Learning factors for stock market returns prediction

BORGES Alexandre, DEZONS Charles, LE ROMANCER Briac

Machine Learning and Applications

Master in Mathematics for Finance and Data

Abstract

We chose to work on a challenge released by Qube Research and Technologies in 2022, for the annual edition of ENS Challenge Data. Here is the challenge page for more information: Click here to visit the page.

Contents

1 Introduction

Predicting the future behavior of financial markets is a fundamental challenge in quantitative finance and machine learning. Among the various tasks in this domain, one of the most widely studied problems is the prediction of stock market returns, i.e., the relative price variations of financial assets over time. Successfully forecasting these returns is of considerable practical importance for portfolio optimization, risk management, and algorithmic trading.

Formally, consider a financial market composed of N stocks. Let $R_t \in \mathbb{R}^N$ denote the vector of returns at time t. The goal is to construct, at each time step t, a predictive signal $S_{t+1} \in \mathbb{R}^N$, based on the information available up to time t, such that the prediction overlap $\langle S_{t+1}, R_{t+1} \rangle$ is positive as often as possible. In other words, the alignment between the predicted and actual return vectors should indicate a profitable trading signal. However, this objective is far from trivial due to the high volatility, non-stationarity, and noise inherent in financial time series.

In this project, we investigate a modern variant of this classic prediction problem by combining ideas from linear factor models and representation learning. Instead of relying on pre-defined, expert-designed features, we aim to learn the explanatory factors directly from data in a structured and constrained way. Our approach is based on the assumption that there exists a set of latent factors that explain most of the predictive structure in the return series, and that these factors can be modeled as linear functions of past returns.

A natural first step is to express the prediction as a linear combination of F learned factors:

$$S_{t+1} := \sum_{\ell=1}^{F} \beta_\ell F_{t,\ell}$$

where $F_{t,\ell} \in \mathbb{R}^N$ represents the ℓ-th factor at time t, and $\beta_1, \ldots, \beta_F \in \mathbb{R}$ are scalar weights to be optimized. This formulation reflects the common financial intuition that market returns can often be decomposed into a combination of systematic components (factors).

Traditionally, the factors $F_{t,\ell}$ are chosen manually based on financial expertise. Popular examples include:

- the *5-day normalized mean return*: $R_t^{(5)}$,
- the *momentum indicator*: $M_t := R_{t-20}^{(230)}$,

where the moving average is defined by:

$$R_t^{(m)} := \frac{1}{\sqrt{m}} \sum_{k=1}^{m} R_{t+1-k}$$

The momentum indicator $R_{t-20}^{(230)}$ is the average return over a 230-day window, delayed by 20 days. The most recent 20 days of data are not used to reduce short-term noise or reversals. It captures the long-term trend, but with a cooling-off period to avoid potential recent overreactions.

However, such hand-crafted features rely heavily on domain knowledge and may fail to capture complex, evolving patterns present in high-dimensional financial data. To overcome this limitation, we propose a data-driven alternative in which each factor is learned as a linear combination of past return vectors. Concretely, we define:

$$F_{t,\ell} := \sum_{k=1}^{D} A_{k\ell} R_{t+1-k}$$

where $A_\ell = (A_{k\ell}) \in \mathbb{R}^D$ is the temporal weight vector (or filter) associated with the ℓ-th factor, and D is a fixed parameter representing the memory or time depth of the model. This construction enables the model to discover latent temporal patterns in past returns that are predictive of future movements.

To ensure that the learned factors are diverse and non-redundant, we impose an orthonormality constraint on the vectors A_ℓ, requiring:

$$\langle A_k, A_\ell \rangle = \delta_{k\ell}, \quad \forall k, \ell$$

with

$$\delta_{kl} = \begin{cases} 1 & \text{if } k = l \\ 0 & \text{otherwise} \end{cases}$$

This constraint implies that the matrix $A := [A_1, \ldots, A_F] \in \mathbb{R}^{D \times F}$ belongs to the *Stiefel manifold*, i.e., the space of matrices with orthonormal columns. This significantly complicates the optimization process, as the parameter space becomes non-linear and curved. However, it brings geometric structure to the model, potentially improving generalization and interpretability.
Thus, the full predictive model is parameterized by:

- a matrix $A \in \mathbb{R}^{D \times F}$ with orthonormal columns,
- a coefficient vector $\beta \in \mathbb{R}^F$.

In summary, this project aims to explore a constrained linear factor model for stock return prediction, in which both the factors and the predictive weights are learned directly from data. We hope to capture informative latent structures while maintaining model parsimony and interpretability.

2 Data Exploration

2.1 Data Description

In this project, we rely on historical stock return data as the foundation for training and evaluating our predictive model. The dataset spans a period of three years and includes daily return values for a set of 50 stocks from the same stock market. These 50 assets form the training universe, and their historical returns are used to learn the model parameters (A, β).

After training, we evaluate the generalization performance of the learned model on a testing universe composed of 50 different stocks, also observed over the same time period. This separation between training and testing assets ensures that our model does not overfit to a particular set of stocks and helps assess its ability to extract generalizable return-predictive signals.

The cleaned training input is provided in the form of a dataframe $X_{\text{train}} \in \mathbb{R}^{50 \times 754}$, where each row corresponds to a distinct stock and each column to a trading day. The associated target matrix Y_{train} contains the returns to be predicted, aligned accordingly.

For all experiments, we fix the following hyperparameters:

- Time depth $D = 250$, corresponding to roughly one trading year of past returns.
- Number of learned factors $F = 10$.

Figure 1 below displays the cumulative returns of the 50 training stocks over the entire period. This allows a first qualitative assessment of the diversity and dynamics present in the dataset.

2.2 Data Analysis

To better understand the structure and challenges of the data, we performed several preliminary analyses.

We ploted in Figure 2 histograms of daily returns across all assets to evaluate. This reveals that returns are approximately Gaussian but with heavy tails.

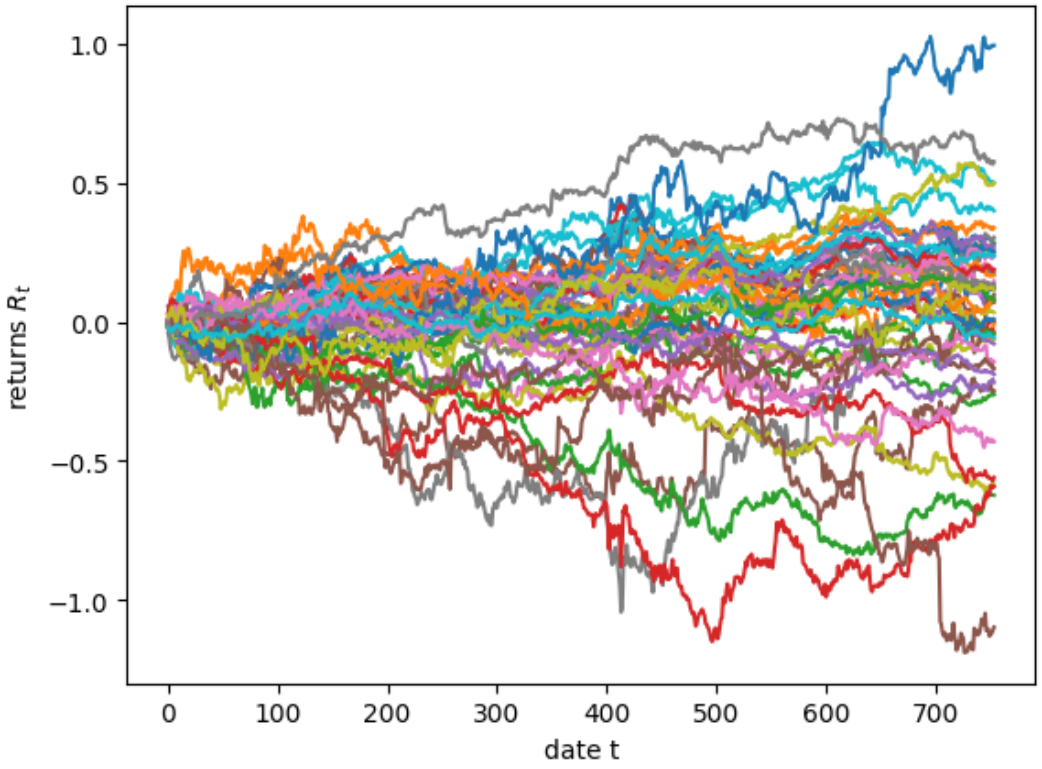

Figure 1: Cumulative returns of the 50 stocks from the training dataset over the 3-year period

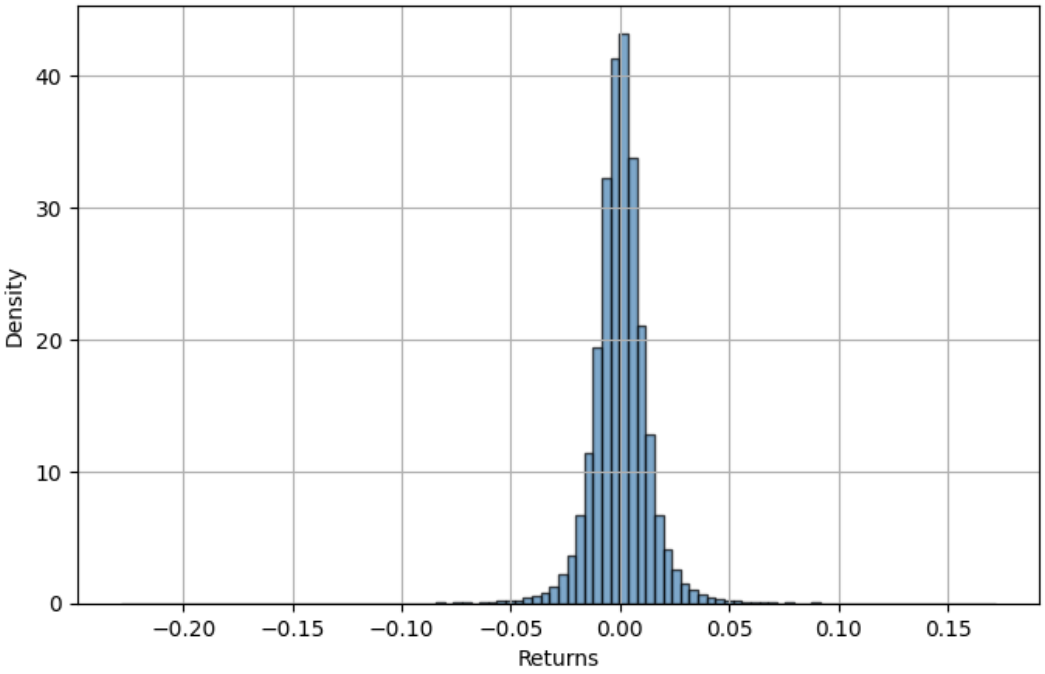

Figure 2: Distribution of the daily returns

We can also compute the empirical correlation matrix between the 50 stocks and visualize it using a heatmap as in Figure 3 . This helps us assess how much co-movement exists in the market and whether latent factors are likely to emerge from such structures. One can notice that several stocks are positively correlated, with correlation higher

than 0.5.

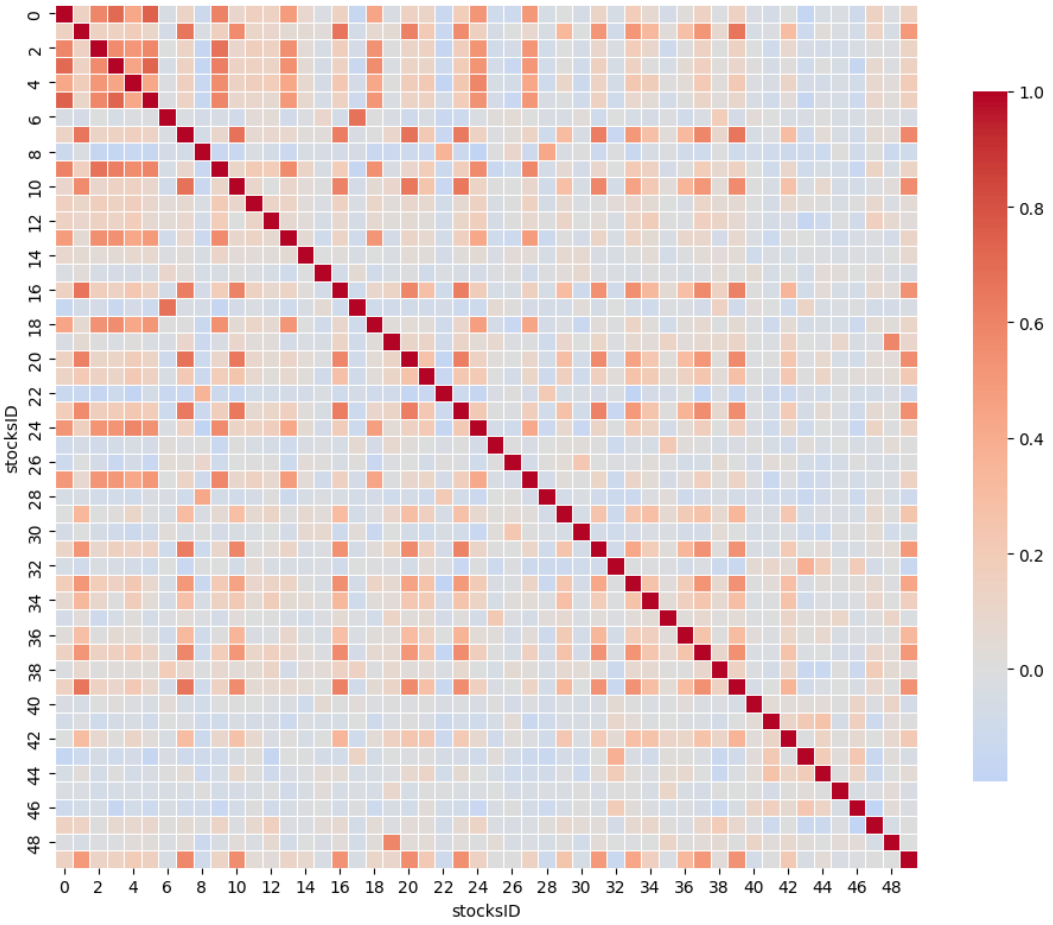

Figure 3: Heatmap of the correlations between the 50 stocks

We find that only three pairs of stocks have correlations greater than 0.7 (pairs of stocks 0/3, 0/5 and 3/5). Most stocks show weak correlation, indicating that the dataset is diversified, reducing concentrated risks. Thus, the dataset seems well-suited for diversification strategies.

We also computed and visualized in Figure 4 the rolling standard deviation (e.g., over 20-day windows) for several assets to capture time-varying volatility. Volatility clustering is a well-known phenomenon in finance and could influence factor learning. Here, the volatility remains between 0.1 and 0.2 over the observed period. For an equity market such as the S&P 500, these are relatively calm and fairly typical market conditions. Although it stays relatively stable, a noticeable increase can be observed between dates $t = 400$ and $t = 500$.

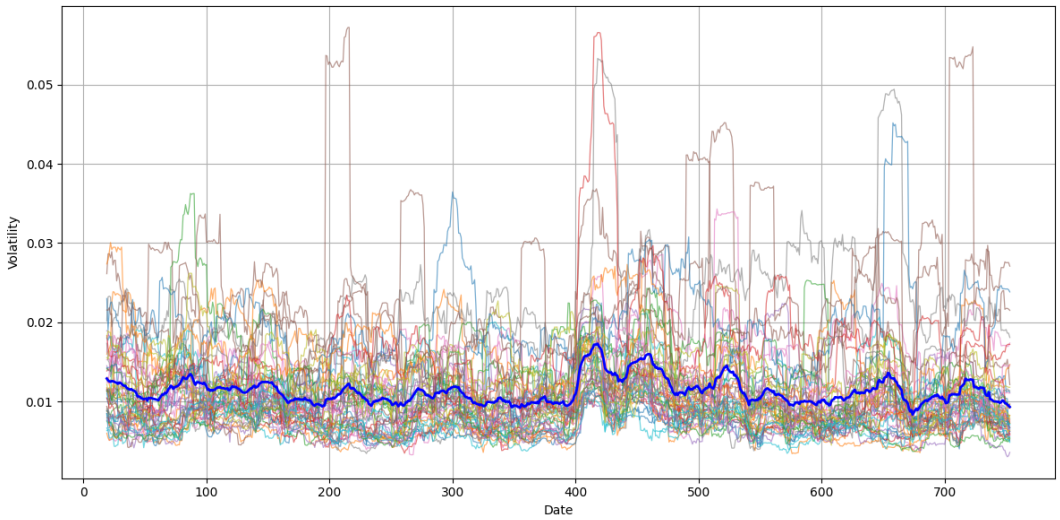

Figure 4: Sliding volatility on 20 days for the 50 stocks (average volatility in blue)

We also wondered about the reasoning behind the organizers' choice to limit the number of factors to $F = 10$. Applying PCA to the return matrix provides insight into how much of the return variance is explained by the top eigenvectors (i.e., latent linear directions). Each principal component captures a linear combination of stocks that best explains the variance of returns over time. This helps us justify the choice of $F = 10$ factors of this challenge. Actually, using $F = 10$ factors enables to explain more than 60% of the variance (see Figure 5).

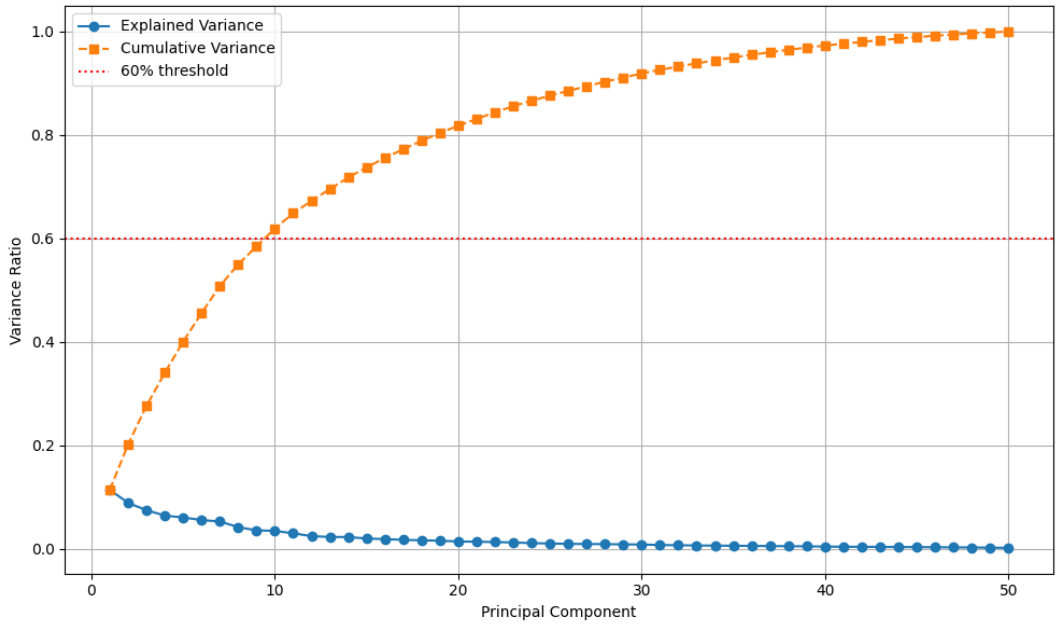

Figure 5: Explained/Cumulative variance using PCA on the 50 stocks

3 First Models

3.1 Evaluation metrics

The quality of the predictive model defined by the parameters (A, β) is assessed using the following metric:

$$\text{Metric}(A, \beta) := \frac{1}{504} \sum_{t=250}^{753} \frac{\langle \tilde{S}_t, \tilde{R}_t \rangle}{\|\tilde{S}_t\| \|\tilde{R}_t\|}$$

where $\tilde{R}_t \in \mathbb{R}^{50}$ represents the return vector of the 50 *testing* stocks at time t, and $\tilde{S}_t \in \mathbb{R}^{50}$ is the model's prediction produced by the learned parameters A and β. The metric corresponds to the average cosine similarity between the predicted and actual return vectors over the evaluation period.
To ensure that the learned factor directions remain orthonormal, as required by the problem's geometric constraint, the evaluation metric includes an additional validity check. If the orthonormality condition is violated, i.e., if

$$|\langle A_i, A_j \rangle - \delta_{ij}| > 10^{-6} \quad \text{for any } i, j,$$

then the metric is assigned the value $\text{Metric}(A, \beta) := -1$, effectively penalizing any solution that violates the orthogonality constraint. Otherwise, the metric takes values in the range $[-1, 1]$, where 1 indicates perfect alignment between predictions and true returns, and -1 indicates perfect anti-correlation.

3.2 AutoRegressive (AR) Model

Before finding the optimal couple (A, β), we will start by studying simpler models where only one of the 2 parameters needs to be learned.
We have seen previously that the sliding volatility is quite stable over time, and we can see below that the mean remains relatively constant around 0. To simplify, we can make a rather rough assumption and assume that our time series may be stationary to apply an autoregressive model where the statistical properties of the time series are considered constant over time.

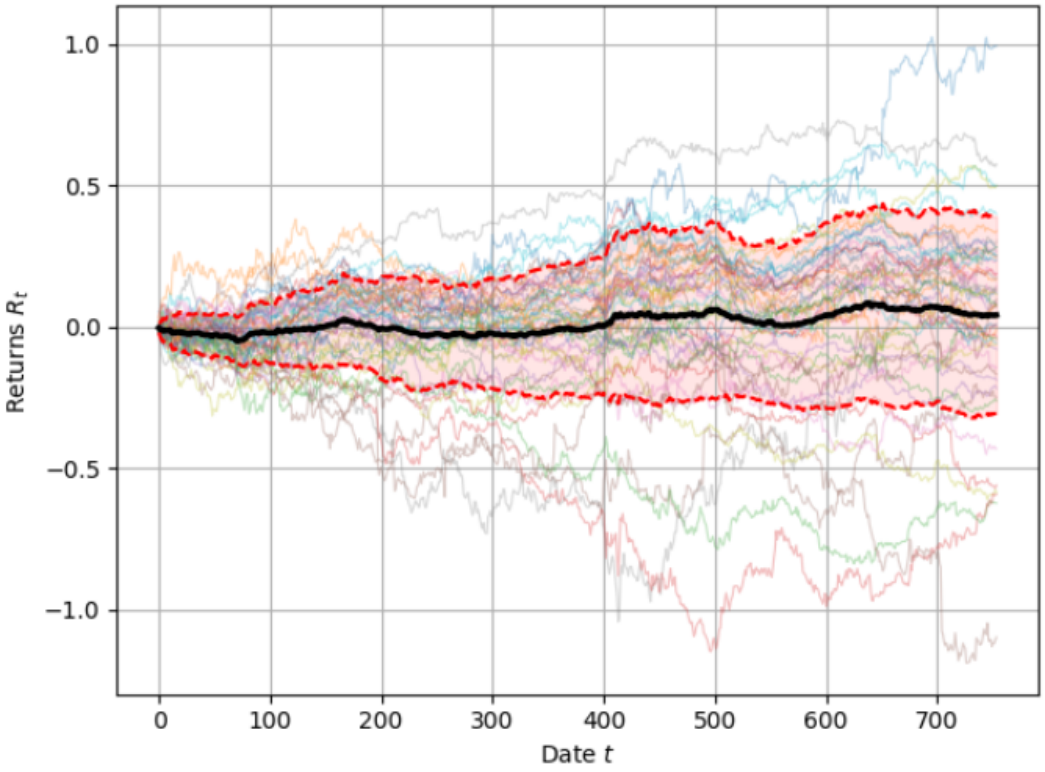

Figure 6: Evolution of the mean cumulative returns

We consider an autoregressive model of order F, denoted by AR(F), which predicts the return at time $t+1$ based on a linear combination of F previous daily returns. The model is defined as:

$$S_{t+1} := \sum_{\ell=1}^{F} \beta_\ell R_{t+1-\ell}$$

where:

- $R_{t+1-\ell}$ is the observed return at lag ℓ,
- $\beta_\ell \in \mathbb{R}$ is the coefficient corresponding to lag ℓ,
- S_{t+1} is the predicted return.

This model is simple and interpretable, as each coefficient directly reflects the influence of a specific lag on the future outcome. Because it reduces to a linear regression problem, it is computationally efficient and easy to implement. It is the first step, before moving to more complex, non-linear models, to highlight autocorrelation in our data, meaning past values likely contain information about future ones.
Let $\mathbf{R_t} \in \mathbb{R}^{N \times F}$ be the matrix of past returns:

$$\mathbf{R_t} = \begin{bmatrix} R_t^{(1)} & R_{t-1}^{(1)} & \cdots & R_{t-F+1}^{(1)} \\ R_t^{(2)} & R_{t-1}^{(2)} & \cdots & R_{t-F+1}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ R_t^{(N)} & R_{t-1}^{(N)} & \cdots & R_{t-F+1}^{(N)} \end{bmatrix}$$

and $\beta \in \mathbb{R}^D$ be the matrix of AR coefficients. Thus the predicted returns at time $t+1$ are:

$$S_{t+1} = \mathbf{R_t}\beta$$

To estimate the coefficients $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_F)^\top \in \mathbb{R}^F$, we minimize the mean squared prediction error in the training data set.

- $X \in \mathbb{R}^{N \times F}$ be the design matrix, where each row consists of lagged returns:

$$X = \begin{bmatrix} R_t & R_{t-1} & \ldots & R_{t-F} \end{bmatrix}$$

- $Y \in \mathbb{R}^N$ be the vector of target returns, with:

$$Y = R_{t+1}$$

The optimal coefficient vector β^* is obtained by solving the least squares problem:

$$\beta^* = \arg\min_{\beta \in \mathbb{R}^F} \|Y - X\beta\|_2^2$$

This problem has the closed-form solution (assuming $X^\top X$ is invertible):

$$\beta^* = (X^\top X)^{-1} X^\top Y$$

Once β^* is learned, predictions on new data can be made via:

$$S_{t+1}^{\text{pred}} = X_{\text{test}} \beta^*$$

This model serves as a linear baseline for comparison with more complex models, for $F = 10$ the performance are:

$$\text{Metric}_{\text{train}} = 0.0240$$

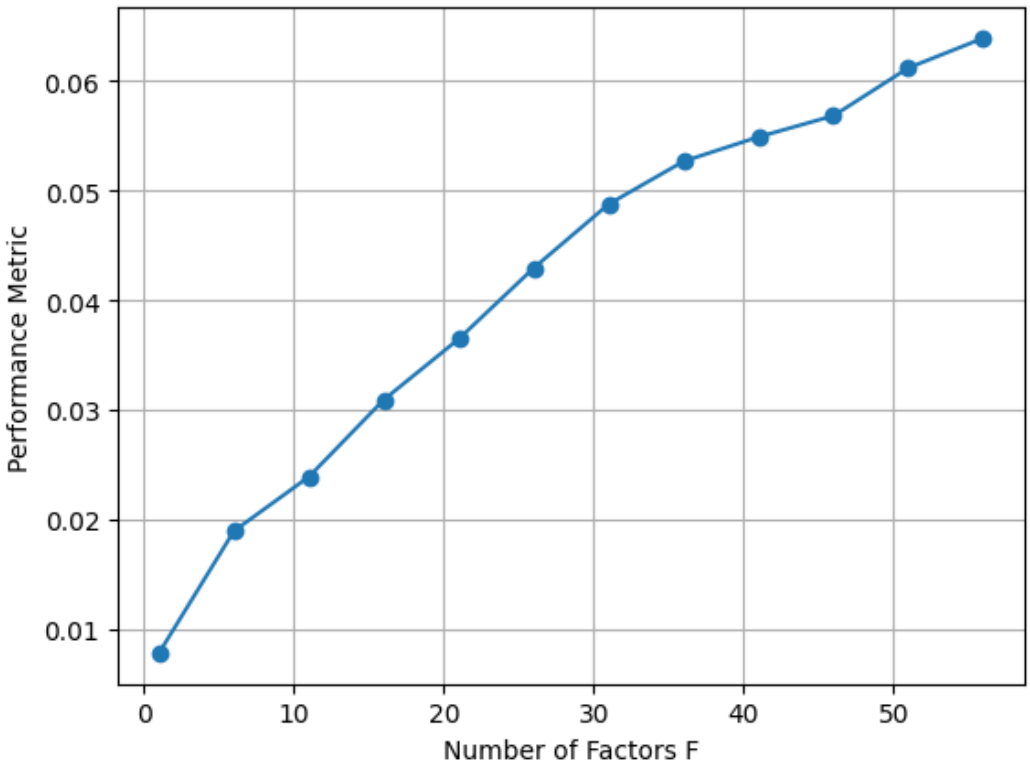

Figure 7: Evolution of the performance depending on F

We tested with many values of F up to 60 factors and we can see that when we increase the number of factors the performance on the train set improves up to a certain level. The risk of overfitting also increases.

3.3 Vector Autoregression (VAR) Model

To go a bit further, to capture the joint dynamics of multiple stock returns, we could consider the Vector Autoregression (VAR) model. Unlike the autoregressive model that treats each time series independently, VAR models allow for interdependencies among multiple stocks.
Let $R_t \in \mathbb{R}^N$ denote the vector of returns for N stocks at time t. A VAR(F) model expresses the return vector at time $t+1$ as a linear combination of its F previous values:

$$R_{t+1} = B_1 R_t + B_2 R_{t-1} + \cdots + B_F R_{t+1-F}$$

where $B_1, \ldots, B_F \in \mathbb{R}^{N \times N}$ are matrices of coefficients. By fitting a VAR model, we can simultaneously model the temporal dependencies of each stock as well as the cross-asset influences. This is particularly relevant here, where stock returns are often correlated with each other and respond to common latent factors.

Although we did not implement it in our project, the VAR model could have captured the joint behavior of stock returns and cross-sectional dependencies. However, we chose not to pursue this approach due to the relatively high dimensionality of the data compared to the available sample size, which could lead to overfitting or unreliable parameter estimates.

4 Benchmark

To measure the performance of our future model, it is a good idea to start by establishing a benchmark to have a performance reference.

We also wondered about the reasoning behind the organizers' choice to limit the number of factors to $F = 10$. While the decision was not explicitly justified, this constraint likely reflects a balance between model expressiveness and generalization. Limiting the number of learned factors helps control model complexity and mitigates the risk of overfitting, especially given the relatively small number of training assets and time points. Moreover, as observed in our principal component analysis (PCA), a small number of directions often captures a substantial portion of the return variance, which supports the idea that a low-dimensional factor space may be sufficient to extract meaningful predictive signals. Nonetheless, it would be interesting to investigate how the performance metric evolves when varying the number of factors beyond this fixed choice.

4.1 QRT's approach

A possible "brute force" approach to tackle this problem is to generate orthonormal vectors $A_1, \dots, A_{10} \in \mathbb{R}^{250}$ at random and fit β on the training data set using linear regression. After repeating this operation many times, we can just select the best result from these attempts. This is the idea of the QRT benchmark. More precisely, the QRT benchmark strategy to beat is to repeat $N_{iter} = 1000$ times the following operations:

- Sample a 250×10 matrix M with iid Gaussian $N(0, 1)$ entries.
- Apply the Gram-Schmidt algorithm to the columns of M to obtain a matrix $A = [A_1, \dots, A_{10}]$ with orthonormal columns.
- Use the columns of A to build the factors and then take β with minimal mean square error on the training dataset.
- Compute the metric on the training data.

The model parameters to choose are the model parameters (A, β) that maximize this metric.
One can note that the orthonormality condition for the vectors $A_1, \dots, A_F$ reads $A^T A = I_F$ for the matrix $A := [A_1, \dots, A_F]$. The space of matrices satisfying this condition is known as the Stiefel manifold, a generalization of the orthogonal group, and one can show that the previous procedure generates a sample from the uniform distribution on this (compact symmetric) space.

5 Supervised Stiefel Factor Model

5.1 Motivation and Design

In contrast to the benchmark approach which relies on randomly sampling factor directions from the space of orthonormal matrices, our strategy aims to learn these directions directly from the data, in a supervised way. The key idea is to leverage the fact that linear combinations of past returns can encode predictive signals, provided that the transformation is well-chosen. Rather than searching randomly, we aim to find the most predictive linear combinations under structural constraints.

We model the return at time $t + 1$ as:

$$S_{t+1} := \sum_{\ell=1}^{F} \beta_\ell F_{t,\ell} \quad \text{with} \quad F_{t,\ell} := \sum_{k=1}^{D} A_{k\ell} R_{t+1-k}$$

This can be rewritten as:

$$S_{t+1} = \sum_{\ell=1}^{F} \beta_\ell \left(\sum_{k=1}^{D} A_{k\ell} R_{t+1-k} \right) = \sum_{k=1}^{D} \left(\sum_{\ell=1}^{F} A_{k\ell} \beta_\ell \right) R_{t+1-k}$$

Mathematically, we represent each factor $F_{t,\ell}$ as a linear combination of past returns using a temporal filter $A_\ell \in \mathbb{R}^D$. Collecting the filters into a matrix $A \in \mathbb{R}^{D\times F}$, the predicted signal at time t becomes:

$$S_{t+1} = \mathbf{R_t} A\beta$$

where $\beta \in \mathbb{R}^F$ are scalar weights learned via regression, and $\mathbf{R_t} \in \mathbb{R}^{N\times D}$ the matrix of returns of the 250 first days.

To ensure diversity and avoid redundancy in the learned factors, we require the matrix A to have orthonormal columns, i.e., $A^T A = I$. This condition places A on the Stiefel manifold, denoted $V_F(\mathbb{R}^D)$, which is the set of all F-tuples of orthonormal vectors in $\mathbb{R}^D$. Concretely, this manifold is defined as:

$$V_F(\mathbb{R}^D) = \left\{ A \in \mathbb{R}^{D\times F} \,\middle|\, A^\top A = I_F \right\}.$$

It generalizes the notion of the orthogonal group. We then pose the following optimization problem:

$$\max_{A\in V_F(\mathbb{R}^D)} \text{Metric}(A, \beta^*(A)), \quad \text{where} \quad \beta^*(A) = \arg\min_\beta \|Y - XA\beta\|^2,$$

with X the design matrix of lagged returns and Y the vector of returns to predict.

To prepare the data for prediction, we reshaped the return matrix into a data frame indexed by (date, stock), where each row corresponds to a specific stock on a given date. The columns represent the 250 lagged daily returns of that stock, with time lags ranging from 1 to 250 days in the past. This representation transforms the time series data into a standard design matrix, allowing each (date, stock) pair to be treated as an independent training example with a fixed-length feature vector of historical returns.
So we constructed a design matrix $X \in \mathbb{R}^{N(T-D+1)\times D}$, where $T = 753$ is the total number of dates and $D = 250$, the number of time lags.
Each row of X corresponds to a unique (date, stock) pair and consists of the past D lagged daily returns for that stock:

$$\mathbf{x}_{t,s} = \begin{bmatrix} R^{(s)}_{t-1} & R^{(s)}_{t-2} & \cdots & R^{(s)}_{t-D} \end{bmatrix} \in \mathbb{R}^D.$$

Stacking these vectors for all stocks $s \in \{1, \ldots, N\}$ and time points $t \in \{D, \ldots, T\}$, we obtain:

$$X = \begin{bmatrix} \mathbf{x}_D \\ \mathbf{x}_{D+1} \\ \vdots \\ \mathbf{x}_T \end{bmatrix} \in \mathbb{R}^{N(T-D)\times D}.$$

where

$$\mathbf{x}_t = \begin{bmatrix} R_{t-1} & R_{t-2} & \cdots & R_{t-D} \end{bmatrix} \in \mathbb{R}^{N\times D}.$$

5.2 Optimization Problem Resolution

To solve this optimization problem, we apply the Trust-Region method on the Stiefel manifold, which performs second-order optimization while ensuring the orthonormality constraint is preserved at every iteration. The loss function corresponds to the negative of the Metric function, i.e the negative mean cosine similarity between predicted and actual return vectors:

$$\text{Loss}(A, \beta) := -\frac{1}{504} \sum_{t=250}^{753} \frac{\langle \tilde{S}_t, \tilde{R}_t \rangle}{\|\tilde{S}_t\| \|\tilde{R}_t\|} = -\frac{1}{504} \sum_{t=250}^{753} \cos(\tilde{S}_t, \tilde{R}_t)$$

5.3 Optimization Algorithm

To solve this problem, we use the `pymanopt` library, which enables optimization over matrix manifolds.

Our algorithm proceeds as follows:

1. Construct the lagged return matrix $X \in \mathbb{R}^{N(T-D+1)\times D}$ from the training data.
2. For a given A, compute the factor matrix XA, then estimate $\beta \in \mathbb{R}^F$ via least-squares regression.
3. Reshape and normalize the predictions, then compute the average overlap with true returns.
4. Optimize over $A \in V_{10}(\mathbb{R}^{250})$ to maximize the overlap.

5.4 Results on Training and Test Data

Our model achieved a cosine similarity score of:

$$\text{Metric}_{\text{train}} = 0.1479 \qquad \text{and} \qquad \text{Metric}_{\text{test}} = 0.0833$$

In comparison, the QRT benchmark reached only 0.0458 on the training set and 0.0354 on the test set. This confirms that a data-driven, supervised approach with geometric constraints significantly outperforms random sampling strategies. Despite using only linear operations, the model succeeds in extracting meaningful temporal patterns from noisy financial data.

5.5 Visual Interpretation of Learned Structure

Figure 8 presents the evolution of the daily cosine similarity between predictions and true returns across the training period. The values remain mainly positive, indicating the presence of a stable predictive signal.

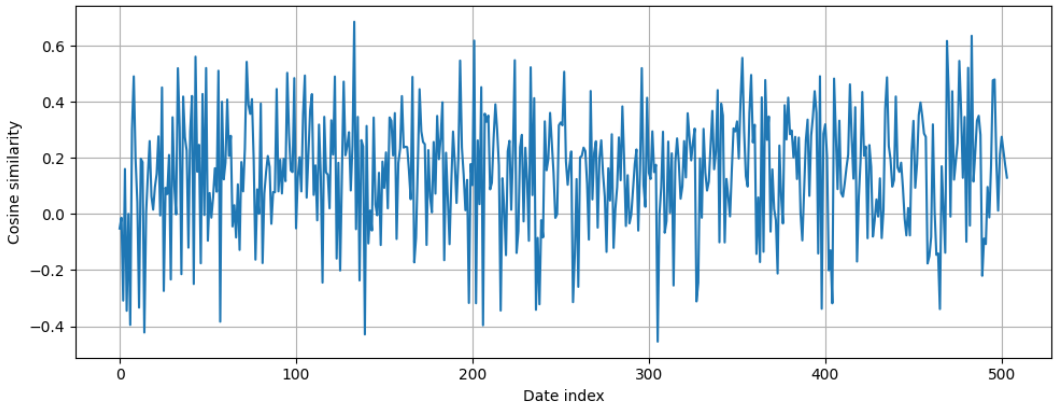

Figure 8: Temporal overlap between predictions and true returns on the training set

Figure 9 displays a heatmap of the learned matrix A. Each column corresponds to a learned factor, while rows represent the lag index (from most recent return at the top to oldest at the bottom). The color gradient highlights the contribution of each lag to the factor: red for positive influence, blue for negative. This visualization allows us to distinguish between short-term momentum filters and long-horizon reversal structures.

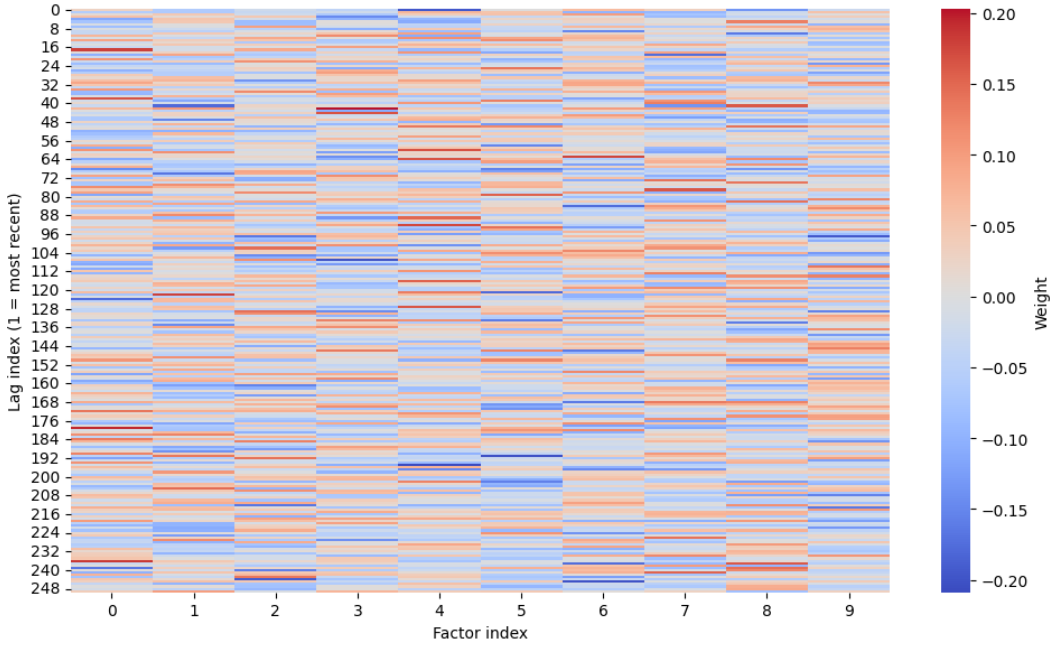

Figure 9: Heatmap of the learned factor matrix A

In Figure 10, we show the learned factor weights β. These values indicate how strongly each temporal factor contributes to the final prediction.

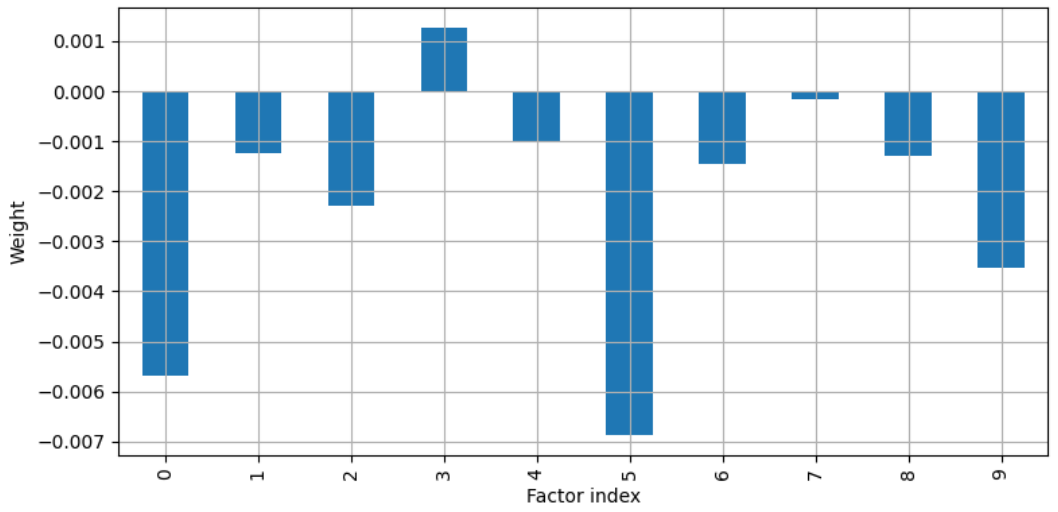

Figure 10: Bar plot of the learned weights β

Finally, Figure 11 provides the distribution of cosine similarities over time. The histogram is clearly skewed toward positive values, confirming that our predictions are aligned with real returns more often than not.

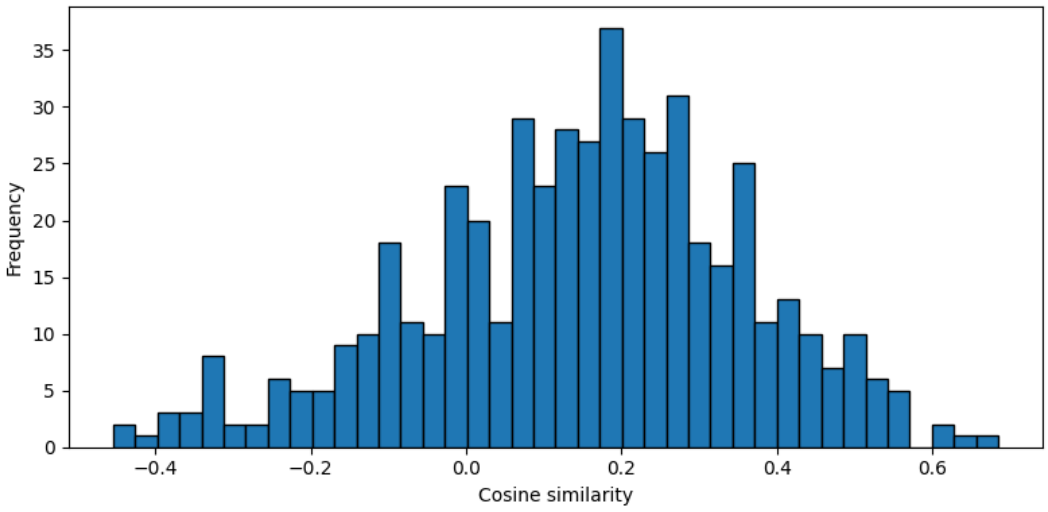

Figure 11: Distribution of prediction overlaps over time

5.6 Discussion

This model demonstrates that supervised learning of orthonormal temporal factors offers a robust and interpretable alternative to random sampling. By optimizing directly on the Stiefel manifold, we avoid redundancy and ensure stability in the learned directions. The significant improvement over the benchmark validates this geometric approach.

Further extensions could include the addition of regularization, initialization from financial priors, or hybridization with nonlinear models such as LSTMs or transformers. Nonetheless, even in its current linear form, the model performs remarkably well and lays the groundwork for future exploration.

6 Conclusion

In this project, we addressed the problem of predicting stock market returns using a data-driven factor model. By formulating the task as an optimization over the Stiefel manifold, we designed a supervised model that learns orthonormal temporal filters directly from historical returns. The model achieved a training score of 0.1479 and a test score of 0.0833, significantly outperforming the benchmark provided by QRT, whose respective scores were 0.0458 and 0.0354.

These results demonstrate the benefits of combining linear factor modeling with geometric optimization techniques. Although our approach remains relatively simple and interpretable, it captures predictive structure more effectively than random projections. As of the final submission, our team ranks 119^{th} on the challenge leaderboard. Future work may include more advanced architectures or multi-factor extensions to further improve performance in out-of-sample settings.