Project Instruction: Building a Python Library for Non-Parametric Risk Measures in Fixed Income

---

Objective

Develop a Python library to model non-parametric risk measures for **single fixed-income positions** and **fixed-income portfolios**. The library will calculate Value at Risk (VaR), Expected Shortfall (ES), and Extreme Value Theory (EVT)-based risk metrics using quantile and bootstrap methods.

Key Components

1. **Class 1**: `FixedIncomeNprmSingle` (for single positions)
2. **Class 2**: `FixedIncomeNprmPort` (for portfolios)

Step-by-Step Implementation Guide

---

1. Class `FixedIncomeNprmSingle`

Models non-parametric risk measures for a single fixed-income position.

1.1 Initialization (`__init__`)

Purpose: Validate inputs and initialize parameters.
Parameters:

- `returns` (np.ndarray): Historical returns of the fixed-income security.
- `position` (float): Quantity held (positive = long, negative = short).
- `alpha` (float, default=0.05): Significance level for VaR/ES.
- `method` (str, default="quantile"): Calculation method ("quantile" or "bootstrap").
- `n_bootstrap_samples` (int, default=10,000): Bootstrap samples if method="bootstrap".

Steps:

1. Import input validation functions (e.g., `validate_returns_single`, `check_position_single`).
2. Validate inputs using these functions.
3. Initialize attributes: `var`, `es`.
4. Automatically call `fit()` to compute VaR and ES.

1.2 Method `fit()`

Purpose: Compute VaR and ES using the specified method.

Formulas:

- **Quantile Method**:
  - **Long Position**:
    [\text{VaR} = -\text{position} \times Q_\alpha(\text{returns})]
    [\text{ES} = -\text{position} \times \mathbb{E}[\text{returns} ,|, \text{returns} < Q_\alpha(\text{returns})]]
  - **Short Position**:
    [\text{VaR} = -\text{position} \times Q_{1-\alpha}(\text{returns})]
    [\text{ES} = -\text{position} \times \mathbb{E}[\text{returns} ,|, \text{returns} > Q_{1-\alpha}(\text{returns})]]
- **Bootstrap Method**:
  - Generate `n_bootstrap_samples` of returns.
  - Compute VaR as the mean of the α-quantile across all samples.
  - Compute ES as the mean of losses exceeding VaR in each sample.

Implementation:

1. Use `np.quantile` for quantile-based calculations.
2. For bootstrap, use `np.random.choice` to resample returns.
3. Round results to 4 decimal places and ensure non-negative values.

1.3 Method `summary()`

Purpose: Return a dictionary of risk metrics.
Metrics:

- `var`, `es`, `maxLoss`, `maxExcessLoss`, `maxExcessLossOverVar`, `esOverVar`.

Steps:

1. Compute `maxLoss` as the worst loss (long: min return × position; short: max return × position).
2. Calculate ratios (e.g., `esOverVar = es / var` if `var ≠ 0`).

1.4 Method `evt()`

Purpose: Estimate VaR/ES using Extreme Value Theory (GPD).

Formulas (GPD Parameters):

- Fit GPD to tail losses using Maximum Likelihood Estimation (MLE).
- Shape (ξ) and scale (β) parameters are estimated via `scipy.optimize.minimize`.
- **VaR under EVT**:
  $$\text{VaR} = u + \frac{\beta}{\xi} \left[\left(\frac{n}{n_u}(1-\alpha)\right)^{-\xi} - 1\right]$$
- **ES under EVT**:
  $$\text{ES} = \frac{\text{VaR} + \beta - \xi u}{1 - \xi}$$

Steps:

1. Extract tail losses beyond the `quantile_threshold` (default=0.95).
2. Optimize GPD log-likelihood to estimate ξ and β.
3. Handle long/short positions by adjusting tail direction.

---

2. Class `FixedIncomeNprmPort`

Extends risk measures to a portfolio of fixed-income positions.

2.1 Initialization (`__init__`)

Parameters:

- `returns` (np.ndarray): Matrix of returns (rows=periods, columns=securities).
- `positions` (list): List of positions for each security.

Steps:

1. Validate inputs (e.g., `len(positions) == returns.shape[1]`).
2. Compute cumulative portfolio returns in `fit()` .

2.2 Method `fit()`

Purpose: Compute portfolio VaR and ES.

Formulas:

- **Portfolio Returns**:
  [\text{Portfolio Return}*t = |sum*{i=1}^n (\text{Position}*i |times |text{Return}*{t,i})]
- **VaR/ES**: Same as `FixedIncomeNprmSingle` but applied to aggregated portfolio returns.

Implementation:

1. Use `np.sum(axis=1)` to aggregate returns.

2.3 Method `MargVars()`

Purpose: Compute Marginal VaR for each position.

Formula:
[\text{MVaR}*i = \frac{\partial \text{VaR}}{\partial \text{Position}_i} \approx \frac{\text{VaR}*{\text{new}} - \text{VaR}_{\text{original}}}{\Delta \text{Position}_i}]

Steps:

1. Perturb each position by `scale_factor` (default=0.1).
2. Recompute VaR and measure the difference.
3. Restore original positions after each iteration.

---

3. Input Validation & Support Functions

- Create functions in `inputsControlFunctions.py` (e.g., `validate_returns_single`, `check_alpha`).
- Ensure all inputs are non-negative where required (e.g., `alpha ∈ (0,1)`).

---

4. Testing

1. **Unit Tests**:
   - Validate VaR/ES calculations against known examples.
   - Test edge cases (e.g., `position = 0`, `alpha = 0.01`).
2. **Example Dataset**:
   - Use simulated or historical bond returns.

Deliverables

1. Python modules:
   - `non_parametric_fixed_income.py` (main logic).
   - `inputsControlFunctions.py` (validation).
   - `SupportFunctions.py` (e.g., `gpd_log_likelihood`).
2. Documentation:
   - Docstrings for all methods.
   - Example Jupyter Notebook.

Appendix: Formulas

- **VaR (Quantile)**: ($Q_\alpha = \text{quantile}(\text{returns}, \alpha)$).
- **ES**: ($\mathbb{E}[\text{loss} ,|, \text{loss} \geq \text{VaR}]$).
- **GPD Log-Likelihood**:
  $$\ell(\xi, \beta) = -n_u \ln \beta - \left(1 + \frac{1}{\xi}\right) \sum \ln\left(1 + \frac{\xi}{\beta}(x_i - u)\right)$$

Next Steps:

- Start with `FixedIncomeNprmSingle`, implement `fit()` and `summary()`.
- Proceed to EVT and portfolio classes.
- Test each method incrementally.

In [ ]:

```

```