UCLA
UCLA Electronic Theses and Dissertations

Title

Decision-Focused Fine-Tuning for Illiquid Asset Portfolio Optimization

Permalink

https://escholarship.org/uc/item/1jz8r94z

ISBN

9798190919806

Author

WU, YUE

Publication Date

2026-09-08

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA

Los Angeles

Decision-Focused Fine-Tuning for Illiquid Asset Portfolio Optimization

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Applied Statistics and Data Science

by

Yue Wu

2026

© Copyright by

Yue Wu

2026

ABSTRACT OF THE THESIS

Decision-Focused Fine-Tuning for Illiquid Asset Portfolio Optimization

by

Yue Wu

Master of Applied Statistics and Data Science

University of California, Los Angeles, 2026

Professor George Michailidis, Chair

This thesis studies the relationship between return prediction and portfolio optimization for illiquid assets. Traditional predict-then-optimize models first predict asset returns and then use those predictions as inputs to a portfolio optimizer. However, a model with lower prediction error does not always lead to better portfolio decisions. Due to the noisy signals, limited data, and liquidity-related frictions, this issue is especially serious for illiquid assets, where small prediction errors may lead to different allocation decisions.

To address this problem, this thesis develops a Decision-Focused Fine-Tuning (DFF) framework based on a pretrained return prediction model. In this framework, a correction layer is inserted between the prediction model and the portfolio optimizer. This layer makes controlled adjustments to the original return predictions, and the adjustment size is controlled

by the parameter ϵ, which bounds the maximum relative adjustment applied to the original predictions. Then the adjusted predictions will be passed into a differentiable Markowitz-style portfolio optimization layer, and the correction layer is trained to reduce downstream decision regret.

Experiments are conducted on illiquid stock portfolios, corporate bond portfolios, and mixed stock–bond portfolios. The results show that decision-focused fine-tuning can improve downstream optimization quality compared with a traditional two-stage baseline. The experiments also demonstrate that different settings of ϵ lead to different levels of improvement, highlighting the importance of this parameter in balancing predictive stability and decision quality. Overall, this thesis shows that the Decision-Focused Fine-Tuning framework provides a practical approach for linking return prediction with portfolio optimization in illiquid financial markets.

The thesis of Yue Wu is approved.

Nicolas Christou

Yingnian Wu

George Michailidis, Committee Chair

University of California, Los Angeles

2026

Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my deepest gratitude to my thesis advisor, Professor George Michailidis. His guidance, patience, encouragement, and thoughtful support helped shape this thesis at every stage and gave me the confidence to bring this work to completion.

I am also sincerely grateful to the members of my thesis committee for their time, expertise, and valuable feedback. Their comments and suggestions have greatly strengthened this work.

I would like to thank the University of California, Los Angeles, for providing an inspiring academic environment and valuable opportunities that have helped me grow as a more skilled researcher.

I am also deeply grateful to my friends, who brought constant encouragement during difficult moments and joy throughout my academic journey.

Finally, I owe my most profound gratitude to my parents. For your sacrifices, your love, and your unwavering faith in me, no words of thanks can ever be enough. This journey would not have been possible without your constant companionship and support, and this achievement belongs to you as much as it does to me.

Chapter 1

Introduction

1.1 Research Motivation

In financial markets, portfolio optimization has always been an important problem in decision-making. In a typical investment process, investors first estimate future asset returns and then use these estimates to decide how to allocate their investments across different assets. This traditional process is often called the predict-then-optimize framework, which is simple and has been widely used for many years because each part can be handled separately [1, 2].

Although this separate method can achieve effective results, it also creates some issues. The goal of a return prediction model is usually to reduce prediction error, such as mean squared error (MSE) and mean absolute error (MAE). In the traditional view, we use a smaller prediction error to make the predicted returns closer to the observed returns on average. However, this result does not always mean that the final portfolio decision will be better [1–3]. In portfolio optimization, even small prediction errors can lead to different asset weights, especially when the optimizer is sensitive to the input estimates. Therefore, a prediction model with a small prediction error may not produce the best allocation decision.

This issue may be less significant for more liquid assets, but it becomes more serious when the assets are illiquid. Illiquid assets are different from traditional assets; they often have

noisier signals, less frequent trading, limited historical observations, and stronger liquidity-related constraints. Return prediction is much more difficult due to these characteristics, and the allocation decision is sometimes more sensitive to estimation errors. In order to solve this problem, the optimization goal should not only be to make accurate predictions, but also to make the predictions more suitable for the final portfolio optimization problem.

The mismatch between prediction accuracy and portfolio decision quality is common in practice. Thus, this thesis focuses on illiquid assets and studies how return prediction can be better connected with portfolio optimization. Instead of treating prediction and optimization as two separate steps, this thesis considers a decision-focused fine-tuning framework that adjusts predicted returns in a controlled way so that they can lead to better downstream portfolio decisions [2, 4].

1.2 Thesis Contributions

To address the limitations of traditional models, this thesis proposes a Decision-Focused Fine-Tuning (DFF) framework for portfolio optimization, motivated by recent work on decision-focused fine-tuning [4]. The advantage of this framework is that it connects the prediction stage and the optimization stage, which can make the allocation decision more reasonable. Figure 1.1 provides an overview of the proposed DFF workflow.

The first contribution of this thesis is to introduce a correction layer between a prediction model and the portfolio optimizer. This correction layer adjusts the return predictions and makes them better aligned with the final portfolio decision. The second contribution is that this layer controls the size of the adjustment through the parameter ϵ. This constraint bounds the maximum relative adjustment applied to the original predictions. The third contribution is to train the correction layer using downstream decision regret. After the predictions are adjusted, they are passed into a Markowitz-style mean-variance portfolio optimization layer, and the correction layer is updated according to the quality of

the resulting portfolio allocation. Finally, this thesis conducts experiments on illiquid assets, including stock portfolios, corporate bond portfolios, and mixed stock–bond portfolios. The experiments compare the DFF framework with a traditional model and study how different values of ϵ affect portfolio performance.

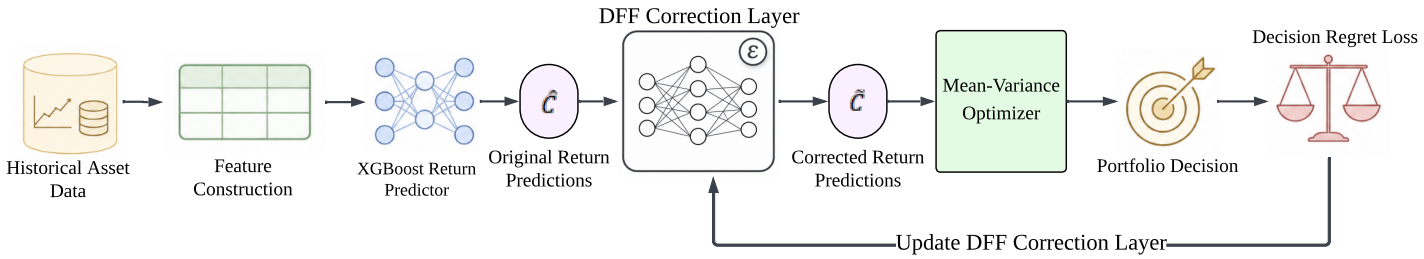

Figure 1.1: Workflow of the proposed DFF framework. A fixed XGBoost return predictor generates original return predictions, which are adjusted by the DFF correction layer and passed into a Markowitz-style mean-variance optimizer. The decision regret loss updates only the correction layer.

Chapter 2

Background and Related Work

2.1 Predict-then-Optimize and Portfolio Optimization

In traditional financial markets, the predict-then-optimize model is a common framework that has been used in many areas [1, 5]. In this framework, the first stage uses a predictive model to estimate uncertain future asset returns. In the second stage, portfolio optimization uses the predicted returns as inputs to make allocation decisions.

Let X_t be the feature information available at time t. The feature information may include historical returns, volatility, trading volume, and other market or liquidity-related variables. The return prediction model can be defined as

$$\hat{c}_t = f_\phi(X_t), \tag{2.1}$$

where $\hat{c}_t$ is the predicted return vector and f_ϕ is the prediction model with parameters ϕ. In the traditional two-stage prediction model, the training goal is usually to reduce prediction error. After the predicted returns are produced, they are passed into a portfolio optimizer.

The mean-variance model is a widely used optimization framework [6, 7]. The main idea of this model is to balance expected return and portfolio risk. Given the predicted return

vector $\hat{c}_t$ and the covariance matrix Σ_t, the optimizer chooses portfolio weights by solving

$$\begin{aligned} w_t = \arg\max_{w} \quad & \hat{c}_t^\top w - \gamma w^\top \Sigma_t w \\ \text{s.t.} \quad & \mathbf{1}^\top w = 1, \\ & w \geq 0. \end{aligned} \tag{2.2}$$

Here, w_t represents the portfolio allocation, Σ_t captures the risk structure among assets, and γ controls the trade-off between return and risk.

This framework is simple and intuitive, but the prediction model and the optimizer have different evaluation metrics. The prediction model is usually evaluated by accuracy, while the optimizer is evaluated by the quality of the final portfolio decision [1, 2, 7]. These two goals are related, but sometimes accurate prediction results can still lead to poor portfolio weights. This issue is especially significant when the optimizer is sensitive to small changes. A small difference in the predicted return of one asset may change its relative attractiveness and lead to a different allocation. Therefore, the final portfolio decision depends not only on the overall prediction error, but also on how the prediction errors interact with the optimization model.

For this reason, two-stage models may not always produce the best allocation decisions in complex financial markets. In fact, this is a two-part model, and it motivates methods that connect the prediction stage more directly with the optimization stage by considering whether the predictions lead to better portfolio decisions.

2.2 Decision-Focused Learning and Fine-Tuning

Unlike traditional two-stage models, decision-focused learning reduces the mismatch between prediction accuracy and decision quality by evaluating predictions through their impact on the final decision [2, 3, 7]. In this thesis, the decision-focused idea is implemented through a fine-tuning framework rather than by retraining the whole prediction model.

The prediction model, XGBoost, first generates the original predicted return vector $\hat{c}_t$ [8]. Then, a correction layer is added between the prediction model and the portfolio optimizer [4]. This correction layer adjusts the original prediction and produces a corrected return vector $\tilde{c}_t$.

The size of the correction is controlled by the parameter ϵ. This parameter bounds the maximum relative adjustment applied to the original predictions. A smaller ϵ allows only limited adjustment, while a larger ϵ gives the correction layer more flexibility. However, a very large ϵ may also make the corrected predictions less stable. Therefore, ϵ controls the balance between prediction stability and decision improvement.

After producing the corrected prediction $\tilde{c}_t$, the optimizer generates portfolio weights based on the Markowitz-style mean-variance structure. To evaluate the quality of this decision, the model compares it with an oracle decision, which is obtained using the realized return vector c_t. Decision regret measures the difference in realized portfolio objective between the decision based on the corrected prediction and this oracle benchmark. A smaller regret indicates that the resulting portfolio decision is closer to the oracle in terms of realized objective value.

The key point of the DFF framework is that the pretrained XGBoost predictor remains fixed. During training, only the correction layer is updated. This keeps the original prediction model stable while allowing the final portfolio decision to be improved through decision-focused training.

2.3 Illiquid Assets and Liquidity Measures

In financial markets, illiquid assets are assets that are difficult to trade quickly. Compared with highly liquid assets, illiquid assets usually have lower trading frequency, fewer market participants, larger trading frictions, and noisier price signals. These characteristics are important in portfolio construction because they affect the reliability of return signals and

the feasibility of trading decisions. For stocks, low liquidity is often related to low trading volume and large price movements relative to trading activity. For corporate bonds, since many bonds do not trade every day, liquidity is more closely related to trading frequency, trading gaps, and the amount of par value traded [9]. This thesis focuses on illiquid stocks and corporate bonds because prediction errors may have stronger effects on portfolio decisions in low-liquidity settings.

Chapter 3

Data and Experimental Design

3.1 Data Overview

This study contains three portfolio settings. The first setting is a stock-only portfolio, which is constructed from a Russell 2000-like stock universe. The final stock experiment uses 30 selected illiquid stocks based on liquidity-related measures. The second setting is a corporate bond-only portfolio, which is constructed from TRACE corporate bond transaction data [10]. The final bond experiment also uses 30 selected illiquid corporate bonds based on trading activity and trading gap measures. The third setting is a mixed stock–bond portfolio, which combines the separately constructed stock and bond universes into a single cross-asset portfolio.

For each portfolio setting, the raw data are transformed into model-ready instances for return prediction and portfolio optimization. Stocks and bonds are processed separately because they have different trading patterns. After processing the stock and bond datasets, the dates shared by both datasets are matched to create the mixed stock–bond portfolio.

3.2 Stock Universe Construction

We construct the stock portfolio from the Russell 2000-like stock universe because it contains many small-cap stocks, which are more likely to have lower liquidity than large-cap stocks. The stock data contain daily open price, high price, low price, close price, adjusted close price, and trading volume. The first step is to collect the ticker list, and then we download the daily stock price data from 2018 to 2022. After downloading the raw data, we clean the data by removing invalid rows, missing prices, non-positive adjusted prices, and non-positive trading volume. We also remove stocks with very low prices by requiring the median adjusted price to be at least 5 dollars. In addition, each selected stock is required to have a complete trading history over the sample period, so that the final stock panel is aligned across all assets.

After the basic data cleaning, daily returns and dollar trading volume are computed for each stock. The daily return is defined as

$$r_{i,t} = \frac{P_{i,t} - P_{i,t-1}}{P_{i,t-1}}, \tag{3.1}$$

where $P_{i,t}$ is the adjusted closing price of stock i at time t. The dollar trading volume is computed as

$$DVOL_{i,t} = Close_{i,t} \times Volume_{i,t}. \tag{3.2}$$

The stock illiquidity measure is based on the Amihud-style illiquidity measure [11, 12]:

$$ILLIQ_{i,t} = \frac{|r_{i,t}|}{DVOL_{i,t}}. \tag{3.3}$$

A higher value of $ILLIQ_{i,t}$ means that the stock has a larger price movement relative to its trading volume, which indicates lower liquidity. To reduce the influence of extreme values, the illiquidity measure is winsorized within each stock at the 1st and 99th percentiles.

The final stock liquidity ranking is based on two components, average illiquidity and

average dollar volume. For each stock, the average ILLIQ and average dollar volume are computed over the sample period. Stocks with higher average ILLIQ are ranked as less liquid, while stocks with lower average dollar volume are also ranked as less liquid. These two rankings are added together to form a composite illiquidity score. Stocks with smaller composite scores are treated as more illiquid. As shown in Figure 3.1, the top-ranked illiquid group has higher average ILLIQ and lower average dollar volume, which supports the use of the composite liquidity ranking.

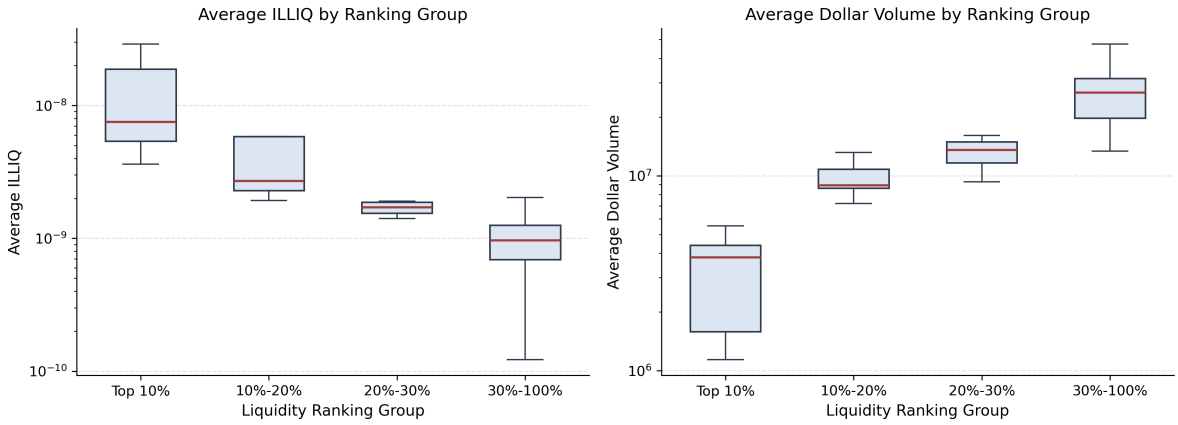

Figure 3.1: Stock liquidity measures across illiquidity ranking groups. The top-ranked illiquid stocks have higher average ILLIQ and lower average dollar volume, supporting the stock liquidity screening procedure.

Based on this liquidity ranking, we choose the top 30 illiquid stocks for the stock-only experiment. This choice keeps the portfolio size large enough to represent a diversified illiquid stock universe and the optimization problem computationally manageable.

After the stock universe is selected, the data are transformed into model-ready instances for the DFF experiment. Each instance corresponds to one trading date and contains asset-level features, realized future returns, and a covariance matrix. The asset-level features include lagged returns, rolling return statistics, momentum measures, volume-related features, dollar-volume features, and illiquidity-related features. The realized future return is defined over a 21-trading-day horizon.

3.3 Corporate Bond Universe Construction

The bond-only portfolio is constructed from TRACE corporate bond transaction data. The sample period is also from 2018 to 2022, which is consistent with the stock experiment.

The raw TRACE data are cleaned at the transaction level by keeping normal published trades and removing observations with missing or non-positive prices and trading volume. Each bond is identified by its CUSIP, and the cleaned trades are aggregated into a daily CUSIP-level panel. For each CUSIP-date pair, we compute daily trading volume, volume-weighted average price, volume-weighted average yield, trade count, and daily return. Since many corporate bonds trade infrequently and bond liquidity is closely related to trading activity and trading gaps [9, 13, 14], we require each candidate bond to have at least 80 trading days and 12 active months during the sample period.

The bond illiquidity ranking is based on trading activity and trading gap measures. For each remaining CUSIP, we compute average monthly trade count, average monthly trade days, zero-trade ratio, average days since last trade, and average daily par volume. Bonds with fewer trades, fewer trading days, higher zero-trade ratios, longer trading gaps, and lower par volume are considered less liquid. These measures are then combined into a composite bond illiquidity score. Bonds with weaker trading activity and longer trading gaps receive higher illiquidity scores. Similar to the stock experiment, we also select the top 30 illiquid corporate bonds for the bond-only experiment.

In Figure 3.2, the top-ranked illiquid bonds show lower trading activity, higher zero-trade ratios, longer trading gaps, and lower average par volume, which supports the bond liquidity screening procedure.

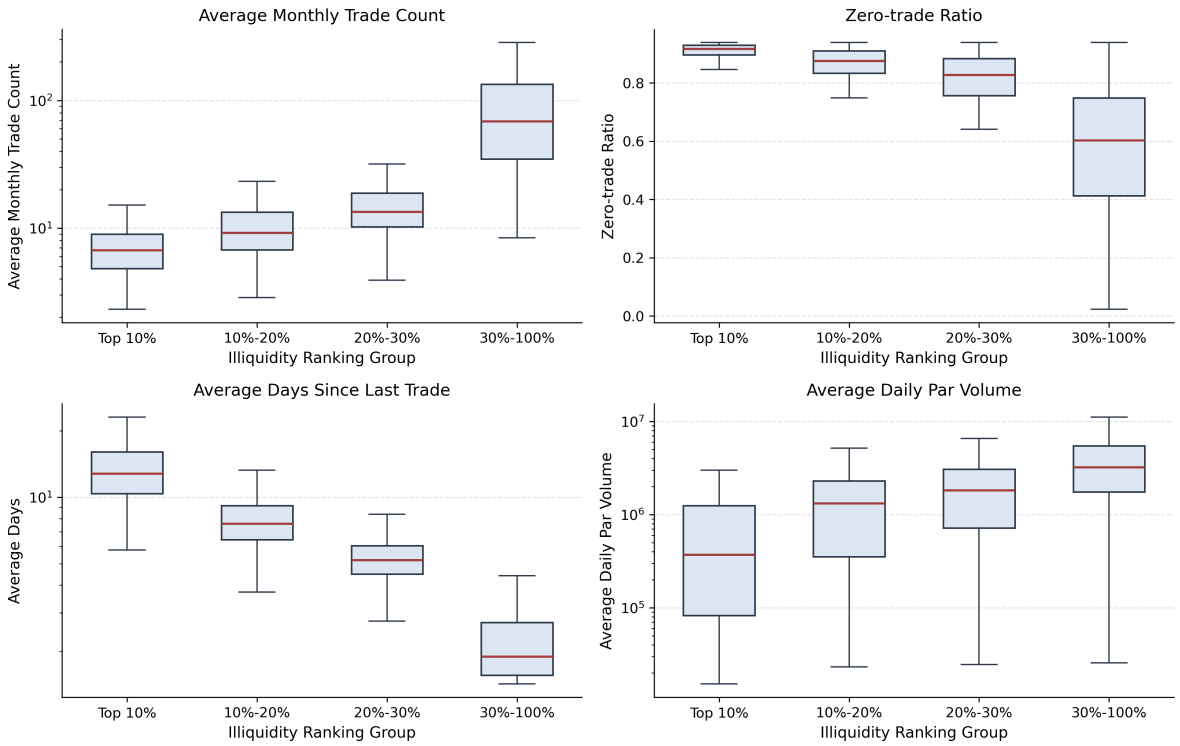

Figure 3.2: Corporate bond liquidity measures across illiquidity ranking groups. The top-ranked illiquid bonds show lower trading activity, higher zero-trade ratios, longer trading gaps, and lower average par volume, supporting the bond screening procedure.

Recent studies also show that machine learning can be useful for studying corporate bond illiquidity and return patterns [9, 15]. After the bond universe is selected, the data are transformed into model-ready instances for the DFF experiment. Since many bonds do not trade every day, bond prices are forward-filled after the first valid observation, while trading volume and trade count are set to zero on no-trade days. The bond features include lagged returns, rolling return statistics, momentum measures, volume-related features, trade-count features, zero-trade indicators, and days-since-trade measures. The realized future return is defined over a 21-business-day horizon. Each bond instance also contains a covariance matrix for the downstream portfolio optimizer.

3.4 Mixed Stock–Bond Portfolio Construction

After constructing the stock and bond datasets separately, we create the mixed stock–bond portfolio by combining the selected stock and bond universes, which contains 60 assets in total. Combining stocks and corporate bonds creates a setting where the optimizer allocates across assets with different liquidity and trading characteristics [12, 13].

Since the stock and bond datasets may not have exactly the same available dates, the first step is to match the common dates shared by both datasets. Only dates that appear in both the stock instance list and the bond instance list are kept for the mixed portfolio experiment. This ensures that each mixed instance contains stock and bond information on the same decision date.

For the features, the mixed dataset places stock and bond assets into one common feature format. Stock assets keep their own stock features, and the bond feature columns are set to zero. Bond assets keep their own bond features, and the stock feature columns are set to zero. The dataset also adds two asset-type indicators so that the model can distinguish stocks from bonds. These indicators are defined as

$$(\text{is\_stock}, \text{is\_bond}) = \begin{cases} (1,0), & \text{for stock assets,} \\ (0,1), & \text{for bond assets.} \end{cases} \tag{3.4}$$

The realized future return vector is constructed by combining the stock future returns and bond future returns. In addition, the asset names are labeled with prefixes so that the two asset types can be clearly distinguished.

3.5 Final Experimental Dataset

After the stock, bond, and mixed stock–bond universes are constructed, each dataset is prepared for the portfolio optimization experiment. Table 3.1 summarizes the three portfolio

settings.

Table 3.1: Summary of experimental portfolio settings.

Portfolio Setting	**Asset Universe**	**No. of Assets**	**Selection Method**
Stock-only	Russell 2000-like stocks	30	Selected using high average ILLIQ and low average dollar volume.
Bond-only	TRACE corporate bonds	30	Selected using low trading activity, high zero-trade ratio, long trading gaps, and low par volume.
Mixed stock–bond	Selected stocks and selected bonds	60	Constructed by combining the 30 selected stocks and 30 selected bonds on common dates.

Each model-ready instance corresponds to one decision date and contains an asset-level feature matrix, a realized future return vector, and a covariance matrix. These components are used for return prediction, portfolio evaluation, and downstream Markowitz-style optimization.

The realized future return is defined over a 21-trading-day horizon for stocks and a 21-business-day horizon for corporate bonds. In the mixed stock–bond setting, the stock and bond instances are matched on their common dates before being combined. Therefore, each mixed instance contains both stock and bond information on the same decision date.

3.6 Data Split and Leakage Control

In this study, the experiments use a time-ordered split instead of a random split. Randomly shuffling observations across dates may introduce look-ahead bias, because future market information could indirectly affect the training set. For all portfolio settings, the sample period is from 2018 to 2022. Observations from 2018 to 2021 are used for training and validation, while observations from 2022 are held out for final testing. Within the pre-2022 period, the earlier 90% of observations are used as the training set, and the last 10% are used as the validation set. Table 3.2 summarizes the time-based split used in the experiments.

Table 3.2: Time-based data split used in the experiments.

Split	Time Period	Purpose
Training	Earlier 90% of 2018–2021 observations	Fit the XGBoost return predictor and train the DFF correction layer.
Validation	Last 10% of 2018–2021 observations	Select hyperparameters such as ϵ and monitor model performance.
Testing	2022 observations	Evaluate the final two-stage baseline and DFF method.

Because the realized return target uses a future 21-day horizon, observations close to the train-test boundary may contain overlapping return information across the split. To reduce this form of leakage, the experiments use a 21-day embargo around the boundary between the pre-2022 period and the 2022 test period.

Chapter 4

Methodology

4.1 Predict-Then-Optimize Baseline

This study chooses the traditional predict-then-optimize framework as the baseline method. For each decision date t, let X_t denote the asset-level feature matrix and let c_t denote the realized future return vector. The return prediction model produces an estimated return vector,

$$\hat{c}_t = M(X_t), \tag{4.1}$$

where $M(\cdot)$ is an XGBoost regression model, and XGBoost is used as the pretrained return predictor because it can handle nonlinear relationships and tabular financial features.

After the predicted return vector $\hat{c}_t$ is obtained, it is passed into a Markowitz mean-variance portfolio optimizer. This optimizer balances predicted return against portfolio variance, where the covariance matrix Σ_t represents the risk of asset returns. The baseline portfolio weights are computed by solving

$$w_t(\hat{c}_t) = \arg\min_{w} \left\{ -\hat{c}_t^{\top} w + \gamma w^{\top} \Sigma_t w \right\}, \tag{4.2}$$

subject to

$$\mathbf{1}^\top w = 1, \qquad w \geq 0. \tag{4.3}$$

Here, w is the portfolio weight vector, Σ_t is the covariance matrix, and γ is the risk-aversion parameter. The first term rewards higher predicted return, while the second term penalizes portfolio risk. The constraints require the portfolio to be fully invested and long-only.

This baseline is widely used and it separates prediction from optimization. The main goal of the XGBoost model is to reduce prediction error, while the final goal is to produce more reasonable portfolio decisions. These two goals are not always related, especially for illiquid assets.

4.2 Decision-Focused Fine-Tuning Framework

The proposed Decision-Focused Fine-Tuning framework adds a correction layer between the XGBoost predictor and the portfolio optimizer [2, 4, 16]. The XGBoost model first produces the original predicted return vector $\hat{c}_t$. Then, the correction layer adjusts this prediction and produces a corrected return vector $\tilde{c}_t$.

The correction can be viewed as

$$\tilde{c}_t = \hat{c}_t + \Delta_t, \tag{4.4}$$

where Δ_t is a learned adjustment. In the implementation, this adjustment is parameterized in a multiplicative form. For each asset, the corrected return is computed as

$$\tilde{c}_t = \phi_\theta(X_t, \hat{c}_t) \odot \hat{c}_t. \tag{4.5}$$

The function ϕ_θ is a small neural network correction layer. It produces an asset-level correction factor for the original XGBoost prediction.

To keep the corrected predictions close to the original predictions, the correction factor

is constrained by ϵ. In the implementation, this is done using a sigmoid transformation,

$$\phi_\theta(X_t, \hat{c}_t) = (1 - \epsilon) + 2\epsilon \cdot \sigma(h_\theta(X_t, \hat{c}_t)), \tag{4.6}$$

where $h_\theta(\cdot)$ is the neural network output before the sigmoid function and $\sigma(\cdot)$ is the sigmoid function. This construction guarantees that

$$\phi_\theta(X_t, \hat{c}_t) \in [1 - \epsilon, 1 + \epsilon]. \tag{4.7}$$

Therefore, ϵ controls the maximum relative adjustment size. When ϵ is small, the correction layer can only make limited changes to the original prediction. When ϵ is larger, the correction layer has more flexibility to improve downstream decisions.

During training, the XGBoost predictor is fixed. Only the correction layer parameters θ are updated. This keeps the original prediction model stable while allowing the final portfolio decision to be improved through decision-focused training.

4.3 Portfolio Optimization and Decision Regret

After the corrected prediction $\tilde{c}_t$ is obtained, it is used as the input to the same Markowitz mean-variance portfolio optimizer [6, 7]. The optimizer chooses the portfolio weights by solving

$$w_t(\tilde{c}_t) = \arg\min_w \left\{ -\tilde{c}_t^\top w + \gamma w^\top \Sigma_t w \right\}, \tag{4.8}$$

subject to the same constraints in Equation 4.3. The first term lets the portfolio choose assets with higher predicted returns, while the second term penalizes portfolio risk.

To evaluate the portfolio after the true future returns are observed, this thesis uses the realized Markowitz objective,

$$f(w; c_t, \Sigma_t) = -c_t^\top w + \gamma w^\top \Sigma_t w. \tag{4.9}$$

Here, c_t is the realized future return vector. Since this objective is minimized, a smaller value means a better portfolio decision.

For comparison, this thesis defines an oracle portfolio, which is the portfolio that would be chosen if the true future returns c_t were known in advance,

$$w_t^* = \arg\min_w \left\{ -c_t^\top w + \gamma w^\top \Sigma_t w \right\}, \tag{4.10}$$

subject to the same full-investment and long-only constraints. This oracle portfolio cannot be used in practice, because future returns are unknown at the decision time. However, it is a useful tool to evaluate the performance of the portfolio decision.

The decision regret is defined as the difference between the realized objective value of the DFF portfolio and the realized objective value of the oracle portfolio,

$$\mathcal{L}_{regret}(t) = f(w_t(\tilde{c}_t); c_t, \Sigma_t) - f(w_t^*; c_t, \Sigma_t). \tag{4.11}$$

If this regret is small, then the portfolio chosen using the corrected prediction is close to the oracle portfolio in terms of realized performance. If the regret is large, then the corrected prediction leads to a worse portfolio decision.

The correction layer is trained to minimize the average regret over all training dates,

$$\min_\theta \frac{1}{T} \sum_{t=1}^{T} \mathcal{L}_{regret}(t). \tag{4.12}$$

In this way, the correction layer is not trained only to make the return prediction more accurate. Instead, it is trained to make the corrected prediction more useful for the final portfolio optimization problem [17, 18].

4.4 Training and Evaluation

The training procedure has four main steps. First, the XGBoost model is trained to predict future asset returns from historical asset-level features. Second, the trained XGBoost model is fixed and used to generate original return predictions. Third, the DFF correction layer adjusts these predictions. Finally, following differentiable optimization layer methods [19, 20], the corrected predictions are passed into the differentiable Markowitz mean-variance optimizer, and the correction layer is trained using decision regret.

The two-stage baseline directly uses $\hat{c}_t$ in the optimizer, while the DFF method uses $\tilde{c}_t$. Therefore, the two portfolio decisions are

$$w_t^{twostage} = w_t(\hat{c}_t), \text{ and } w_t^{DFF} = w_t(\tilde{c}_t). \tag{4.13}$$

The first evaluation metric is prediction mean squared error. For the original two-stage prediction, it is defined as

$$\text{MSE}_{twostage} = \frac{1}{TN} \sum_{t=1}^{T} \sum_{i=1}^{N} (\hat{c}_{t,i} - c_{t,i})^2. \tag{4.14}$$

For the corrected DFF prediction, it is defined as

$$\text{MSE}_{DFF} = \frac{1}{TN} \sum_{t=1}^{T} \sum_{i=1}^{N} (\tilde{c}_{t,i} - c_{t,i})^2, \tag{4.15}$$

where N is the number of assets in the corresponding portfolio setting.

However, prediction error alone does not fully measure portfolio quality. Therefore, this thesis also evaluates decision regret for both the two-stage baseline and the DFF method. The average regret of the two-stage baseline is

$$\text{Regret}^{twostage} = \frac{1}{T} \sum_{t=1}^{T} \left[f(w_t^{twostage}; c_t, \Sigma_t) - f(w_t^*; c_t, \Sigma_t) \right]. \tag{4.16}$$

Similarly, the average regret of the DFF method is

$$\text{Regret}^{DFF} = \frac{1}{T}\sum_{t=1}^{T}\left[f(w_t^{DFF}; c_t, \Sigma_t) - f(w_t^*; c_t, \Sigma_t)\right]. \tag{4.17}$$

To make regret easier to compare across portfolio settings, this thesis also reports normalized decision regret. For the two-stage baseline, normalized decision regret is defined as

$$\text{NDR}^{twostage} = \frac{\sum_{t=1}^{T}\left[f(w_t^{twostage}; c_t, \Sigma_t) - f(w_t^*; c_t, \Sigma_t)\right]}{\sum_{t=1}^{T}|f(w_t^*; c_t, \Sigma_t)|}. \tag{4.18}$$

For the DFF method, normalized decision regret is defined as

$$\text{NDR}^{DFF} = \frac{\sum_{t=1}^{T}\left[f(w_t^{DFF}; c_t, \Sigma_t) - f(w_t^*; c_t, \Sigma_t)\right]}{\sum_{t=1}^{T}|f(w_t^*; c_t, \Sigma_t)|}. \tag{4.19}$$

By comparing $\text{Regret}^{twostage}$ with Regret^{DFF} and $\text{NDR}^{twostage}$ with NDR^{DFF}, the experiments directly test whether DFF improves portfolio decisions relative to the traditional two-stage baseline. The MSE metrics are also reported to show whether improvements in portfolio quality are aligned with improvements in return prediction accuracy.

In addition to these main metrics, this thesis also records diagnostic quantities such as the cosine similarity between $\widehat{c}_t$ and $\widetilde{c}_t$ and the change in prediction RMSE. These diagnostics are used to check whether the DFF correction remains close to the original prediction.

Chapter 5

Results

This chapter presents the test results for the stock-only, bond-only, and mixed stock-bond portfolio settings. The main focus is normalized decision regret (NDR), which directly measures the quality of the final portfolio decision relative to the oracle benchmark. For the main reported results, the correction strength ϵ is selected using validation-period regret, and the test period is used only for final out-of-sample evaluation.

5.1 Normalized Decision Regret Across Epsilon

Figure 5.1 reports the test normalized decision regret of the two-stage baseline and the DFF method across different values of ϵ. This figure is used as a sensitivity analysis for the correction size. The two-stage baseline directly uses the original XGBoost prediction $\hat{c}_t$, so its performance does not depend on ϵ. Therefore, its NDR is shown as a reference level in each portfolio setting. In contrast, DFF uses ϵ to control how strongly the correction layer can adjust the original prediction. The vertical marker in each panel indicates the value of ϵ selected using validation-period regret.

Across the three portfolio settings, DFF achieves lower NDR than the two-stage baseline at the validation-selected values of ϵ. The stock-only and mixed stock-bond settings

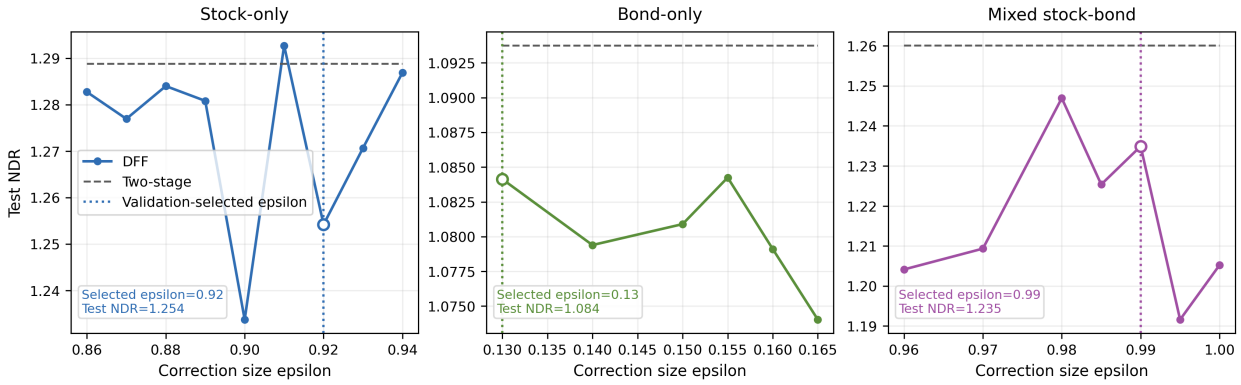

Figure 5.1: Normalized decision regret across different values of ϵ. Lower NDR indicates better realized portfolio decisions. The curves are shown as a sensitivity analysis, and the vertical marker indicates the value of ϵ selected using validation-period regret.

show clearer reductions in NDR, while the bond-only setting shows a smaller but still positive reduction. This suggests that the benefit of decision-focused fine-tuning varies across asset classes and portfolio settings.

5.2 Improvement Over the Two-Stage Baseline

To make the comparison more direct, Figure 5.2 reports the improvement of DFF over the two-stage baseline across different values of ϵ. Improvement is computed as the two-stage NDR minus the DFF NDR. Since lower NDR means a better portfolio decision, a positive improvement means that DFF has lower normalized decision regret than the two-stage baseline.

At the validation-selected values of ϵ, the improvement is positive in all three portfolio settings. The stock-only setting shows the largest improvement, followed by the mixed stock-bond setting and the bond-only setting. These results support the main idea of decision-focused fine-tuning: the goal is not only to produce accurate return predictions, but to produce predictions that lead to better downstream portfolio decisions.

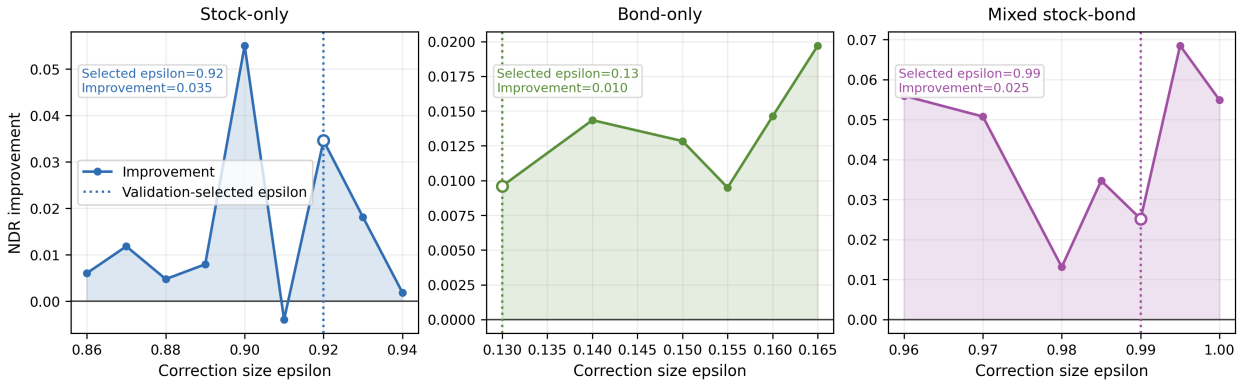

Figure 5.2: Improvement of DFF over the two-stage baseline across different values of ϵ. Positive values indicate that DFF has lower normalized decision regret. The vertical marker indicates the value of ϵ selected using validation-period regret.

5.3 Mixed Stock–Bond Portfolio Allocation

Figure 5.3 shows the average asset-class allocation in the mixed stock-bond test set under the validation-selected value of ϵ. The oracle portfolio is included only as an ex-post benchmark and is computed using realized future returns.

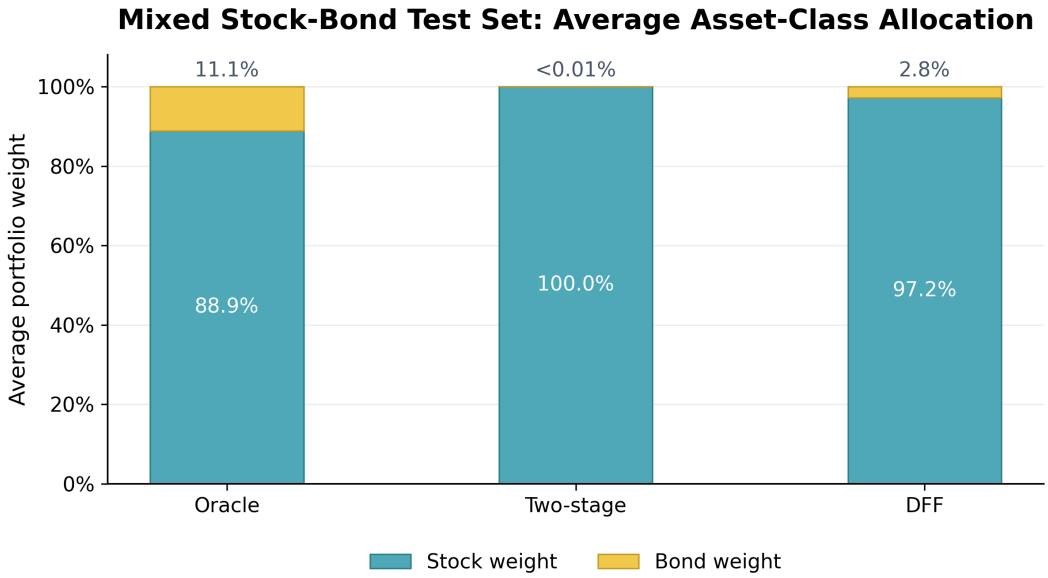

Figure 5.3: Average asset-class allocation on the mixed stock-bond test set. The oracle portfolio is an ex-post benchmark computed using realized future returns.

The two-stage baseline allocates almost all portfolio weight to stocks in the mixed setting. DFF also keeps most of the portfolio weight in stocks, but it assigns a positive average weight to bonds. Compared with the two-stage baseline, DFF shifts a small part of the portfolio from stocks to bonds, moving the allocation modestly toward the oracle benchmark. This suggests that the correction layer can affect not only prediction values but also the downstream portfolio composition.

5.4 Main Performance Summary

Table 5.1 summarizes the main test results. The table reports the value of ϵ selected using validation-period regret, together with the corresponding test-period NDR of the two-stage baseline, the test-period NDR of DFF, the improvement in NDR, and the prediction MSE of both methods.

Table 5.1: Main test performance summary

Setting	ϵ	$\text{NDR}_{twostage}$	NDR_{DFF}	Improvement	$\text{MSE}_{twostage}$	MSE_{DFF}
Stock-only	0.920	1.2888	1.2542	0.0346	0.0244	0.0358
Bond-only	0.130	1.0937	1.0841	0.0096	0.00235	0.00240
Mixed stock-bond	0.990	1.2601	1.2349	0.0252	0.0155	0.0211

The results show that DFF reduces normalized decision regret in all three portfolio settings under the validation-selected values of ϵ. The largest improvement appears in the stock-only setting, followed by the mixed stock-bond setting and the bond-only setting. This suggests that the benefit of DFF varies across asset classes and portfolio settings.

The MSE results show a different pattern. DFF does not necessarily reduce prediction MSE, and in these settings the corrected predictions have higher MSE than the original two-stage predictions. This does not contradict the main result. Instead, it shows that prediction accuracy and portfolio decision quality are not always aligned. In this thesis, the purpose of DFF is to improve the downstream portfolio decision, not simply to minimize prediction error.

Overall, the results suggest that decision-focused fine-tuning can improve portfolio optimization outcomes for illiquid assets. By training the correction layer through decision regret, DFF produces corrected predictions that are more useful for the Markowitz mean-variance optimizer.

Chapter 6

Conclusion

6.1 Main Findings

This thesis studies whether decision-focused fine-tuning can improve portfolio decisions for illiquid assets. Traditional two-stage methods first train a return prediction model and then use the predicted returns in a portfolio optimizer. However, a prediction that looks accurate under a standard prediction metric may not always lead to the best portfolio decision. The empirical results show that DFF reduces normalized decision regret relative to the two-stage baseline in the stock-only, bond-only, and mixed stock-bond settings. The improvement is largest in the stock-only setting, followed by the mixed stock-bond setting, while the bond-only setting shows a smaller but still positive improvement. These results suggest that the DFF correction layer can make the predicted returns more useful for the final portfolio optimization problem.

Overall, the findings support the main idea of the thesis: in portfolio optimization, the most useful prediction is not necessarily the one with the lowest prediction error, but the one that leads to a better investment decision. By training the correction layer with the downstream decision objective, DFF provides a practical way to connect machine learning prediction models with Markowitz mean-variance portfolio optimization for illiquid assets.

6.2 Limitations and Future Work

This thesis has several limitations. First, the experiments are based on the time period from 2018 to 2022. Although this period includes different market conditions, a longer sample period could provide a stronger test of whether the DFF framework remains effective across different market regimes. Second, the correction strength ϵ is evaluated on a finite grid. This grid is useful for studying the sensitivity of the DFF correction layer, but it is not intended to be an exhaustive global search over all possible correction strengths. Third, the current correction layer does not explicitly impose additional stability constraints on the direction of the correction. As a result, the corrected predictions may move away from the original predictor in ways that are useful for the portfolio objective but not always easy to interpret.

Future work can extend this thesis in several directions. First, the framework can be tested on a longer sample period and on additional asset classes. This would help evaluate whether the improvement from DFF is stable across different market environments and different types of illiquid assets. Second, future work could use a more systematic validation-based tuning procedure, such as nested time-series validation or rolling-window model selection, to choose ϵ and other hyperparameters more robustly. Finally, future extensions could study additional stability constraints on the correction layer, such as cosine-similarity bounds, to better control how far the corrected predictions move away from the original predictor.

References

[1] Adam N. Elmachtoub and Paul Grigas. Smart predict, then optimize. *Management Science*, 68(1):9–26, 2022. doi: 10.1287/mnsc.2020.3922. URL https://pubsonline.informs.org/doi/10.1287/mnsc.2020.3922.

[2] Jayanta Mandi, James Kotary, Senne Berden, Maxime Mulamba, Victor Bucarey, Tias Guns, and Ferdinando Fioretto. Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. *Journal of Artificial Intelligence Research*, 81: 1623–1701, 2024. doi: 10.1613/jair.1.15320. URL https://www.jair.org/index.php/jair/article/view/15320.

[3] Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):1658–1665, 2019. doi: 10.1609/aaai.v33i01.33011658. URL https://ojs.aaai.org/index.php/AAAI/article/view/3982.

[4] Jiaqi Yang, Enming Liang, Zicheng Su, Zhichao Zou, Peng Zhen, Jiecheng Guo, Wanjing Ma, and Kun An. Dff: Decision-focused fine-tuning for smarter predict-then-optimize with limited data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39 (25):26868–26876, 2025. doi: 10.1609/aaai.v39i25.34891. URL https://ojs.aaai.org/index.php/AAAI/article/view/34891.

[5] Dimitris Bertsimas and Nathan Kallus. From predictive to prescriptive analytics. *Management Science*, 66(3):1025–1044, 2020. doi: 10.1287/mnsc.2018.3253. URL https://pubsonline.informs.org/doi/10.1287/mnsc.2018.3253.

[6] Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952. doi: 10.1111/j.1540-6261.1952.tb01525.x.

[7] Junhyeong Lee, Inwoo Tae, and Yongjae Lee. Anatomy of machines for markowitz: Decision-focused learning for mean-variance portfolio optimization. *arXiv preprint arXiv:2409.09684*, 2024. URL https://arxiv.org/abs/2409.09684.

[8] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016. doi: 10.1145/2939672.2939785. URL https://dl.acm.org/doi/10.1145/2939672.2939785.

[9] Axel Cabrol, Wolfgang Drobetz, Tizian Otto, and Tatjana Puhan. Predicting corporate bond illiquidity via machine learning. *Financial Analysts Journal*, 80(3), 2024. doi: 10.1080/0015198X.2024.2350952. URL https://rpc.cfainstitute.org/research/financial-analysts-journal/2024/predicting-corporate-bond-illiquidity-via-machine-learning.

[10] FINRA. TRACE corporate bonds daily report card. https://www.finra.org/compliance-tools/report-center/trace/corporate-bonds-daily, 2026. Accessed 2026-06-16.

[11] Yakov Amihud. Illiquidity and stock returns: Cross-section and time-series effects. *Journal of Financial Markets*, 5(1):31–56, 2002. doi: 10.1016/S1386-4181(01)00024-6. URL https://www.sciencedirect.com/science/article/pii/S1386418101000246.

[12] Demetrio Lacava, Angelo Ranaldo, and Paolo Santucci de Magistris. Realized illiquidity. Research Paper 22-90, Swiss Finance Institute, 2023. URL https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4282541.

[13] Mahyar Kargar, Benjamin Lester, David Lindsay, Shuo Liu, Pierre-Olivier Weill, and Diego Zuniga. Corporate bond liquidity during the covid-19 crisis. *The Review of Financial Studies*, 34(11):5352–5401, 2021. doi: 10.1093/rfs/hhab063. URL https://academic.oup.com/rfs/article/34/11/5352/6279757.

[14] Sergey Chernenko and Adi Sunderam. Measuring the perceived liquidity of the corporate bond market. NBER Working Paper 27092, National Bureau of Economic Research, 2020. URL https://www.nber.org/papers/w27092.

[15] Daniele Bianchi, Matthias Buchner, and Andrea Tamoni. Bond risk premiums with machine learning. *The Review of Financial Studies*, 34(2):1046–1089, 2021. doi: 10.1093/rfs/hhaa062. URL https://academic.oup.com/rfs/article/34/2/1046/5843806.

[16] Michael Huang and Vishal Gupta. Decision-focused learning with directional gradients. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/907a9fb75a408f6c3a2ae1bf84c39e44-Abstract-Conference.html.

[17] Noah J. Schutte, Krzysztof S. Postek, and Neil Yorke-Smith. Robust losses for decision-focused learning. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 4868–4875, 2024. doi: 10.24963/ijcai.2024/538. URL https://www.ijcai.org/proceedings/2024/538.

[18] Haeun Jeon, Hyunglip Bae, Minsu Park, Chanyeong Kim, and Woo Chang Kim. Locally convex global loss network for decision-focused learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(25):26805–26812, 2025. doi: 10.1609/aaai.v39i25.34884. URL https://ojs.aaai.org/index.php/AAAI/article/view/34884.

[19] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural*

Information Processing Systems, volume 32, 2019. URL https://arxiv.org/abs/1910.12430.

[20] Bo Tang and Elias B. Khalil. Pyepo: A pytorch-based end-to-end predict-then-optimize library for linear and integer programming. *Mathematical Programming Computation*, 16:297–335, 2024. doi: 10.1007/s12532-024-00255-x. URL https://link.springer.com/article/10.1007/s12532-024-00255-x.