the dataset is divided into two distinct subsets: a training set (80 of the data) and a testing set (20). Model performance is assessed using three standard regression metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). MAE is used as the primary evaluation criterion because it is easy to interpret and less sensitive to outliers, while MSE and RMSE provide complementary information by penalizing larger prediction errors more heavily. Together, these metrics offer a clear view of each model's accuracy and stability.

3.3 Implementation

The implementation of the system relies on an Python architecture that separate data handling/-downloading, portfolio analysis, feature engineering, model training, and user interaction into different files. This design help for clarity, reproducibility and easy debugging in case of problem.

First of all, the workflow begins in `main.py`, where the data (holdings) are collected and stored in a `pandas` DataFrame. Historical market prices are downloaded automatically using a dedicated function defined in `manual_input.py`. The function extract the ticker of each holding, retrieves data and prices through `yfinance`.

```python
def fetch_prices_from_holdings(holdings: pd.DataFrame, start="2020-01-01", end=None):
    """
    Fetch price data for the tickers in holdings.
    Always returns a clean DataFrame: index = dates, columns = tickers,
    values = adjusted close prices.
    """
    tickers = holdings["ticker"].astype(str).unique().tolist()
    data = yf.download(
        tickers,
        start=start,
        auto_adjust=True,
        progress=False
    )
    if isinstance(data.columns, pd.MultiIndex):
        data = data["Close"]
    return data.sort_index().ffill()
```

Listing 1: Fetching and cleaning historical prices.

Following this and some basic KPI computation, the code begins the volatility-forecasting component by transforming the portfolio returns into a structured dataset suitable for supervised learning. This part is handled by the `build_vol_dataset()` function in `risk_models.py`. This function computes:

- rolling realized volatility windows (5,10,20 days),
- rolling cumulative returns (1,5,10 days),
- the target variable: next-week realized volatility.

This create a `pandas` dataset `(X, y)` used to train the machine learning models.

```
def build_vol_dataset(port_ret: pd.Series, vol_windows=(5, 10, 20), ret_windows
    =(1, 5, 10),horizon: int = 5) -> Tuple[pd.DataFrame, pd.Series]:

    """
   Build feature matrix X and target y for next-week volatility prediction.

    Features (X):
      - past realized volatilities over windows in vol_windows
      - past cumulative returns over windows in ret_windows

    Target (y):
      - realized volatility over the *next* horizon days.
    """
    port_ret = port_ret.sort_index() #to make sure my return are well sorted
    feats = pd.DataFrame(index=port_ret.index) #create emptydataframe

    for w in vol_windows: #for each windows or days
        vol_w = realized_vol(port_ret, window=w, annualized=True) #compute
    volatility over the windows w
        feats[f"rv_{w}d"] = vol_w #then store it in the dataframe "feats"

    for w in ret_windows: #same as vol but does the average of the given window
     of days
        cumret = port_ret.rolling(w).sum()
        feats[f"ret_{w}d_sum"] = cumret

    future_vol = (
        port_ret.rolling(horizon).std(ddof=0).shift(-(horizon - 1)) * np.sqrt
    (252) #create target volatility based on the nest five days vol
    ) #it also place the target at the right date in the tbale so it's for the
    right expected future date
    y = future_vol.rename(f"rv_next_{horizon}d")

    data = feats.join(y, how="inner") #mix both features and target
    data = data.dropna(how="any") #eliminate the one with NaN as entry (
    basically the one where we can't compute based on the horizon)

    X = data.drop(columns=[y.name])
    y = data[y.name]

    return X, y
```

Listing 2: Dataset Creation for ML

The final stage of the project concerns the training, testing, and evaluation of the forecasting models. This component is implemented in the file `model_training.py`. It uses the `sklearn` library to train two of the three models, namely the Linear Regression and the RandomForest while the Naive model serves as a simple benchmark. A chronological train-test split is done and ensure no look-ahead bias. Each model is then evaluated using MAE, MSE and RMSE, with MAE used as the primary criterion due to its interpretability. MSE and RMSE provide additional insight into the magnitude of prediction errors by penalizing large deviations more heavily. Based on these measures, the system automatically identifies the best-performing model, which is then used to generate the final volatility forecast.

```python

def train_and_evaluate_all(
    X: pd.DataFrame,
    y: pd.Series,
    train_frac: float = 0.8,
) -> Tuple[pd.DataFrame, Dict[str, object]]:
    """
    Split data, train Naive / Linear / Random Forest, and return
    a metrics table plus the trained models.
    """
    X_train, X_test, y_train, y_test = time_series_split(X, y, train_frac=train_frac)


    y_pred_naive = naive_predict(X_test, col_name="rv_5d") #Naive model
    naive_metrics = regression_metrics(y_test, y_pred_naive)


    lin = train_linear_regression(X_train, y_train) #linear regression model
    y_pred_lin = pd.Series(lin.predict(X_test), index=y_test.index)
    lin_metrics = regression_metrics(y_test, y_pred_lin)


    rf = train_random_forest(X_train, y_train) #random forest model
    y_pred_rf = pd.Series(rf.predict(X_test), index=y_test.index)
    rf_metrics = regression_metrics(y_test, y_pred_rf)


    results = pd.DataFrame( #Build a single DataFrame with rows = models, columns = MAE/MSE/RMSE
        [naive_metrics, lin_metrics, rf_metrics],
        index=["Naive", "LinearReg", "RandomForest"],
    )

    models = {
        "naive": None,              # baseline has no fitted object
        "linear": lin,
        "random_forest": rf,
    }

    return results, models
```

Listing 3: Training & Comparing Models.

4 Results

This section presents the results obtained from the portfolio analysis and volatility forecasting pipeline. Since the system retrieves market data dynamically, numerical values may vary between runs. For the purposes of this report, the results were generated and analyzed on

23 December, based on Loris' portfolio, as described in the `README.md` of the repository. Although outcomes may differ depending on the date and portfolio provided, empirical results and test cases consistently demonstrate stable model performance, thereby supporting the following analysis.

4.1 Performance Evaluation

Model performance is evaluated using three regression metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). MAE is used as the primary evaluation criterion due to its interpretability, while MSE and RMSE provide additional insight into the magnitude of prediction errors. Table 1 reports the performance of the three forecasting models.

Table 1: Model Performance

Model	**MAE**	**MSE**	**RMSE**
Naive	0.2251	0.1067	0.3267
Regression	0.1891	0.0741	0.2723
Random Forest	**0.1800**	0.0746	0.2732

The Random Forest model consistently achieves the lowest MAE across multiple tests on different dates and portfolios with enough holdings, indicating better predictive performance compared to both the Linear Regression and the Naive models. This result highlights the importance of capturing non-linear relationships in volatility dynamics which are key characteristic of financial time series.

Although the second best model is the Linear Regression, it remains limited by its linear assumption, resulting in slightly lower predictive accuracy. This small gap may be explained by the limited size of the portfolio used and the presence of external market factors that influence the volatility. These outside events represent complex and difficult data to interpret and predict. The Naive model, which uses recent historical volatility as a forecast, provides a useful baseline but exhibits the highest prediction error as expected.

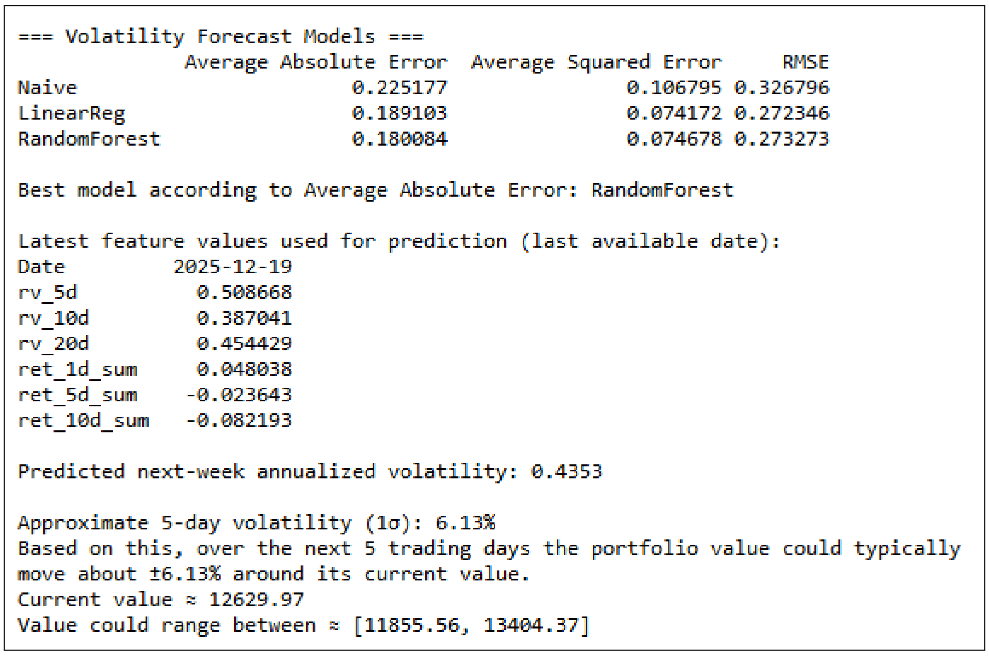
Based on these results, the Random Forest model is selected as the final predictor and used to generate the next-week volatility forecast for the user's portfolio.

4.2 Visualizations & Prediction

Once the right model is chosen, the system predicts the *next-week annualized volatility*, which represent the expected magnitude of portfolio return variability scaled to a one-year equivalent.To make this forecast relevant and usable for short-term analysis, the annualized volatility is then converted into a five-day measure using the square root of time rule.

$$\sigma_{5d} = \sigma_{\text{annual}} \times \sqrt{\frac{5}{252}} \tag{1}$$

This volatility is then interpreted as a one-standard deviation range and uses the current portfolio value to derive an approximate upper and lower bound for its possible value over the next five days. The following figure shows the results obtained in the test made previously.



```
=== Volatility Forecast Models ===
              Average Absolute Error  Average Squared Error     RMSE
Naive                       0.225177               0.106795 0.326796
LinearReg                   0.189103               0.074172 0.272346
RandomForest                0.180084               0.074678 0.273273

Best model according to Average Absolute Error: RandomForest

Latest feature values used for prediction (last available date):
Date         2025-12-19
rv_5d          0.508668
rv_10d         0.387041
rv_20d         0.454429
ret_1d_sum     0.048038
ret_5d_sum    -0.023643
ret_10d_sum   -0.082193

Predicted next-week annualized volatility: 0.4353

Approximate 5-day volatility (1σ): 6.13%
Based on this, over the next 5 trading days the portfolio value could typically
move about ±6.13% around its current value.
Current value ≈ 12629.97
Value could range between ≈ [11855.56, 13404.37]
```

Figure 1: Loris' Portfolio result as of 23.12.2025

Given the available data and the Random Forest model, the system predicts a next-week volatility of **0.4353**, expressed on an annualized basis. Using the square-root-of-time rule, this corresponds to an approximate **6.13%** volatility over the next five trading days, implying an expected portfolio value range between **$11,855.56** and **$13,404.37**.

5 Discussion

Overall, the model performs well in integrating portfolio analytics with short-term volatility prediction. The code allows a clear separation between dynamic data collection, model training and prediction. The *Random Forest* model consistently achieved the lowest prediction error in MAE, making it the best model to use between the three. This confirms the ability of this model to capture non-linear patterns in financial time series. Additionally, the translating process of the volatility into an interpretable range of potential value proved to be effective in making the result easily interpretable and intuitive to new users.

An interesting and unexpected result is the relatively strong performance of the *Linear Regression* model compared the *Random Forest*. While it remains the second best-performing, the gap between the two models is smaller than initially anticipated as financial markets usually never follows linear outcomes. This may be due to the limited size of the portfolio or the restricted time horizon for the data collected or it may suggest that a portion of short-term portfolio volatility can be explained by simple linear relationships in recent return and historical volatility.

The main challenge of the project lies in the dynamic nature of the data. Because portfolio can be differently built and prices are fetched at runtime, data update on a daily basis and results vary with each execution. To address this issue, a base portfolio was constructed and evaluated at a specific date, providing a clear visualization of representative data and results suitable for baseline interpretation. Additionally, missing values and heterogeneous trading calendars across assets required careful handling, making the datasets and Data Frames not always usable at first.

Although the system delivers acceptable results, its approach relies on several simplifying assumptions, thus implying crucial limitations. First, volatility scaling is based on the square-root-of-time rule (1), which assumes *iid* returns and does not capture volatility clustering during abnormally turbulent periods. Second, the model predicts volatility rather than returns, which provides information about risk but rather than future price movements. Finally, the feature set is limited to short-term historical indicators and does not incorporate macroeconomic variables, regime changes, or transaction costs. All of these limitations suggest that the predictions should interpreted as probabilistic estimates rather than precise forecasts.

6 Conclusion and Future Work

6.1 Summary

This project aimed to develop system for portfolio analysis and short-term risk forecasting, addressing the challenge faced by investor managing assets across different trading platforms. The application allows users to centralize their holdings, whether they are Equity, ETFs or even crypto, retrieve historical market data from Yahoo, and compute key portfolio performance indicators such as returns, volatility Sharpe ratio and drawdown.

Beyond the basic analysis, the main part of the project lies in the integrated *Machine Learning* framework to forecast short-term volatility. Using features derived from historical returns and realized volatility, three models are evaluated : *Naive, Linear Regression*, and *Random Forest.* Results Show that *Random Forest* consistently achieves the best predictive performance based on MAE. The final output shows volatility forecasts into an interpretable range of potential portfolio value, giving the user a practical understanding of risk and how it might affect his portfolio in the next trading days.

6.2 Future Directions

As the system is based on several simplified assumptions, some extensions and improved engineering could further improve the methodology and applicability. Future work could explore more advanced models for volatility forecasting which may better capture temporal dependencies and regime changes in financial time series. Moreover, additional feature engineering, including macroeconomic indicators or provisions for financial statement release, could also improve the predictive accuracy.

From a practical point of view, the application could be extended into a more graphical and UI enhanced dashboard, thus enabling easier interaction for new and non-technical users. Finally, real-world technical enhancements like transaction costs, re balancing strategies or periodic and automatic investments integration could make the system more suitable for practical investment decision and portfolio management.

References

[1] Ran Aroussi. *yfinance: Yahoo! Finance Market Data Downloader*. `https://github.com/ranaroussi/yfinance`. 2020.

[2] Eugene F. Fama. “Efficient Capital Markets: A Review of Theory and Empirical Work”. In: *The Journal of Finance* 25.2 (1970), pp. 383–417.

[3] Harry Markowitz. “Portfolio Selection”. In: *The Journal of Finance* 7.1 (1952), pp. 77–91.

[4] matplotlib Development Team. *matplotlib: Visualization with Python*. `https://matplotlib.org`. 2024.

[5] NumPy Developers. *NumPy*. `https://numpy.org`. 2024.

[6] OpenAI. *ChatGPT*. `https://chat.openai.com`. Used as an AI-assisted tool for code development, debugging, and documentation. 2024.

[7] Pandas Development Team. *pandas: Python Data Analysis Library*. `https://pandas.pydata.org`. 2024.

[8] Fabian Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[9] Ruey S. Tsay. *Analysis of Financial Time Series*. 3rd ed. Wiley, 2010.

[10] Yahoo Finance. *Yahoo Finance Market Data*. `https://finance.yahoo.com`. Accessed via the `yfinance` Python package. 2024.

A Additional Figures

Include supplementary figures or tables that support but aren't essential to the main narrative.

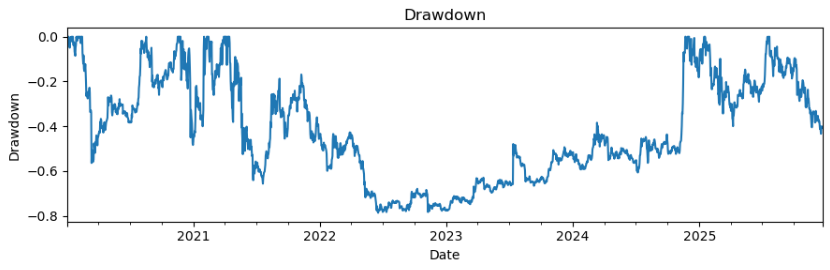


Figure 2: Drawdown example of Loris'Portfolio

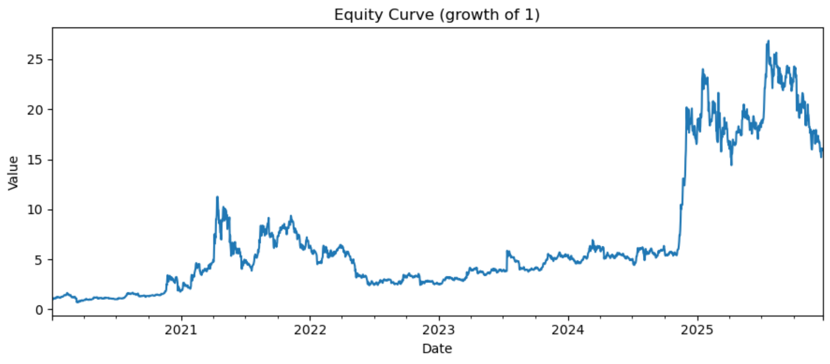


Figure 3: Equity Curve example

B Code Repository

GitHub Repository: `https://github.com/LorisBe/FutureVolatility_project_LB.git`

Information for installation and run provided in the README.md