Market Risk Project

Topic: Value at risk Calculation & PnL Backtesting Based on SP500

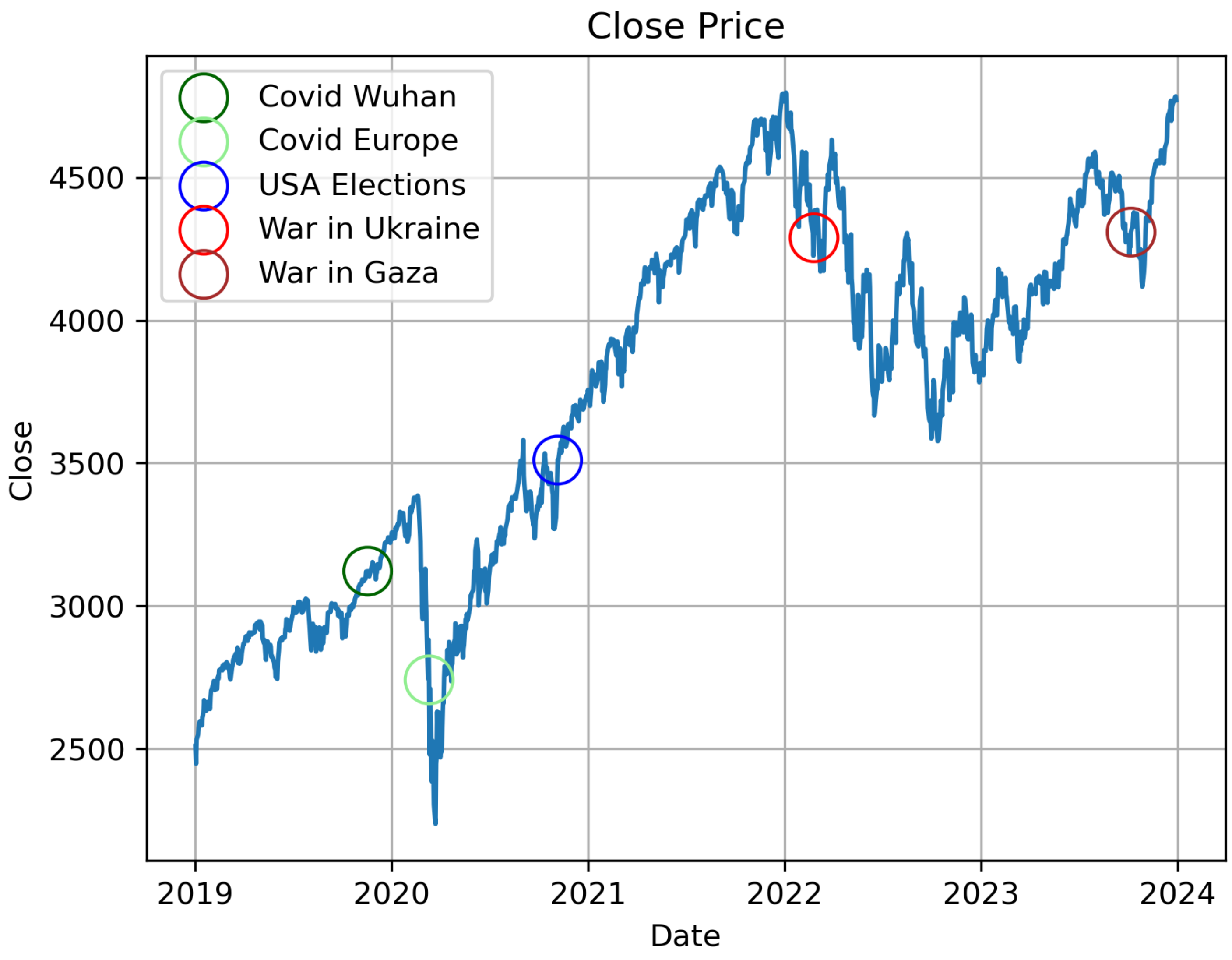
Name：Honglin Qian

Positive Research Ltd

Date: 2024 – 4 – 30

Introduction

This project aims to calculate value-at-risk measure and construct Pnl backtesting via SP500, we have downloaded the data from Yahoo Finance. We will use SP500 index as our main data source.



1 Statistical Test

In the first step we calculate some statistical parameter to check the statistic characteristic of **percentage return** for further usage.

1.1 Shapiro-Wilk Test for normality

The Shapiro-Wilk test is a statistical test used to determine whether a data sample comes from a normally distributed population. It's particularly useful for small to moderate-sized datasets (typically less than 50, but it can be used for larger datasets up to 2000).

- **Null Hypothesis (H0)**: The data is normally distributed.
- **Alternative Hypothesis (H1)**: The data is not normally distributed.

The mathematical formula for the Shapiro-Wilk test is defined as follows:

Given a sample dataset: $x_1, x_2, \ldots, x_n$, where n is the sample size.

1. Order the data: First, sort the sample data to obtain $x_1, x_2, \ldots, x_n$.

2. Calculate the test statistic W:

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

where:

- $\bar{x}\ is\ the\ mean\ of\ the\ sample\ data, i.e., (\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i).$
- $x_{(i)} represents\ the\ i - th\ sorted\ sample\ data.$
- *Coefficients a_i are calculated from the expected values of the order statistics of a standard normal distribution. Specifically, a_i are the weights derived to maximize the test statistic W. If W is close to 1, it suggests that the sample is from a normal distribution.*

Based on the calculated W value and the corresponding p-value, you determine whether the data follows a normal distribution. If the p-value is less than the chosen significance level (typically 0.05), you reject the null hypothesis, indicating that the data does not follow a normal distribution.

The Shapiro-Wilk test is typically performed using statistical software or programming languages like Python (using the *"scipy.stats.shapiro"* function). It is very effective for detecting deviations from normality, especially in small sample sizes.

Reasons: When calculating Value at Risk (VaR) in finance and statistical analysis, if the data (percentage return series) follows a normal distribution is a common assumption. This assumption simplifies the calculation of VaR and makes the model easier to use. However, in the real world, the returns of financial assets often do not perfectly follow a normal distribution. Therefore, it is important to perform a normality test (such as the Shapiro-Wilk test) before calculating VaR.

1.2 Ljung-Box Test for Autocorrelation

- **Null Hypothesis (H0)**: There is no significant autocorrelation in the time series data up to the specified number of lags (white noise).
- **Alternative Hypothesis (H1)**: There is significant autocorrelation in the time series data up to the specified number of lags.

The Ljung-Box test statistic Q is calculated using the following formula:

$$Q = \mathrm{n}(\mathrm{n}+2)\sum_{\mathrm{k}=1}^{\mathrm{h}}\frac{\widehat{\mathrm{r}_{\mathrm{k}}^{2}}}{\mathrm{n}-\mathrm{k}}$$

Where:

- n is the sample size of the time series.
- h is the number of lags being tested.
- $\widehat{r_k^2}$ is the sample autocorrelation at lag k.

Steps for the Ljung-Box Test

1. **Calculate Autocorrelation Coefficients**: For a given time series, compute the sample autocorrelation coefficients $\widehat{r_k^2}$ for different lags.
2. **Compute the Test Statistic** Q: Calculate the Ljung-Box test statistic Q using the formula.
3. **Check Statistical Significance**: Compare the calculated Q value against the critical value from the chi-square distribution with h degrees of freedom. If the Q value is greater than the critical value, or the corresponding p-value is less than the chosen significance level (such as 0.05), reject the null hypothesis, indicating that there is significant autocorrelation in the time series.

Reasons: VaR models, such as the variance-covariance method, historical simulation, or Monte Carlo simulation, often assume that the time series data (percentage return series) is independently and identically distributed (i.i.d.). However, financial time series data (e.g., stock returns, interest rate changes) may exhibit autocorrelation, meaning that current returns may be influenced by past returns.

1.3 Levene Test for homoskedasticity

The Levene test is a statistical test used to assess the equality of variances for a variable calculated for two or more groups. It is an important test in the context of analysis of variance (ANOVA) because one of the assumptions of ANOVA is that the variances of the populations from which different samples are drawn are equal (homogeneity of variance).

- **Null Hypothesis (H0):** The variances of the groups are equal (homogeneity of variance).
- **Alternative Hypothesis (H1):** At least one group has a variance different from the others (heterogeneity of variance).

The Levene test statistic W is calculated using the following formula:

$$W = \frac{(N-k)}{(k-1)} \frac{\sum_{i=1}^{k} N_i (Z_{i\cdot} - Z_{\cdot\cdot})^2}{\sum_{i=1}^{k} \sum_{j=1}^{N_i} (Z_{ij} - Z_{i\cdot})^2}$$

Steps to Conduct the Levene Test

1. **Calculate the Deviations**: For each group, calculate the deviations of each observation from the group median (or mean).
2. **Compute the Test Statistic** W: Use the formula provided to compute the test statistic W.
3. **Determine the p-value**: The test statistic W follows an F-distribution with $k-1$ and $N-k$ degrees of freedom. Compare the computed W value against the critical value from the F-distribution or use the p-value approach to determine statistical significance.
4. **Interpret the Results**: If the p-value is less than the chosen significance level (commonly 0.05), reject the null hypothesis, indicating that there is evidence of heterogeneity of variances among the groups.

Reasons: When calculating VaR, especially using parametric methods like the variance-covariance approach or other models that rely on statistical assumptions (e.g., linear regression models for risk factors), one common assumption is that the variances of returns (or other financial metrics) are constant over time (homoscedasticity). If the variance of returns changes over time (heteroscedasticity), it can affect the accuracy of the VaR estimation.

1.4 ADF Test for Stationarity

The Augmented Dickey-Fuller (ADF) test is a statistical test used to determine whether a time series is stationary or has a unit root, indicating non-stationarity. In time series analysis, stationarity is an important property because many statistical models, including ARIMA, require the data to be stationary to make valid inferences. A stationary time series has constant mean, variance, and autocorrelation structure over time.

- **Null Hypothesis (H0):** The time series has a unit root (i.e., it is non-stationary).
- **Alternative Hypothesis (H1):** The time series does not have a unit root (i.e., it is stationary).

1. Test Statistic: This is a value calculated from your data to evaluate whether to reject the null hypothesis (i.e., the time series has a unit root and is non-stationary).
2. Critical Values: These are threshold values at different confidence levels (e.g., 1%, 5%, 10%) used to determine whether the Test Statistic is significant.

Decision Rule:

- If the ADF Test Statistic is less than (more negative than) the Critical Values, you reject the null hypothesis, indicating the time series is **stationary.**
- If the ADF Test Statistic is greater than the Critical Values, you fail to reject the null hypothesis, indicating the time series may be **non-stationary.**

Example: Suppose you perform the ADF test and get the following results:
- ADF Test Statistic: -3.50
- Critical Values: 1%: -3.96, 5%: -3.41, 10%: -3.12

In this case:

- The ADF Test Statistic (-3.50) is less than the 10% Critical Value (-3.12), so you reject the null hypothesis at the 10% significance level, meaning the time series is stationary.
- The ADF Test Statistic (-3.50) is greater than the 5% Critical Value (-3.41), so you fail to reject the null hypothesis at the 5% significance level, indicating the time series may be non-stationary.
- The ADF Test Statistic (-3.50) is greater than the 1% Critical Value (-3.96), so you fail to reject the null hypothesis at the 1% significance level, indicating the time series may be non-stationary.

This means that at the 10% significance level, the time series is stationary, but at the stricter 5% and 1% significance levels, the time series may be non-stationary.

Reasons: VaR models, particularly those based on historical simulation or parametric methods like the variance-covariance approach, often assume that the data is stationary. Stationarity means that the statistical properties of the data (mean, variance, autocorrelation) are constant over time. If the time series data is non-stationary, it can lead to misleading risk estimates because non-stationary data might exhibit trends, changing variances, or other patterns that violate the assumptions of the model.

Below we present the table with p-values of tests which were described in the previous slide performed on our 4 indexes:

Test	**S&P 500**
Shapiro-Wilk	1.27e-30
Ljung-Box	1.96e-53
Levene	1.32e-27
ADF	5.81e-19

2 Value-at-risk

2.1 Estimator

Let x denote vector of simulated one-day P&L scenarios and n its length. By $\bar{x}$ we denote mean and by $\bar{\sigma}$ we denote standard estimator of standard deviation. For price with index *i,* we calculate *n* previous return rates. Then we create x by multiplying every return rate by the close price with *i*-th index.

2.2 Value at Risk

Value-at-Risk (VaR) is a statistical technique used to measure the potential risk of loss in the value of an investment portfolio over a defined period for a given confidence level. VaR helps financial institutions and investors understand and manage market risk. There are three main methods for calculating VaR: Historical Simulation, Variance-Covariance (also known as Parametric Method), and Monte Carlo Simulation.

Coding Explanation:

1. **Lookback Period**:
   - The lookback period (lookback) is a specified window of historical data used to calculate each VaR value. This window determines the amount of past data considered when computing the risk for each point in time.
2. **Calculation and Storage in V**:
   - In functions (Historical_Var, Var_Norm, VarWeighted), the array V is initialized with a size that starts from lookback to the length of "self.closePrices – 1". This means the VaR calculation begins **only after enough historical data has been accumulated** to fill the lookback window.
   - For example, if lookback = 250, the first VaR value in V corresponds to the period after 250 data points (days, weeks, etc., depending on the data frequency) have been used to calculate it.
3. **Indexing in V**:
   - The index "i" lookback is used when assigning values to the array V. This indexing indicates that the VaR calculation for a particular time "i" is stored at position "i – lookback" in the array V.
   - Therefore, the first element of V corresponds to the VaR calculated **after** the initial lookback period of data. This continues sequentially for each subsequent period.

Summary:

- The array V does not contain VaR values for the initial lookback period because there is not enough historical data to perform the calculation until after the lookback period is completed.
- Each value in V represents the VaR calculated for a period **starting after** the lookback window, providing a time series of risk measurements beginning from the first point where enough data is available to compute a reliable VaR.

2.2.1. Historical Simulation

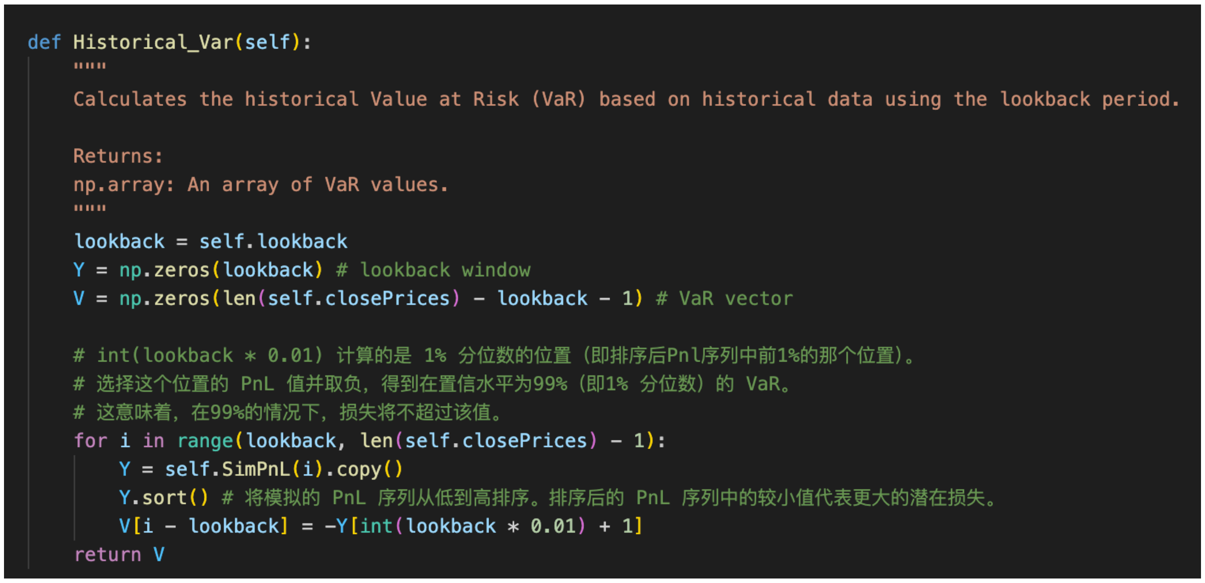
The Historical Simulation method uses historical market data to calculate VaR. It assumes that past market performance will continue to influence future outcomes and directly uses past return distributions to predict future risks.

Calculation Steps

1. **Collect Historical Data**: Gather historical price data for the asset or portfolio and calculate daily returns.
2. **Sort Returns**: Sort all returns from the smallest to the largest.
3. **Determine VaR Percentile**: Based on the desired confidence level (e.g., 95% or 99%), find the corresponding historical return percentile. For a 99% confidence level, locate the return that represents the bottom 1%.
4. **Calculate VaR**: VaR is the negative value of the corresponding percentile return.

Advantages and Disadvantages

- **Advantages**: Simple and intuitive, does not require assumptions about return distributions.
- **Disadvantages**: Relies heavily on historical data and may not account for structural changes or extreme market conditions.



```python
def Historical_Var(self):
    """
    Calculates the historical Value at Risk (VaR) based on historical data using the lookback period.

    Returns:
    np.array: An array of VaR values.
    """
    lookback = self.lookback
    Y = np.zeros(lookback) # lookback window
    V = np.zeros(len(self.closePrices) - lookback - 1) # VaR vector

    # int(lookback * 0.01) 计算的是 1% 分位数的位置（即排序后Pnl序列中前1%的那个位置)。
    # 选择这个位置的 PnL 值并取负，得到在置信水平为99%（即1% 分位数）的 VaR。
    # 这意味着，在99%的情况下，损失将不超过该值。
    for i in range(lookback, len(self.closePrices) - 1):
        Y = self.SimPnL(i).copy()
        Y.sort() # 将模拟的 PnL 序列从低到高排序。排序后的 PnL 序列中的较小值代表更大的潜在损失。
        V[i - lookback] = -Y[int(lookback * 0.01) + 1]
    return V
```

2.2.2. Variance-Covariance Method

The Variance-Covariance method assumes that returns are normally distributed and estimates VaR by calculating the mean and standard deviation of the asset or portfolio returns.

Calculation Steps

1. **Calculate Mean Return and Standard Deviation**: Compute the average return and standard deviation of the portfolio returns.
2. **Determine the z-Score for Confidence Level**: Find the z-score corresponding to the chosen confidence level (e.g., for 95% confidence level, the z-score is 1.65).
3. **Calculate VaR**: Use the formula:

$$\mathrm{VaR}_{0.01} = -(\mu + \sigma \cdot z_{0.01})$$

Advantages and Disadvantages

- **Advantages**: Quick and simple to calculate, suitable for linear risk factors.
- **Disadvantages**: Assumes normal distribution of returns, ignoring tail risks and non-linear risks.

```python
def VarNorm(self):
    """
    Calculates the Gaussian (Normal) Value at Risk (VaR) using the mean and standard deviation.

    Returns:
    np.array: An array of Gaussian VaR values.
    """
    lookback = self.lookback
    V = np.zeros(len(self.closePrices) - lookback - 1)
    for i in range(lookback, len(self.closePrices) - 1):

        # 是正态分布在 1% 置信水平下的分位数。由于正态分布的对称性，这个值为负，表示极端情况下的损失水平。
        V[i - lookback] = -(self.mu(i) + self.sigma(i) * sps.norm.ppf(0.01, loc=0, scale=1))
    return V
```

2.2.3. Exponentially Weighted VaR

Weights Calculation:

The weights are computed exponentially, giving more weight to more recent observations. The formula for the weights is:

$$w_i = \frac{(1-\lambda) \cdot \lambda^i}{1 - \lambda^{\mathrm{lookback}}}$$

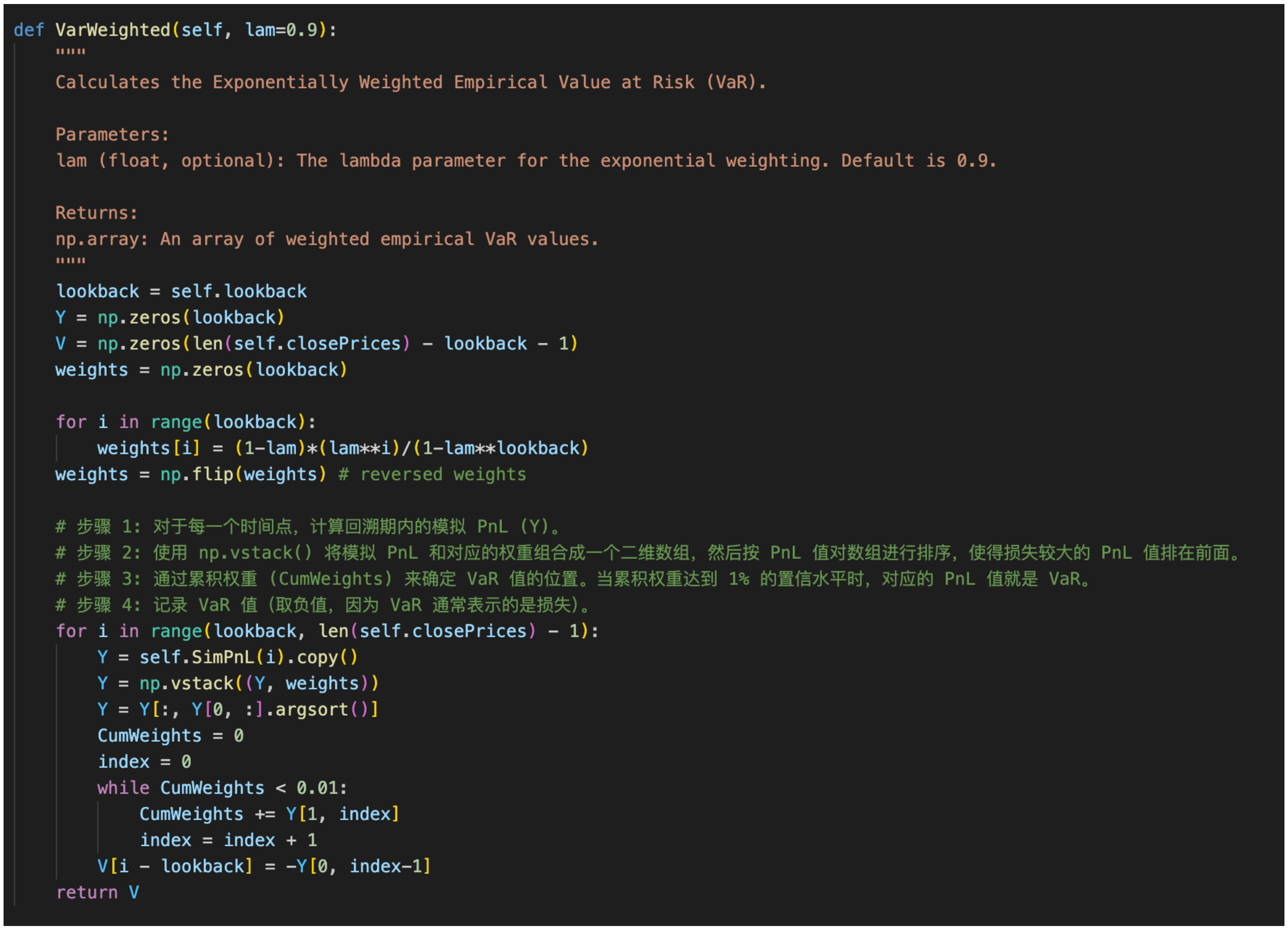
where:

- w_i is the weight for the i-th observation.
- λ (Denoted as lam in the code) is the decay factor, with a default value of 0.9.
- Lookback: is the number of past observations considered.

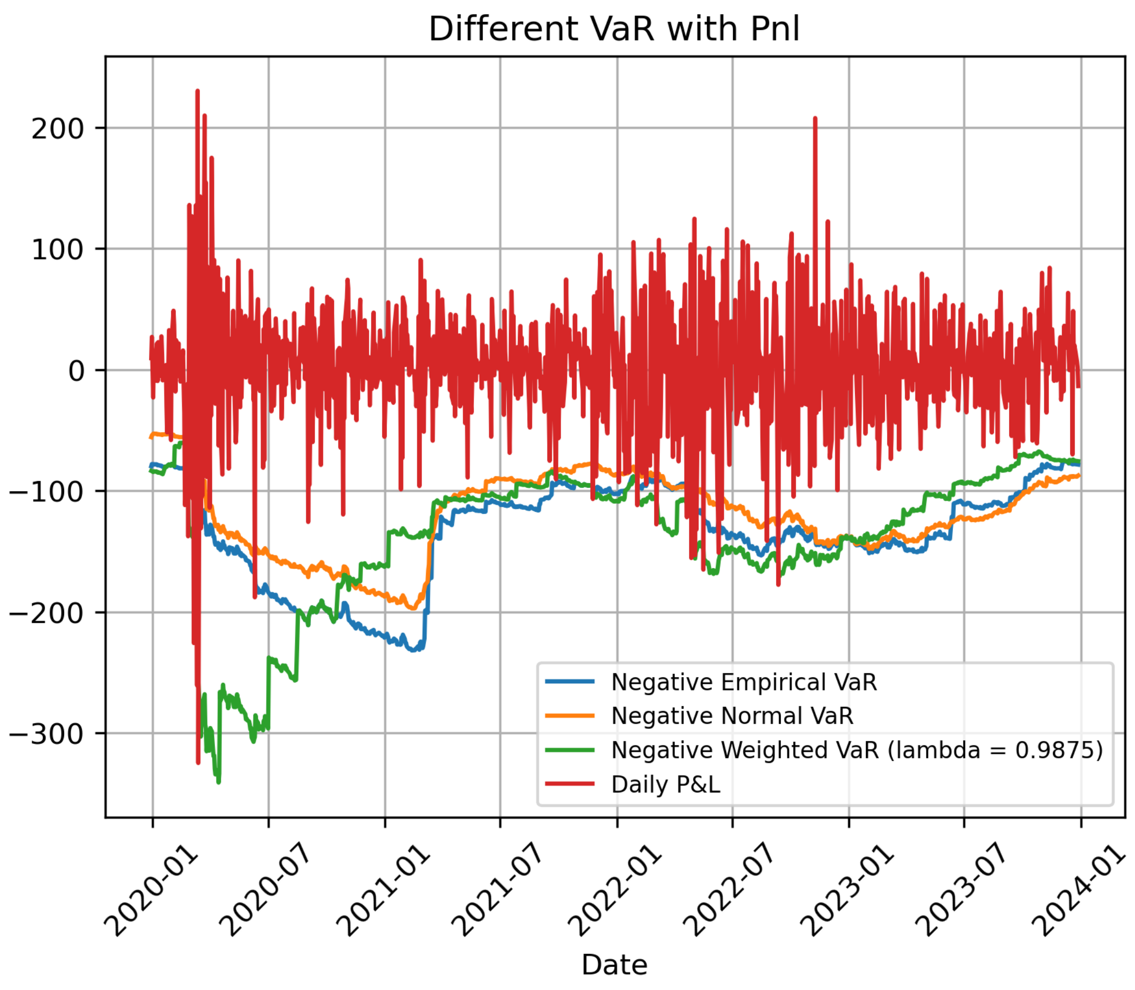
VaR Calculation:

The VaR is calculated by finding the point at which the cumulative weight reaches the 1% threshold. The code performs the following steps:

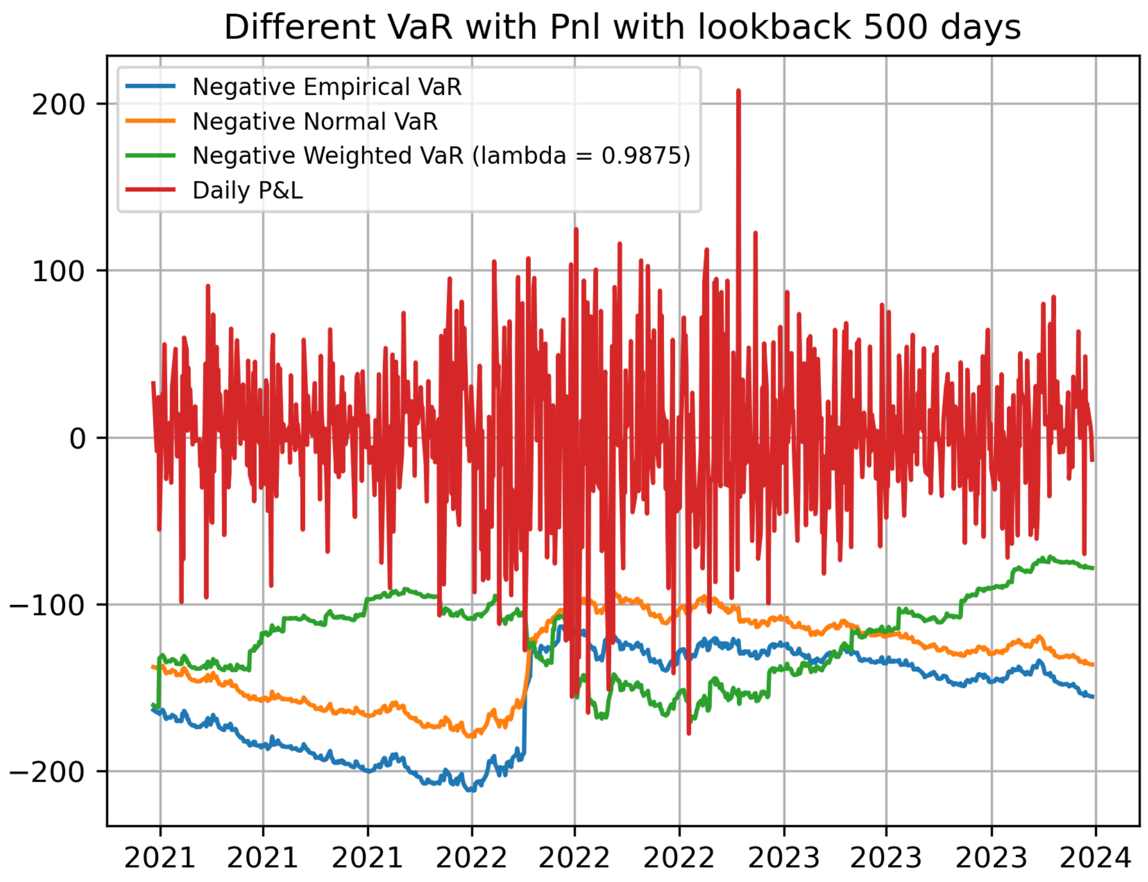
- **Simulate PnL (Y) for each time window**.
- **Sort PnL and weights**: Combine (“np.vstack”) and sort PnL values by their corresponding weights.
- **Compute cumulative weights (“CumWeights”)** and find the position where cumulative weight is just over 1%.
- **Determine VaR**: The corresponding PnL value at this cumulative weight threshold is the VaR.



```python
def VarWeighted(self, lam=0.9):
    """
    Calculates the Exponentially Weighted Empirical Value at Risk (VaR).

    Parameters:
    lam (float, optional): The lambda parameter for the exponential weighting. Default is 0.9.

    Returns:
    np.array: An array of weighted empirical VaR values.
    """
    lookback = self.lookback
    Y = np.zeros(lookback)
    V = np.zeros(len(self.closePrices) - lookback - 1)
    weights = np.zeros(lookback)

    for i in range(lookback):
        weights[i] = (1-lam)*(lam**i)/(1-lam**lookback)
    weights = np.flip(weights) # reversed weights

    # 步骤 1: 对于每一个时间点，计算回溯期内的模拟 PnL (Y)。
    # 步骤 2: 使用 np.vstack() 将模拟 PnL 和对应的权重组合成一个二维数组，然后按 PnL 值对数组进行排序，使得损失较大的 PnL 值排在前面。
    # 步骤 3: 通过累积权重 (CumWeights) 来确定 VaR 值的位置。当累积权重达到 1% 的置信水平时，对应的 PnL 值就是 VaR。
    # 步骤 4: 记录 VaR 值 (取负值，因为 VaR 通常表示的是损失)。
    for i in range(lookback, len(self.closePrices) - 1):
        Y = self.SimPnL(i).copy()
        Y = np.vstack((Y, weights))
        Y = Y[:, Y[0, :].argsort()]
        CumWeights = 0
        index = 0
        while CumWeights < 0.01:
            CumWeights += Y[1, index]
            index = index + 1
        V[i - lookback] = -Y[0, index-1]
    return V
```



Different VaR with Pnl
200
100
0
−100
−200
−300
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01
Date
Negative Empirical VaR
Negative Normal VaR
Negative Weighted VaR (lambda = 0.9875)
Daily P&L



Different VaR with Pnl with lookback 500 days
200
100
0
−100
−200
2021
2021
2021
2022
2022
2022
2023
2023
2023
2024
Negative Empirical VaR
Negative Normal VaR
Negative Weighted VaR (lambda = 0.9875)
Daily P&L

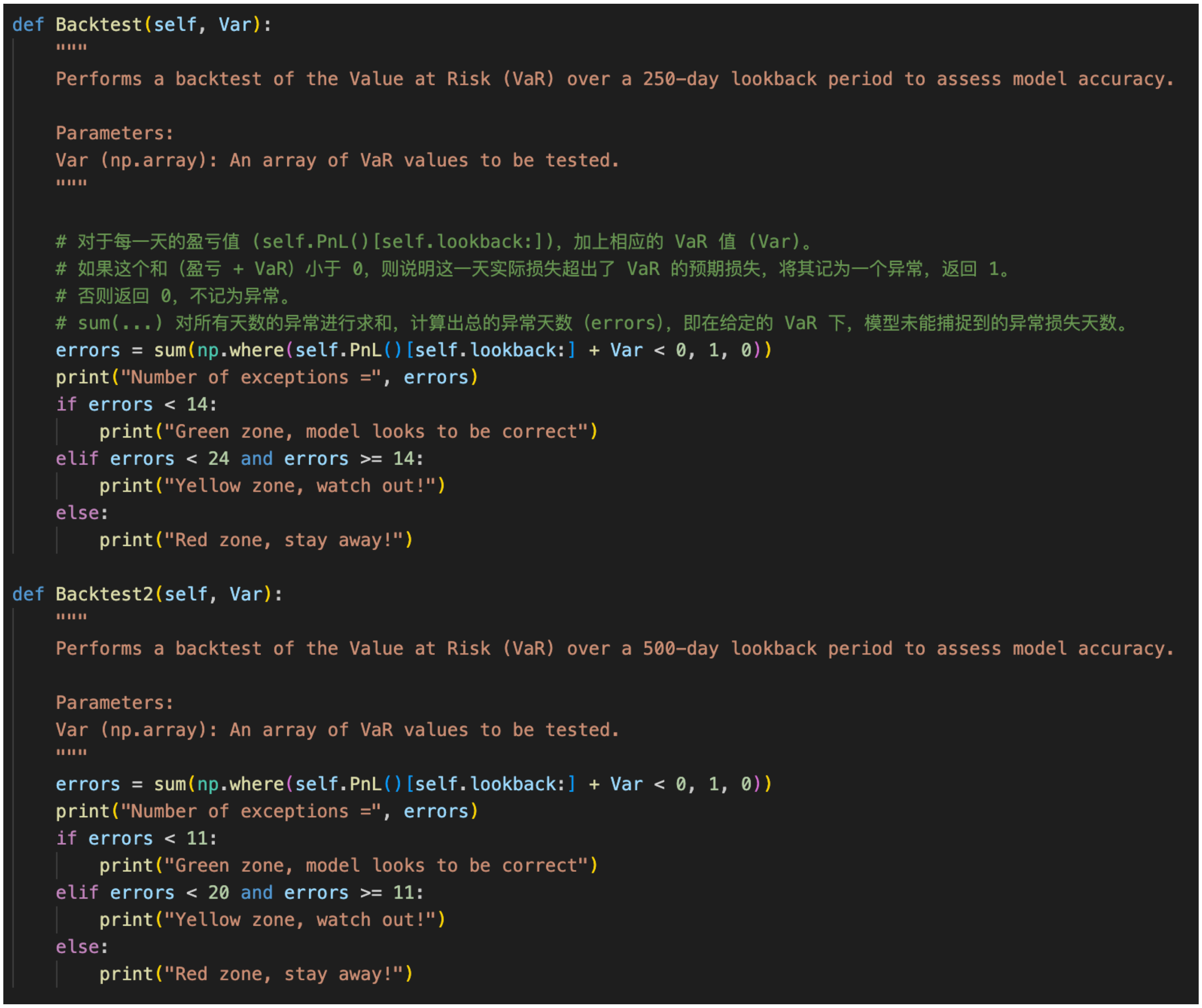
3 Backtesting

The "Backtest" and "Backtest2" functions in the provided code are used to assess the accuracy of a Value at Risk model over different lookback periods, specifically 250 days and 500 days.

Let's denote:

$$\text{errors} = \sum_{i=\text{lookback}}^{N} \mathbb{1}(\text{PnL}_i + \text{VaR}_i < 0)$$

- PnL_i is the as the profit and loss on day i.
- VaR_i is as the Value at Risk (VaR) on day i.
- N as the total number of days in the lookback period (either 250 or 500)



```python
def Backtest(self, Var):
    """
    Performs a backtest of the Value at Risk (VaR) over a 250-day lookback period to assess model accuracy.

    Parameters:
    Var (np.array): An array of VaR values to be tested.
    """

    # 对于每一天的盈亏值 (self.PnL()[self.lookback:]), 加上相应的 VaR 值 (Var)。
    # 如果这个和 (盈亏 + VaR) 小于 0, 则说明这一天实际损失超出了 VaR 的预期损失, 将其记为一个异常, 返回 1。
    # 否则返回 0, 不记为异常。
    # sum(...) 对所有天数的异常进行求和, 计算出总的异常天数 (errors), 即在给定的 VaR 下, 模型未能捕捉到的异常损失天数。
    errors = sum(np.where(self.PnL()[self.lookback:] + Var < 0, 1, 0))
    print("Number of exceptions =", errors)
    if errors < 14:
        print("Green zone, model looks to be correct")
    elif errors < 24 and errors >= 14:
        print("Yellow zone, watch out!")
    else:
        print("Red zone, stay away!")

def Backtest2(self, Var):
    """
    Performs a backtest of the Value at Risk (VaR) over a 500-day lookback period to assess model accuracy.

    Parameters:
    Var (np.array): An array of VaR values to be tested.
    """
    errors = sum(np.where(self.PnL()[self.lookback:] + Var < 0, 1, 0))
    print("Number of exceptions =", errors)
    if errors < 11:
        print("Green zone, model looks to be correct")
    elif errors < 20 and errors >= 11:
        print("Yellow zone, watch out!")
    else:
        print("Red zone, stay away!")
```

Backtesting can be done for all calculated VaRs (k=1007) of **250** days period, in which case:

- the green zone ends with 13 breaches.
- the yellow zone starts with 14 breaches.
- the red zone starts with 24 breaches.

Empirical	**21**
Normal	**32**
Weighted	**14**

Backtesting for all ($k = 755$) calculated VaRs with **500** days lookback period:

- the green zone ends with 10 breaches.
- the yellow zone starts with 11 breaches.
- the red zone starts with 20 breaches.

Empirical	**10**
Normal	**12**
Weighted	**8**

Let's denote:

- PnL_i as the profit and loss on day i.
- VaR_i as the Value at Risk (VaR) on day i.
- The window length is 250 days.

For the i-th rolling window, the formula for calculating the number of exceptions is:

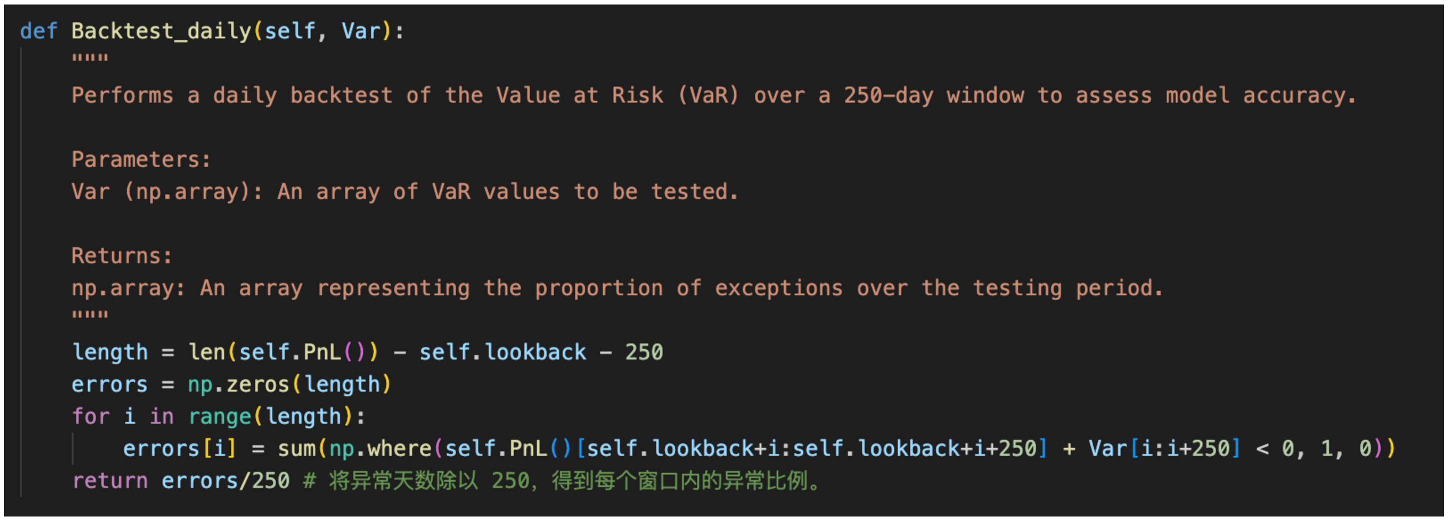
$$\text{errors[i]} = \sum_{\text{j=i}}^{\text{i+250}} \mathbb{1}(\text{PnL}_{\text{j}} + \text{VaR}_{\text{j}} < 0)$$

where $\mathbb{1}$ is an indicator function that returns 1 if x is true and 0 otherwise.

The result returned is the proportion of exceptions:

$$\text{proportion\_of\_exceptions[i]} = \frac{\text{errors[i]}}{250}$$

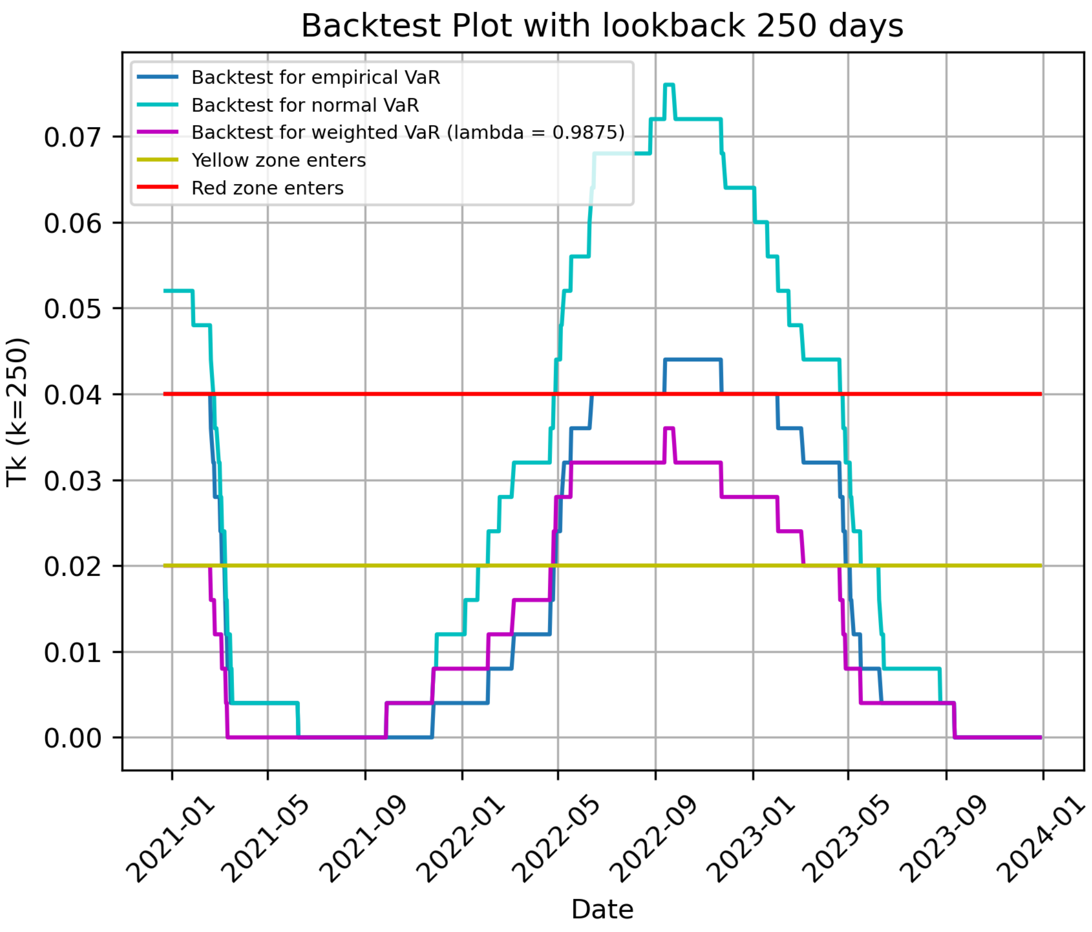
The purpose of this function is to evaluate the accuracy and robustness of the VaR model by calculating the proportion of exceptions within each rolling window. If the proportion of exceptions is too high, it may indicate that the VaR model is not adequately predicting risk.

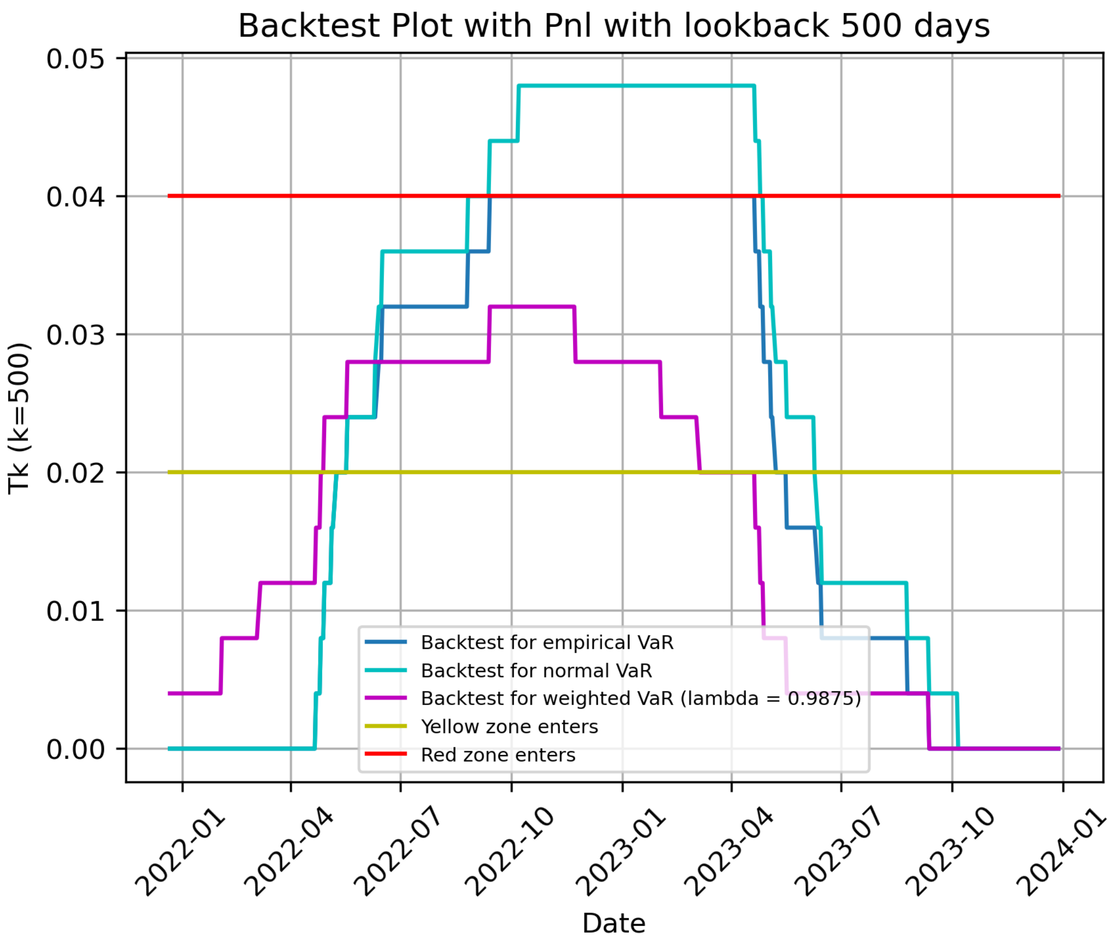


```python
def Backtest_daily(self, Var):
    """
    Performs a daily backtest of the Value at Risk (VaR) over a 250-day window to assess model accuracy.

    Parameters:
    Var (np.array): An array of VaR values to be tested.

    Returns:
    np.array: An array representing the proportion of exceptions over the testing period.
    """
    length = len(self.PnL()) - self.lookback - 250
    errors = np.zeros(length)
    for i in range(length):
        errors[i] = sum(np.where(self.PnL()[self.lookback+i:self.lookback+i+250] + Var[i:i+250] < 0, 1, 0))
    return errors/250 # 将异常天数除以 250，得到每个窗口内的异常比例。
```

Depending on the value of T_k we can fall into one of three zones, for *k*=250 we have:

- Green: $T_k \in [0,\ 0.02)$ - correct estimator has about 90% chance to be there,
- Yellow: $T_k \in [0.02\ 0.04)$ - correct estimator has about 10% chance to be there,
- Red: $T_k \in [0.04,\ 1]$ - correct estimator has about 0.01% chance to be there.


Backtest Plot with lookback 250 days
Backtest for empirical VaR
Backtest for normal VaR
Backtest for weighted VaR (lambda = 0.9875)
Yellow zone enters
Red zone enters
Tk (k=250)
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
2021-01
2021-05
2021-09
2022-01
2022-05
2022-09
2023-01
2023-05
2023-09
2024-01
Date


Backtest Plot with Pnl with lookback 500 days
Backtest for empirical VaR
Backtest for normal VaR
Backtest for weighted VaR (lambda = 0.9875)
Yellow zone enters
Red zone enters
Tk (k=500)
0.05
0.04
0.03
0.02
0.01
0.00
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01
Date

4 Conclusion

The best estimators for us are **historical and weighted historical VaRs**.

We see that the weighted one is slightly harder to implement and we must find appropriate decay factor. However we are more **flexible** in this case.

To conclude, we decided to choose **weighted historical VaR** as the best estimator. As stated by the regulator, we chose a 250-day or 500-day lookback period.