Minerva Technologies Task

Edoardo Bertoli
bertoliedoardo99@gmail.com

August 8, 2024

Contents

1 Introduction

This is the report for the task assigned to me by the team at Minerva Technologies. The purpose of this task is to develop a long-only trading model of five stock indices resulting in a portfolio weight matrix. The input data is comprised of daily historical logarithmic returns from the 30th September 2003 to the 31st December 2019 for the five stock indices, as well as a series of other factors: rates, commodities, Forex currency crossings. Macroeconomic indices of the stock indices' countries and fundamentals of the stock were also given.

Given the amount of data and task's objective I decided to use a simple but effective model: the portfolio will be optimized using a risk-parity approach were the allocation of assets is determined by imposing that each asset's contribution to the portfolio total variance is equal. In order to find the covariance matrix of expected returns based on the historical data, necessary to optimize the portfolio, I decided to use a linear multi-factor model combined with Principal Component Analysis on the factors in order to reduce their dimensionality but also eliminate linear correlation between factors, grouping them by how much they contribute to the overall variation of the factors data.

The model will be tested by dividing the dataset as follows: the last year of data will be reserved as the testing data, the rest will be used for training purposes. For each week the model will be retrained and the weights recalculated in order to calculate the portfolio weight matrix and test the portfolio return and volatility. Maybe I can also do K-Fold Cross validation or use a rolling-window approach to test the model in different time periods.

Even though normality is required only for the residuals of the linear regression, having data that is approximately normally distributed makes calculation easier since most of the information is captured by the first two moments: mean and variance. For this reason Section 3 is dedicated to testing data for its normality as well as for stationarity and cross-asset relationships to better understand the nature of the data and better interpret the result of the model.

This approach is very simple and disregards completely financial data that shows heavy time dependence and non-linear relations with the factors. To treat this kind of data, more complicated models can be explored, such as a Long Short-Term Memory recurrent neural network paired with Layerwise Relevance Propagation to interpret the expected output of the neural network (CITATION NEEDED). The same goes for the risk parity approach that can be improved by considering non-normally distributed returns(CITATION NEEDED).

2 Data Cleaning

The input data given for this task is comprised of:

1. Five historical series of daily logarithmic returns pertaining to stock indices from five different geographic regions. (Identified throughout the report as 'Stock returns')
2. Six historical series of Fundamental Indicators for each geographic area. ('Fundamentals')
3. Two historical series of daily log returns for each geographic area. ('Rates returns')
4. Two historical series of Macroeconomic indicators for each geographic area. ('Macro indices')
5. Three historical series of daily log returns related to currency crossings. ('Forex returns')
6. Three historical series of log returns for three different commodities. ('Commodities returns')

The historical series span from 30th September of 2003 to 31st of December of 2019. Before running any type of cleaning process I applied a log transformation to the historical series of Macroeconomic indices and Fundamentals Indicators.

From a first look at the data it is obvious that some cleaning is required. There are invalid numeric values like zeroes for logarithmic returns and NaN (Not-a-Number). I decided to use a careful and simple approach to avoid introducing too much noise in the data. First I replaced every type of invalid value into a NaN, then I forward and backward filled the data to make sure each value in each Dataframe is a valid number (float64).

At this point a pretty straightforward problem arises: forward and backward filling just copies data, either from after or before, leaving us with a lot of repeated values. To smooth out this values a rolling average of five days is applied only on the previously filled values.

This is the most simple approach in order to do the minimum required to clean the input data. Another way could be to just throw away every value that is not valid, but it is better to try and keep as much data as possible for consistency, especially since part of the data will be reserved for testing the model's performance.

More invasive approach can be used, like Winsorization and LOWESS (Locally Weighted Scatterplot Regression), but the risk of changing the data distribution too much without really moving towards a normal distribution is high.

During the data cleaning process the historical data was resampled into different timeframes. The resampling into daily data has been done right after the data cleaning process.

The module **clean_data.py** is dedicated to cleaning the data. It will read raw data from a file called 'data.xlsx' and will output an xlsx file called 'formatted-data.xlsx' with the cleaned data.

3 Historical Data Analysis

This stage is dedicated to extracting as much information as possible on the cleaned data. This step is necessary to have an idea of what kind of data we have in order to be able to correctly inteprept the results of the chosen model and understand its limitations. The data used in this test is from the 'formatted-data.xlsx' file. It has only been cleaned and scaled to zero mean and unit variance when necessary. It has not yet been divided into training and testing. The custom python module **eda.py** is dedicated to this section.

3.1 Normality

The assumption that stock returns follow a univariate normal distribution not always holds for financial data. When it does, at least approximately, calculation and interpretation of calculations tend to be easier because most of the data information can be captured by the first two moments: mean and variance. For this reason the stock returns data is checked against a standardized gaussian distribution to understand each stock's behaviour and understand how heavy the assumption of data distributed normally is.

In Table 1 the results of fitting each stock returns against a normalized Gaussian distribution are shown. The data has been standardized to unit variance and zero mean. For each asset the first four moments are provided, as well as the p-value for the Anderson-Darling test. The ratio for p-value acceptance is 0.05%. For this calculation I used the function *scipy.stats.goodness_of_fit* from the python library *Scipy*. The reference values for a normalized Gaussian distribution are:

- Mean: 0
- Standard Deviation: 1
- Skewness: 0
- Excess Kurtosis (Kurtosis - 3): 0

Stock return	Mean	Standard deviation	Skewness	Excess Kurtosis	Anderson-Darling p-value
Indice Azionario Paese 1	-6.701652777742044e-18	1.0001179175760244	-0.26527445129737953	6.252408544066505	0.0001
Indice Azionario Paese 2	1.0052479166613066e-17	1.0001179175760244	-0.06867531417815065	6.560510364685937	0.0001
Indice Azionario Paese 3	-3.350826388871022e-18	1.0001179175760244	-0.06963505684987985	7.389173595223623	0.0001
Indice Azionario Paese 4	4.607386284697655e-18	1.0001179175760244	-0.06963505684987985	9.171379931995125	0.0001
Indice Azionario Paese 5	-7.539359374959799e-18	1.0001179175760244	-0.15364747544085475	8.853385699414373	0.0001

Table 1: Descriptive statics and results of fit of stock returns against a standardized Gaussian distribution.

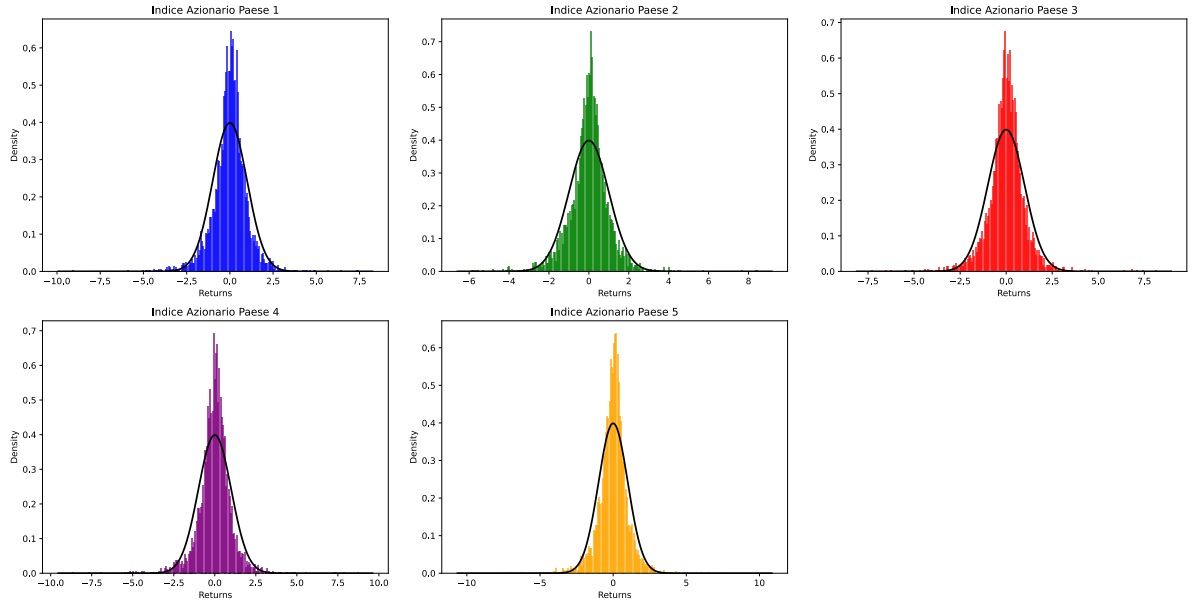

Figure 1: Comparison of stock returns fits against a standardized Gaussian distribution

As we can see both from the plot in Figure 1 and Table 1 the problem is clear: the first two moments are extremely close to being those of a standardized gaussian, in fact the shape of the histogram is not extremely different from that of gaussian distributed data. The skewness is relatively small, in fact we cannot see any extreme rupture of symmetry to the right or left. The problem is exactly the excess kurtosis which is way too big, because of the huge amount of data clustering at the center, which would make the gaussian distribution way too peaked. This also explains the low pvalue of the Anderson-Darling test which by taking into account the skewdness of the data has to conclude that the data is not normal. There have been proposed ways of modeling the stock returns as more complex distributions, like elliptical distributions as shown in [**risk-parity-hard**].

3.2 Time Dependence

Stationarity is a very important characteristic of financial data. It has to be taken into consideration especially when using Multi-Factor models that do not inherently consider the time relations between data. Also the assumption of stock returns following a normal distribution requires a non time-varying mean and standard deviation. For this reason in Figure 2 are shown the time plots of the weekly standard deviation of the stock returns.

As we can see the variance is mostly oscillating between 0 and 1, apart from huge spikes of volatility in the stock returns that coincide with periods of extreme economic uncertainty (like the 2007-2009 "Great Recession"). This needs to be taken into account when interpreting the portfolio results, especially to understand how much the model can adapt to sudden changes in the market.

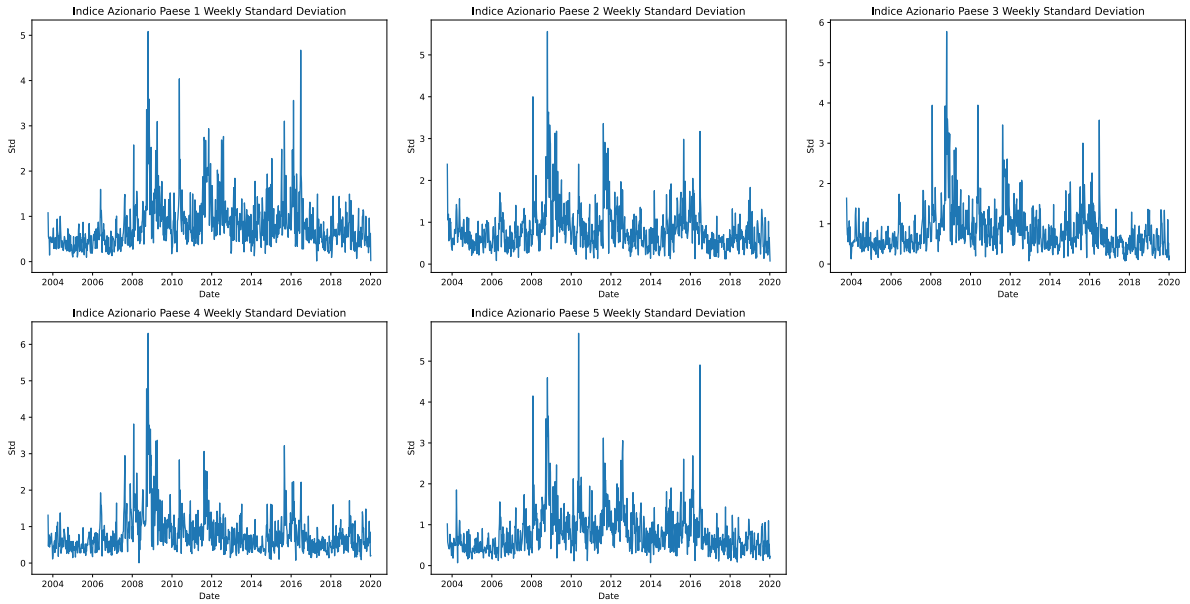

Figure 2: Comparison of stock returns fits against a standardized Gaussian distribution

3.3 Correlation Analysis

Linear correlation between factors and stock is extremely important to study in order to efficiently implement Principal Component Analysis on the factors and understand the correlation between factors and stock returns. The correlation between factors will be eliminated choosing the direction of greater variance using PCA. As we can see in Figure 3 there are two main areas of relative higher correlation: between the stock returns (top-left) and between macroeconomic indices and fundamentals of the stock returns (triangle below the diagonal). For this reason I will first apply PCA between Macro indices and fundamentals. Then I will apply again PCA between the results of the previous steps and the rest of the data, apart from the stock returns, on which no PCA will be applied.

3.4 Historical data visualization

Another very important piece of information can be extracted from the historical behaviour of the stock returns throughout the whole dataset. In Figure 4 the stock returns are plotted against time, from the first to the last date available in the dataset. As we can see the stock index for Country 1, 2, 3 and 4 close the timeframe with a positive log return, while the index stock of Country 5 closed with a negative log return. Also we can see that the stock index of the first country has the greatest return over this time-period, while Country 2, 3 and 4 were pretty close during the entire time-frame. These information must be kept in mind when evaluating the allocation of these assets in the portfolio.

4 Linear Multi-Factor Model with PCA

This section describes the process of implementing and training a multi-factor model in conjunction with Principal Component Analysis on the factors for forecasting expected returns and volatility to later use for the portfolio optimization.

First it's important to establish the goal of this model: finding a relationship between stock returns and all the other historical daily data in the dataset, which we will call factors (rates, currency crossings, commodities, macro indices, fundamentals of stock), such that it's possible to use this relationship to forecast future expected stock returns knowing the values of the factor at

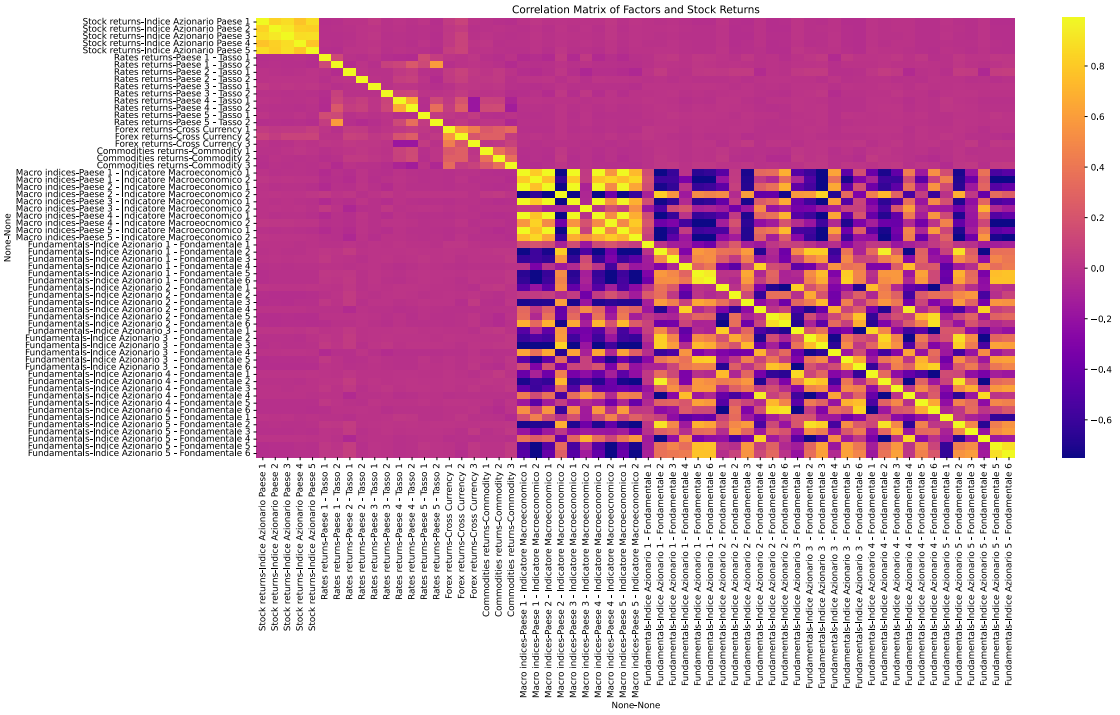

Figure 3: Correlation matrix heatmap of the whole dataset given as input for this task

the current time. Three main questions immediately come to mind: **what distribution should one use to model stock returns as random variables ? What kind of mathematical relationship is there between the factors and the stock returns ? How should one choose which factors to use ?**

The first question requires an assumption, usually it is required that either the actual returns or the log returns follow a normal distribution. As shown in the previous section, the stock log returns are 'decently' close to being described by a normal distribution if not for the presence of clustering near the center and not normal tails. For this task we will proceed as if the stock returns (as logarithmic daily data) follow a univariate gaussian distribution so that the vector of stocks follows a multivariate normal distribution:

$$\vec{r} = N(\vec{\mu}, \vec{\sigma}) \tag{1}$$

where: $\vec{r} = (r_1, r_2, r_3, r_4, r_5)$ is the stock returns vector, $\vec{\sigma} = (\sigma_1, \ldots, \sigma_5)$ is the vector of stock returns' standard deviations.

The portfolio returns will follow this normal multivariate distribution:

$$\vec{r_p} = N(\vec{w} \cdot \vec{\mu}, \vec{w}^T \mathbf{\Sigma} \vec{w}) \tag{2}$$

where: $\vec{w}$ is the vector of weights which will define the portfolio allocation. The weights are the result of the optimization process. $\Sigma = (\sigma_{i,j})$ $(i, j = 1, ..., 5)$ is the covariance matrix of expected returns. Normality is not really required for the multi-factor model but it is required when optimizing the portfolio using a risk-parity approach where all the information present in the objective function solely comes from the covariance matrix of the forecasted stock returns. This needs to be taken into account when interpreting both the model and the portfolio optimization results. I COULD CITE THE TEXTBOOK I USED!

Now for the second question we need another assumption: the relationship between the stock returns and the chosen factors is **linear**, so:

$$r_i = a_i + B_{1,i} F_{1,i} + ... + \epsilon_i \tag{3}$$

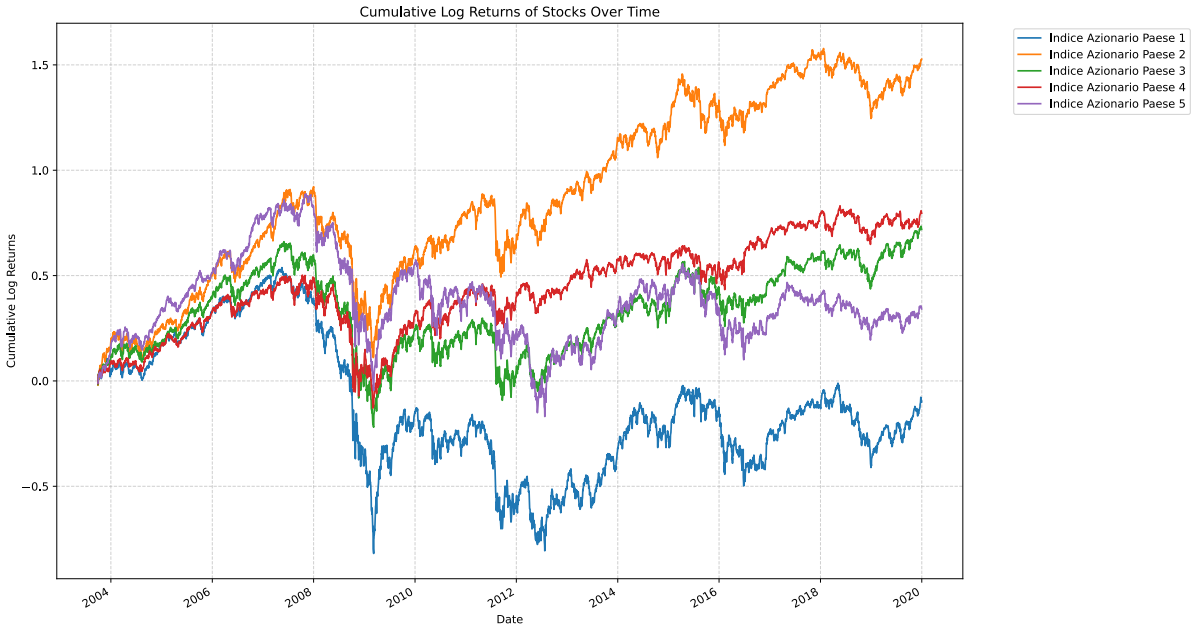

Figure 4: Cumulative log returns for each stock index over the entire dataset time period

where ϵ is the random error from the regression model used to determine the parameters in the above equation. The only thing a linear regression model requires is the normality of ϵ. Also it would help to have factor returns that are not correlated between each other, in order to lower the probability of overfitting.

Finally to tackle the problem of choosing the factors and at the same time dealing with correlation between factors I decided to use Principal Component Analysis on the factors in order to reduce their dimensionality by choosing only the first K principal components such that a certain percentage of the total variance of the factors is explained, usually around $80 - 90\%$ is used.

Equation (3) can be written in vector-matrix notation:

$$\vec{r}_{t+1} = \vec{b_0} + \mathbf{B}\vec{F}_t + \vec{\epsilon} \tag{4}$$

Where: $\mathbf{B} = (b_{i,j})$ is the matrix of factors exposures, i runs over the number of stock returns (N) while j runs over the number of factors (K) chosen for the regression model, thus it's a (N,K) matrix. The matrix of the factors exposures is the objective of the regression model. The vector $\vec{F}_t$ is a (1,K) vector which represents the factor at time t. The vector $\vec{r}_{t+1}$ represents the expected returns at time t+1. The vector $\vec{\epsilon}$ represents the vector of errors of the regression model.

Given this model, it can be shown that the covariance matrix of expected returns can be written as:

$$\mathbf{\Sigma} = \mathbf{B}\mathbf{\Sigma}_F\mathbf{B}^T + \mathbf{S} \tag{5}$$

Where: $\mathbf{S}$ is a diagonal matrix consisting of each stocks' error ϵ's variance. $\mathbf{\Sigma}_F$ is the covariance matrix of the factors used in the regression model. Since we are using PCA, this matrix will always be diagonal. It is very important to notice that the only place where time dependence of data is considered is in the regression model where the stock returns of time t+1 are calculated based on the factors at time t. There are more sophisticated methods to model the time-dependence of data, for example using multiple lagged features, not only the day before, or using more advanced time series models like ARIMA or GARCH for capturing complex temporal patterns.

There exists more complex models which incorporate both time-volatility and non-linear relations between stock returns and factors. For example in [**nakagawa**] it is proposed the use of a Long Short-Term Memory Recurrent Deep Neural Network, combined with Layer-wise Relevant

Propagation (LRP) in order to interpret the results. As they show in the table comparing different methods the LSTM + LRP model definitely outperforms the linear multi-factor model.

The idea is to use a linear regression model to calculate both **B** and **S**. Then it is possible to obtain the covariance matrix of expected returns and use it to optimize the portfolio as shown in Section 6.

4.1 Principal Component Analysis on Factors

As we have seen in Figure 3 the highest correlation is between Macro returns and Fundamentals. For this reason, and also to simply the interpretation of the remaining principal components, I decided to use standard PCA between Macro returns and Fundamentals. With these principal components selected I will run standard PCA again against all the other remaining factors. The standard Principal Components Analysis basically diagonalizes the covariance matrix and finds its eigenvectors. Using these eigenvectors as the transformation matrix it is possible to write every element of the initial base into the new base where the covariance matrix is diagonal. One can then choose how many principal components to keep, thus how many of the eigenvectors with the biggest eigenvalues to keep. These principal components are directions in the new base that explain the most variance of the analyzed data. CHECK THIS SHIT PARAGRAPH

Ater PCA I was able to keep 19 principal components out of 56 total factors such that the remaining principal components explain 95% of the total variance of factors.

ADD GRAPH OF PERCENTAGE OF FACTORS CONTRIBUTING TO THE TOP PRINCIPAL COMPONENT

4.2 Cross Validation of Linear Regressors

Here I want to use k-fold cross validation (dividing using TimeSeries) to compare different linear regressor and see which one performs better on the training data.

4.3 Implementation

- Explain the linear regression model used and why we chose a Robust estimator (underlying normality of data assumption is not necessary)

5 Model's Performance Analysis

- RMSE / MSE
- R-Squared
- Residual analysis (normality and correlation, heteroschedasticity)
- Mean Absolute Error
- Rolling window approach

6 Portfolio Optimization

1. risk-parity (uses only covariance matrix of forecasted returns
2. mean-variance (sucks)

The most important thing is to handle correctly the risk!

Rebalancing: train model, forecast, optimize, calculate weight, then change timeframe, retrain, forecast, optimize and calculate weight I think this is how you create a portfolio weight matrix. Then you can also simulate a real scenario, starting from data and given a certain frequency you update the data and remove the oldest timeframe data and repeat. VERY IMPORTANT TO CALCULATE STUFF ABOUT RISK AND PERFORMANCE OF PORTFOLIO:

1. Sharp Ratio
2. Mean Squared Error
3. VaR
4. sVaR
5. ask claude
6. study optimal weights over time using rolling window rebalancing

This is the idea:

Divide dataset into some years of training (like 5) and 1 year of testing. Run the model and calculate weights. Make sure the relationship between returns and factors in the regression model is lagged like:

$$y_t = bf_{t-1} \tag{6}$$

Then test this model with a balancing frequency of one week: start with first weights from training data, calculate portfolio return in that week given actual prices. Use that week data to rerun the model, recalculate new weights fore the next week and repeat. Like this you can create a portfolio weight matrix of weeks and assets weights over the year of testing data.

Then I could do two approaches: use rolling window (shift training and testing sets by one year in the future and repeat calculation of new portfolio weight matrix) or I could use different year for the testing data to see how my model reacts to different markets (K-Fold Cross Validation is the way to do every possible combination).

Anyway each time I calculate the weights I have to calculate stats for the portfolio such as returns and other stuff (Sharpe Ratio, VaR, SVaR and others)

I could do the overall performance testing on the last year and maybe testing only on the periods of greater variance movement in order to see how the model performs under critical economic conditions. There is alwyas K-Fold cross validation available.