Transaction Costs and Short Term Price Signals: a Happy Marriage

Author:
Steven J. Supit

Supervisors:
S. H. L. C. G. Vermeulen
W. Tilgenkamp

April 24, 2020

Abstract

Traders tend to opt for long trade durations to minimize aggregate transaction costs over many trades. In this research I show that this is not necessarily optimal when short term stock price information is available. I show that price movement during a trade explains up to 60% of the variation of transaction costs. Furthermore, I show that we can use short term information to capitalize on this. By changing the duration or timing of trades according to machine learning predictions using logistic regression, neural nets, and LSTM's we can improve transaction costs up to 32 basis points per trade. This amounts to a net gain per transaction which generally does not consistently occur.

Contents

1 Introduction

In this thesis I conduct an analysis on the effect of price change on trading costs. I show how to improve the transaction costs of large institutional trades by changing the duration or the timing of the trade using otherwise unprofitable short term trading signals.

1.1 Findings and Contributions

Short term trading signals have shown to be difficult to put in practice due to the inherent lack of magnitude of the signal, causing transaction costs to be higher than the resulting returns. In this thesis I propose to use such signals not as buy or sell signals, but rather as a tool for timing trades. Large institutional trades take time to execute and are thus exposed to price changes during the trade. Using short term signals as a prediction of the direction of the underlying stock price we can thus change the duration of a trade, or delay the trade, to attain a more favorable trading price, and thus improve the transaction costs.

This idea follows the proposition of Perold (1988) and Kissell (2006) that the implementation shortfall of a portfolio can be divided into several parts, including transaction costs, timing costs, and exposure costs. Furthermore, I follow the idea of Israelov and Katz (2011) that short term information can be used to guide traders in the execution of trades. By allowing traders to act on short term information I argue that the goals of portfolio managers and traders are consolidated, which Kissell and Malamut (2007) show is often not the case and a partial cause of implementation shortfall.

Firstly, I show that market exposure is a main driver of transaction costs of current trades and that it explains more than 60% of the variation in transaction costs. This finding is robust over time and does not go against earlier findings in transaction cost literature. Almgren, Thum, Hauptmann, and Li (2005) and Zarinelli, Treccani, Farmer, and Lillo (2015) show that transaction costs follow a concave dependency on the size of a trade and that size is a main driver of transaction costs. I show that this is still the case when taking market exposure into account.

Secondly, I show that findings in machine learning return prediction literature, such as Fischer and Krauss (2018) and Becker and Leschinski (2018), are similar in this data set and robust over time. I use 240 lagged daily returns to classify next day's return direction and show that classification performance increases with model complexity. Logistic regression performs at 51% accuracy, neural networks perform at 53% accuracy and LSTM's perform at almost 55% showing that taking into account cross-temporal interaction of the

lagged returns and recurring return patterns increases performance respectively.

Thirdly, I show that the signal generated from these predictors can be used to decrease transaction costs and improve timing costs. By changing the duration of trades using these signals we can decrease transaction costs with 3 basis points per trade using logistic regression prediction, 4 basis points per trade using the neural network predictions and up to 11 basis points per trade using LSTM predictions. Furthermore, instead of changing the duration of a trade I show that we can improve the timing costs of the trade by delaying the trade until the next day. By using the return direction predictions as a proxy for timing cost predictions I show that we can reduce transaction costs with 6 basis points per trade using predictions of the neural network, and 32 basis points per trade using the predictions of the LSTM.

These findings are in line with the body of literature into machine learning return direction prediction showing that interaction between regressors and recurring patterns in regressors increase the classification performance of predictors. Furthermore, these findings consolidate the findings of transaction cost literature with that of machine learning prediction and furthers the discussion of optimal execution with the use of prior information.

1.2 Literature

The term *implementation shortfall* was first coined by Perold (1988) and, in its broadest sense, refers to the difference of returns between the implementation of investment decisions on paper versus implementation in reality. On paper an investment decision can be implemented instantaneously, at any time, in any quantity for the best price, whereas in reality commissions, fluctuations in prices and lack of liquidity will hinder optimal execution. Perold (1988) shows, by comparing the Value Line ranking system and funds that use the Value Line ranking system, that shortfall alone can drastically decrease the potential of an investment strategy. Whereas the Value Line ranking system outperformed the market by 20%[1] per year on paper, the Value Line funds only outperformed the market by 2.5% a year. This illustrates the impact shortfall can have on portfolio returns.

Perold (1988) notes that shortfall has two distinct components, namely, execution costs and opportunity costs. Execution costs are the costs that are incurred by executing an order and opportunity costs are the gains that are missed out on when an order can not be executed in its entirety. Wagner and Edwards (1993) further expand this definition

[1] Over the period 1965 - 1986.

and split execution costs in commission, which is the explicit fee paid to the broker, market impact, which is the price adjustment necessary to create enough liquidity to accommodate the trade, and timing cost, which is the change in price that occurs before the execution but after the submission of the order to the trading desk. Kissell (2006) notes that these execution costs can more finely be defined and add price appreciation, which is the change of price during the execution of a trade, and spread, which is the difference between the best offer and best bid price. In this thesis I regard market impact, spread and price appreciation, or market exposure, in modelling explicit costs of execution, which I call transaction costs. I analyze market exposure as a main driver of transaction costs.

Wagner and Banks (1992) show that minimizing transaction costs can have as great an effect on portfolio returns as a major allocation shift. Wagner and Edwards (1993) coin the idea that the moment an investment decision is made should not necessarily be followed by immediate implementation of that decision since market circumstances might hinder attaining the goal of that decision. Managers should apply thoughtful implementation that is in line with the goal of their investment decisions. This is similarly discussed by Loeb (1983) who states that shortsighted investing decreases returns and that careful assessment of liquidity of the instrument and the size of the trade is necessary. This is concretely shown by Kissell (2006) and Kissell and Malamut (2007) who show that implementation shortfall can often be attributed to utilization of trading styles that are inconsistent with the investment objective. Thus, for investment decisions and portfolio implementation to attain the same goal optimal implementation should be incorporated in the investment decision.

Implementation used to be heavily subjected to the whims of the market, however, the advent of digitization and increase in trading volume have lowered execution times and increased overall market liquidity. Together with new technology, investors now have more control over the portfolio implementation process. This has led to a rise of research into optimal portfolio implementation which is aimed at giving managers the tools to best carry out implementation. Keim and Madhavan (1997) and Chan and Lakonishok (1995) show that institutional trades are generally so large that they are split into smaller pieces to minimize transaction costs. How large these pieces are and at what speed they are traded is called the execution strategy or the trade trajectory. Bertsimas and Lo (1998) formulate optimal execution as a dynamic optimization problem since trading takes time, market liquidity is not perfectly elastic and price impacts can affect the course of future prices. By minimizing the market exposure and expected impact of a large trade over a fixed time horizon we can obtain the optimal execution strategy. Bertsimas, Hummel,

and Lo (2000) take this a step further and obtain the optimal sequence of trades as a function of market conditions. Almgren and Chriss (2001) obtain the optimal execution strategy by minimizing a combination of volatility risk and transaction costs arising from permanent and temporary market impact. Furthermore, they show that this impact is mainly driven by order size. Stanzl and Huberman (2000) argue that in the absence of arbitrage opportunities, market impact should be linear in order size, however, in practice this is shown not to be the case. Lillo, Farmer, and Mantegna (2003) show that this impact is concave which is a more general proof of the rule of thumb posed by Torre (1997) who states that impact, and thus transaction costs, follow a square root rule on order size. Almgren (2003) shows that the optimal execution strategy can also be found for such nonlinear impact functions and Almgren et al. (2005) show that impact can directly be estimated using a power function. They find that the impact function has a power of 3/5, however, more recent research by Zarinelli et al. (2015) find that this concave shape is due to a logarithmic dependency rather than a power law.

The optimal execution solutions to these dynamic programming problems usually lead to an execution strategy with high emphasis on speed. This increases costs due to market impact but reduces exposure to adverse price change, which are usually of a far greater magnitude than liquidity costs — percentages versus basis points — so this solution is understandable from a single trade perspective. However, whereas costs due to liquidity will always be present, when trading vast amounts of orders on a daily basis the average exposure to adverse price changes, or market exposure, of all trades aggregated becomes close to zero. Mackintosh and Ewen (2008) show that this has caused traders to swap quick trade strategies for slower trade strategies in order to reduce aggregate transaction costs and thus many optimal execution solutions have become irrelevant in practice.

However, transaction costs due to market exposure are basically the result of negative exposure to short term price changes. Thus, future information on market exposure could be relevant in finding optimal execution strategies. Israelov and Katz (2011) show that short term price information can be used to guide investors on how to time their trade. By strategically timing the market we can decrease negative exposure of portfolios to short term signals and thus decrease transaction costs due to market exposure. Furthermore, short term trading signals that do not cover their own trading cost now become valuable. Given the finding of Mackintosh and Ewen (2008) that traders tend to opt for longer trades to minimize impact costs we can use short term information to either change the exposure of a trade to favorable or unfavorable price change by changing the duration of a trade, or delay a trade to negate unfavorable price change and thus exposing the portfolio

to short term information. This consolidates the objective of the portfolio manager and the trader as mentioned by Kissell and Malamut (2007) since this exposes the investment portfolio to relevant short term information and directs the trade to a more favorable trading price.

Consequently, short term trading signals can be used as a predictor for market exposure. Heston, Korajczyk, Sadka, and Thorson (2011) find that there are predictable patterns in intraday stock returns. Moreover, Linton and Whang (2007) show there is directional predictability in daily stock returns. Han, Zhou, and Zhu (2016) show that past prices contain relevant information for future returns by constructing a trend factor that linearly incorporates a range of moving averages that capture the three major price patterns: the short-term reversal effects, the momentum effect, and the long-term reversal effects. One can easily extend this research by constructing a daily trend factor that predicts the daily price direction of a stock. This shows that past returns contain relevant information to predict market exposure of trades.

A novel way to estimate the direction of daily stock returns is by estimation using nonlinear and nonparametric models such as a logistic regression (Berkson, 1944) or a neural network (McCulloch & Pitts, 1943) for binary classification. These methods are computationally intensive and have only recently seen practical use with the increase in computer power. Hornik, Stinchcombe, and White (1989) and Cybenko (1989) prove that a neural network, given appropriate architecture, is able to generalize any practical type of functional dependency. Since the direction of stock returns entails many nonlinear dependencies and dynamic aspects a neural network is especially well suited for the estimation of this problem. Qiu and Song (2016) and Selvin, Vinayakumar, Gopalakrishnan, Menon, and Soman (2017) show that neural networks optimized with genetic algorithms and respectively convolutional neural networks are capable of identifying and forecasting trends in daily stock market returns. Becker and Leschinski (2018) show that it is possible to meaningfully extend the investigation of neural networks to individual stocks in the Dow Jones Index. Since there might be recurring patterns in stock prices the use of recurrent layers, such as the long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997), in neural networks might be advantageous. Furthermore, since daily returns notoriously contain much noise, denoising might also improve predictions. Bao, Yue, and Rao (2017) pose a framework for neural networks with denoising capabilities and a long short-term memory layer for financial timeseries prediction. Such a framework is put into practice by Fischer and Krauss (2018) for constituents of the S&P 500 and is shown to effectively extract meaningful information from noisy financial timeseries data for daily

trend prediction. The vast amount of data also justifies the use of neural networks since neural networks take many data to train but offer more potential in generalizing to the data. Furthermore, Gu, Kelly, and Xiu (2019) state that neural nets do not suffer from multicollinearity in their generalization. Optimization might slow down as a result but the optimal solution will not suffer. This is also an advantage since many factors that influence stock return directions are distinct but correlated. A limitation however, is that neural networks can not tell us about the underlying economic mechanisms or market microstructure that drives the data generating process. This is the trade-off for using neural nets.

This research contributes to the research of Perold (1988), Wagner and Edwards (1993) and Kissell (2006), by demonstrating the impact of several parts of implementation shortfall on portfolio returns. This research also contributes to field of transaction cost modelling, led by Almgren and Chriss (2001) and Bertsimas et al. (2000), by showing that the market exposure term has a large influence on aggregate transaction costs and should be of more interest in the optimal control literature. This research also adds to the research of Han et al. (2016) by extending the trend-factor to a higher frequency signal and showing that it can be used to expose longer term signals to short term information. It also adds to the plethora of literature of neural networks in financial timeseries prediction and furthers the synthesis of finance and machine learning fields. Furthermore, this research shows that we can consolidate the goals of investors and traders, emphasized by Kissell and Malamut (2007), by putting into practice the ideas posed by Israelov and Katz (2011) using short term price information.

1.3 Central Idea

Most research on reducing costs due to implementation shortfall has been focused on reducing impact. However, as shown by Mackintosh and Ewen (2008), the solutions posed by this research are not generally used in practice and, instead, traders generally opt for flatter trade trajectories. This is to minimize aggregate transaction costs over a great amount of orders and is the optimal solution if no further market information is assumed by the traders. If we look at the definition of the components of shortfall, posed by Kissell (2006), we see several components where we can benefit from such information, namely: market exposure and timing cost. As discussed in Section 1.2, research into predicting short term market information is abundantly available. The proposition of Israelov and Katz (2011) to use this short term information to guide traders in timing

their trades can thus be implemented by changing market exposure or the timing of a trade. This also means that short term price signals that usually have transaction costs that supersede the gains now become useful.

Changing the exposure of a trade to the underlying price appreciation or depreciation can be done by changing the duration of a trade. A faster trade will have a smaller exposure to price change and longer trade will have a higher exposure to price change. Changing the timing of a trade is simply delaying the trade until a more favorable price is available. In both cases the underlying portfolio is exposed to a more favorable price, and thus employing traders to do this will consolidate the goals of the portfolio manager and the trader. The only additional information a trader needs to achieve this is a prediction of the direction of the price. In Table 1 I show the resulting exposure of a trade given the side of an order and the direction of the price.

Table 1: Exposure given order side and price direction. This table shows the sign of the exposure of a trade given the price direction of the underlying stock and the side of the order.

Exposure		Price Direction	
		Down	Up
Order side	Buy	Positive	Negative
	Sell	Negative	Positive

If the direction of the price of the underlying stock is up, and the order is to buy the stock, then the exposure is negative since the price becomes higher and thus less favorable to buy at. The trader should thus employ a trading strategy with a short duration to minimize the exposure of the trade and the underlying portfolio to the adverse price change. This is also the case when the price direction is down and the order is to sell. Similarly, if the direction of the price is up and the order is to sell, then the exposure is positive and thus the trader should employ a trading strategy with a long duration to maximize exposure, or delay the trade to wait for a more favorable price. This is also the case when the price direction is down and the order is to buy. Thus, to summarize, a price direction prediction gives a prediction of the exposure of the trade, a positive exposure prediction should lead to a longer or delayed trade, and negative exposure prediction should lead to a shorter trade. This simplifies the prediction of future prices since a prediction of the direction of the price is sufficient. This is an advantage because the magnitude of returns is notoriously hard to predict.

This thesis is essentially a guideline on how to handle short term information in the portfolio implementation process, and a showcase of the impact it has on transaction costs. Firstly, in Section 2 I describe the data and define the variables used in each

step. Furthermore, I show some preliminary findings using exploratory analysis of several variables. Secondly, in Section 3, I model transaction costs using its main components to see how exposure affects transaction costs. This gives us a grasp of what the effects are of the different variables on transaction costs, but most importantly it shows us the impact of exposure on transaction costs. Thirdly, in Section 4, I define several daily price direction predictors. These give a binary prediction of the direction of the price and consequently a prediction of the exposure of a trade. Given this prediction of exposure we can take an empirically plausible long or short duration and implement this duration in the models from step two which will give us an estimate of the altered transaction costs. We can then compare the exposure-adjusted transaction costs with the true transaction costs and see if changing the duration is beneficial and if the predictors are accurate enough to improve transaction costs. Furthermore, we can analyze the timing costs for the delayed trades and compare those to the actual costs.

2 Data

The data is provided by Robeco and contains all proprietary order data over the period January 2014 to December 2018. An order consists of a block of shares that is to be traded. The execution of such an order is split into several parts according to the liquidity on the market and the execution strategy. The execution of each part is called a fill. For each order we know the size in shares and dollar amount, the ex-ante price, the ex-post average price of the execution, the size as a fraction of the daily traded volume, the amount of fills, the transaction costs and the broker or platform. For each fill we know the size of the fill in shares and dollar amount, the price at which the fill is executed and the date and time of the fill. Furthermore the data includes daily returns of all stocks in Robeco's investible universe[2] over the period January 2010 to December 2018. This includes the returns corresponding to the day and stock of every order. The returns will be used to formulate a market exposure term and for return direction prediction.

2.1 Variables

In this section I describe all variables. Most of these variables are ex-ante measures and are therefore suitable for forecasting, however, a few are generally considered ex-post. I justify the use of the ex-post measures in the following descriptions. The use of ex-ante

[2]Information on the contents of this investible universe are proprietary and can not be disclosed, however, all countries to which the stocks in these orders are denominated are listed in Appendix A

variables enables us to predict transaction costs on trades that have not yet been executed.

Side means whether the trade is a buy or a sell order. This will be denoted as

$$\text{sign}(X) \tag{1}$$

where

$$X = \sum_{i=1}^{n} x_i \tag{2}$$

and where x_i is the size and side of fill i and sign$(\cdot)$ is the sign function. The side of an order is always known ex-ante and can thus be used in the prediction of exposure.

Transaction costs, as used in this research, is defined as the relative difference of the average price paid or received for an order and a benchmark price. Mathematically this is

$$C = \text{sign}(X)\frac{(\bar{S} - S_B)}{S_B} \tag{3}$$

where $\bar{S}$ is the weighted average price of the order, given by

$$\bar{S} = \frac{\sum_i s_i x_i}{\sum_i x_i}, \tag{4}$$

where x_i and s_i are the size and price of fill i respectively and where S_B is the benchmark price. Note that the sign of x_i denotes whether the order is a buy $(+)$ or a sell $(-)$. This, however, does not impact the average price. For the benchmark price I use the previous tick, which is the most recent price before the order. This is similar to the description in Almgren et al. (2005), and in Section 3 this will be the dependent variable. Transaction costs, as described in Equation 3, can be seen as a loss measure for the weighted average price of an order and thus the prediction of transaction costs can be seen as an ex-ante estimation of the average price you will pay or receive for a trade.

Fraction of daily volume (FDV) is the size of the order as a fraction of the average daily volume (ADV) traded during the day. This is an ex-ante estimation since we do not know the total volume that will be traded during a day. Almgren et al. (2005) use the fraction of the ten-day moving average of daily volume, however I use the 21-day moving

median of daily volume. Order size is shown in many research to be the main driver of transaction costs.[3]

Duration is the time it takes to complete the order. This is the difference in time between the first and the last fill of an order. When the duration of a trade is higher you expect the trade to have a higher market exposure and thus the magnitude of the transaction costs will be larger. Duration is an ex-post measure that is shown to be important in transaction cost modelling (Almgren et al., 2005). Traders do not always have full control over the implementation of orders since many orders are implemented algorithmically. However, for the sake of developing a model that can aid in predicting transaction costs I assume that the speed at which is traded is set beforehand.

Returns are the relative difference of the price of a stock at the beginning of the day compared to the beginning of the next day price and can be mathematically represented as

$$R_t = \frac{S_{t+1} - S_t}{S_t} \tag{5}$$

where S_t is the price at the beginning of day t and S_{t+1} is the price at the beginning of the next day. For the price of the beginning of the day I use the fifteen minute average opening price.

Timing costs are the costs or gains missed out on by delaying or not delaying a trade until the next day. It is the relative difference of the benchmark price S_B and the price at the beginning of the next day since in this research I assume a delayed trade will be executed the next day as markets open. This can be computed as

$$TC = \text{sign}(X)\frac{S_{t+1} - S_B}{S_B}. \tag{6}$$

This relative price difference is multiplied by the sign of the order since the a positive price change will negatively impact the timing costs of a buy order and positively impact the timing costs of sell order, and vice versa. Essentially, the price at the beginning of the next day will serve as the benchmark price for the delayed trade.

[3]This is shown in Torre (1997), Bertsimas and Lo (1998) and Almgren and Chriss (2001) among others.

Spread is the difference between the bid-price and the ask-price of a stock. This signifies the relative price gap you have to cross in order to immediately sell or buy after buying or selling. This metric is constantly changing so therefore I use the five-day weighted average spread expressed in basis points as an ex-ante estimate.

Market Cap, or market capitalization, is simply the market value, expressed in dollars, of a publicly traded company's outstanding shares.

2.2 Trader Behaviour and Filters

Kissell and Malamut (2007) show that implementation shortfall can often be attributed to the utilization of a trading style inconsistent with the investment objective. This is because traders and managers have different goals: a manager wants to maximize portfolio utility whereas a trader wants to achieve the best execution price compared to a certain benchmark. This shows that there is a discrepancy in local and global optimality in the portfolio implementation process. Traders have many tricks up their sleeves to beat a given benchmark without this truly adding any utility to the portfolio. A careful selection of the data must be made to ensure that true transaction costs are measured.

Simply removing orders without impact is not desirable since these might include incidents that have no impact by coincidence. Removing these would overstate presence of transaction costs. However, there are several methods traders use to minimize transaction costs that might affect the outcome of the analysis in this research. I remove all orders that are completed using such methods. Furthermore, some cut-off points have to be defined in order to remove outliers that might pollute the data.

1. Traders sometimes use so called 'crossing platforms'. These are used to find a counterparty for an order without showing the desire for this order to the market. The price for these orders are negotiated and the order is consequently executed virtually instantaneously. Given the nature of these orders, they tend to have a favorable price and thus do not incur impact or transaction costs. Furthermore they are shown to have only one fill.

2. I exclude any type of order that is not a market order. Market orders are executed at the best price on the market and thus remove liquidity. This means that only market orders have measurable impact on the market price, and thus incur meaningful transaction costs. Limit orders have a fixed limit price and are usually used when

there is less urgency to trade. Market-on-close orders are different from market orders and are executed at the close against the best price which usually results in one fill. Market orders are thus the only meaningful order type to include.

3. I exclude orders with only one fill. These orders usually occur when the order size is small or when a trader sees an opportunity to execute with minimal costs. I want to analyze the effect of market exposure on transaction costs. Since orders with one fill have no exposure to price change during the trade these are omitted.

4. I exclude orders that take more than one day or that are sent to the traders outside the trading hours of the corresponding exchange. Since these trades are usually, partially, carried over the night their duration is skewed and thus market exposure can not be reliably measured.

5. I exclude orders which are larger than 25% of average daily volume. Less than 3% of the data constitutes of orders larger than 25% of daily volume. These trades are unusually large and the variation in order size of these trades is too great to be of use in generalizing a model.

6. I exclude orders with a traded amount of less than 2000 USD. These are less than 0.001% of orders. The value of trades below this threshold is too low to be of any significance and usually the market exposure of the orders is negligible.

7. I exclude orders with a duration of less than one minute. This is about 0.5% of the orders. These orders generally bare too little information about transaction costs or market exposure to be of value in the forthcoming analyses.

8. I exclude orders that are posted to the trader before the opening time of the corresponding exchange since these often have the opening price as benchmark. Traders can manipulate the opening price by trading a small portion of the order far outside the bid-ask price at the opening, thus causing a favorable benchmark price. This distorts the data.

These filters are similar to the filters used in other transaction cost literature, such as Almgren et al. (2005) and Zarinelli et al. (2015). Table 2 shows the amount of data that is filtered per step. Each value states the amount of observations left after the filter is applied.

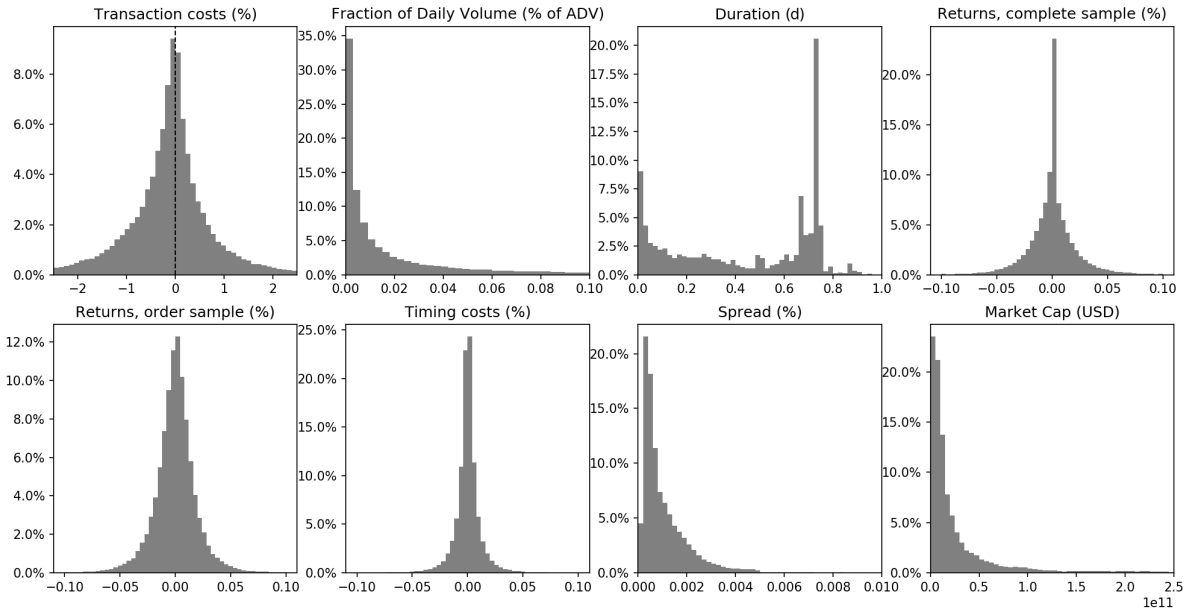
Table 2: Order data filters. This table shows the amount of orders left after each filter.

No Filter	Filter 1	Filter 2	Filter 3	Filter 4	Filter 5	Filter 6	Filter 7	Filter 8
316.198	308.730	170.259	149.480	107.321	103.534	103.458	96.303	80.408

2.3 Descriptive Statistics

After filtering we have 80.408 orders left, 42.740 of which are buy orders and 37.668 of which are sell orders. There is a skewness towards the buy side since Robeco has gained in assets under management over the past five years. Figure 1 and Table 3 show the distribution of the aforementioned variables.

Figure 1: Histograms of order data variables. This figure shows eight histograms of the variables mentioned in Section 2.1. The top row, from left to right shows the histograms of transaction costs as percentage of trading value, fraction of daily volume as percentage of average daily volume, duration as fraction of trading day, returns in percentage in the entire sample. The bottom row, from left to right, shows the histograms of returns in percentage in the order sample, timing costs in percentage, spread in percentage difference between best bid and best ask and market cap in US Dollars.



We see that the mean of the transaction costs is negative and that most of the weight of the distribution falls on the negative side. This is because all transactions incur impact which by definition is negative. This means there is plenty to be improved about the order process to try and shift this distribution to the positive side. We also see in the table that the transaction cost distribution has very fat tails meaning there are still outliers in both directions. Furthermore, we see that the distribution of the fraction of daily volume falls mostly under the 3% of FDV but has a fat tail up to the clipping point of 25%. The durations of the orders are peaked in several places. This is due to the closing of the market in other time zones. Orders that are placed in these markets are preferably executed before market close. Furthermore, I show the distribution of returns

Table 3: Distribution of variables. This table shows the distribution of all variables shown in Figure 1. The columns give the distribution of, from left to right, the transaction costs, fraction of daily volume, duration, return in the complete sample, returns in the order sample, timing costs, spread and market cap. For each variable the mean, standard deviation, skewness, kurtosis, 95^{th} quantile, 75^{th} quantile, 50^{th} quantile, 25^{th} quantile and 5^{th} quantile are given.

	Trans.	FDV	Dur.	R_{full}	R_{order}	Timing	Spread	Market cap
Mean	-14,044	0,026	0,444	0,001	0,000	0,000	0,001	3,22E+10
Std.dev.	112,118	0,043	0,288	0,049	0,021	0,014	0,001	6,81E+10
Skewness	0,090	2,600	-0,298	29,165	0,262	0,597	5,611	6,521055
Kurtosis	19,467	7,052	-1,536	2124,882	62,027	235,889	95,251	62,86859
Quantiles								
95%	141,298	0,124	0,757	0,039	0,030	0,018	0,003	1,40E+11
75%	26,847	0,028	0,722	0,008	0,010	0,004	0,001	2,82E+10
50%	-7,443	0,007	0,536	0,000	0,000	0,000	0,001	1,15E+10
25%	-52,727	0,002	0,142	-0,009	-0,009	-0,004	0,000	5,30E+09
5%	-186,148	0,000	0,010	-0,038	-0,031	-0,018	0,000	1,63E+09

in the complete sample and the distribution of returns on stocks and days corresponding to orders. The first distribution has a laplacian distribution with a high peak at the mean, around zero, and very fat tails, which is to be expected from high frequency returns. The latter distribution shows a more normal shape with a less pronounced peak near the mean, also around zero, and tails that are less fat. Both distribution have a positive mean and skewness indicating that most returns are positive. The first distribution however, has extremely fat tails with minimum of -100% and a maximum of 500%. When compared to the first and third quantile this shows that the data has some extreme outliers. The timing costs are essentially a restricted set of the returns in the order sample. There is a discrepancy between the benchmark price of a trade and the price at the beginning of the day since not all orders are executed at the start of the day. This results in a narrower distribution for the timing costs with a lower standard deviation and lower dispersion between upper and lower quantiles. Spreads are generally small and range between zero to ten basis points. Notably, spreads are always positive since negative spreads would imply arbitrage opportunities since a round-trip of buying and immediately selling would then yield the bid-ask spread premium. Market cap of the stocks ranges hundreds of millions to hundreds of billions. This is generally the full range of market cap as far as institutionally tradable stocks go.

2.4 Exploratory Analysis

In this section I show the relation between transaction costs and daily returns and I propose a market exposure term that can be used as a regression term in a transaction

cost model. A prediction of this term can subsequently be used to guide traders or investors on how to time the trade. Positive transaction costs mean that the average price of an order was more favorable than the most recent tick before the order and vice versa for negative transaction costs. In that sense, positive transaction cost is preferable over negative transaction cost for investors.

The goal of this research is to improve overall transaction costs by evading negative market exposure of a trade. Figure 2 shows how transaction costs are correlated with stock returns and several market exposure terms in four subplots labeled A to D.

Figure 2: Scatterplot of transactions costs and market exposure terms. Subplot A shows the scatterplot of transaction costs against return split into buy and sell orders. Subplot B shows the scatterplot of transaction costs against return times duration split into buy and sell orders. Subplot C shows the scatterplot of transaction costs against return times side times duration split into duration longer and shorter than half a day. Subplot D shows the scatterplot of transaction costs against return sign times order side times duration split into duration longer and shorter than half a day.

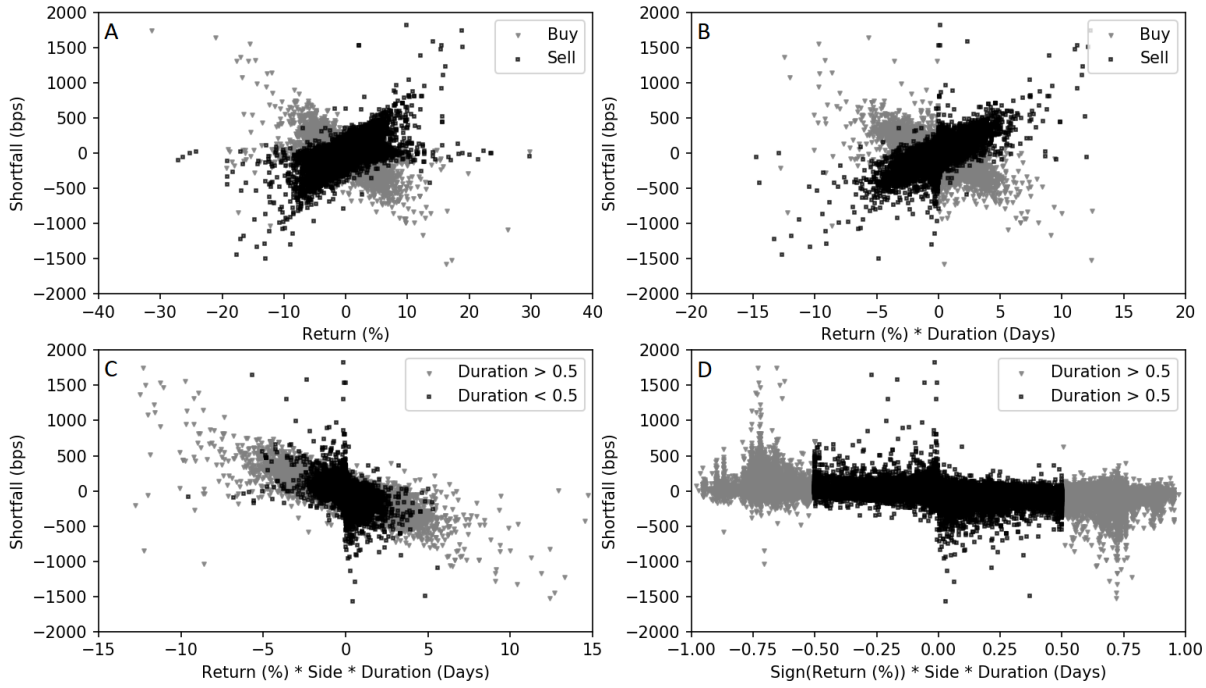


Figure 2A shows a distinct correlation between transaction costs and the daily return of the corresponding stock. In the case of a sell order the transaction costs generally increase in the case of a positive return and decrease in the case of a negative return. Conversely this is mirrored in case of a buy order. Interestingly the scatterplot could be characterized as a band or rectangle with wide tails. This is probably due to absolute returns being higher when volatility is high. The lower volatility is, the smaller the return generally is and the smaller the average change in price is during that day and thus the smaller the dispersion of transaction costs.

The market exposure of an order is not synonymous to the return of the corresponding

stock but rather the duration at which it is exposed to that return. This can be modeled by multiplying the duration, as a fraction of the day, with the return. This gives the duration for which the trade is exposed to this daily return. Figure 2B shows the relation of transaction costs with this exposure term. Again we see that the transaction costs increase/decrease with increasing return in case of sell/buy, although this plot seems more homogeneous. The scatterplot has positive and negative peaks around zero exposure. This is probably due to return being close to zero and thus causing exposure to be close to zero as well. However zero return does not mean minimal price movement. This means that, although exposure is close to zero, transaction costs can still be significant.

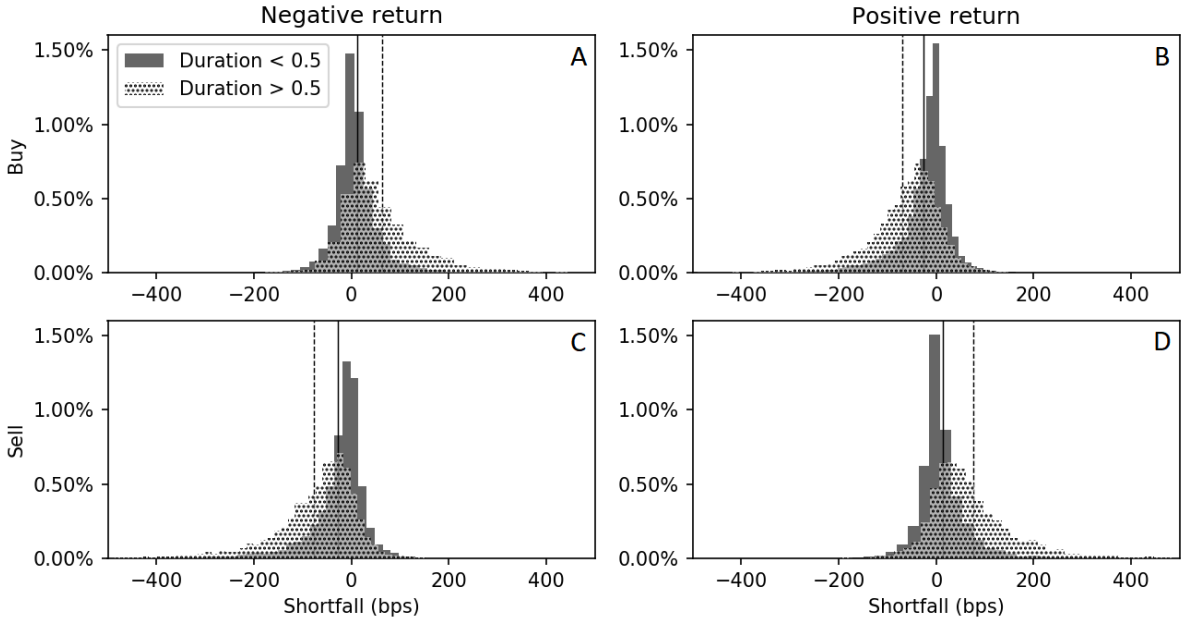
To aggregate the relation of transaction costs with the exposure we multiply the exposure term with the *side* of the order, assuming that the effect of the exposure term is symmetric for buy and sell orders. Figure 2C shows the relation of transaction costs with the aggregated exposure term. Here orders are shown with a duration of longer and shorter than half a day seperately. This clearly shows that orders with a low duration have a slightly higher dispersion but smaller absolute transaction costs in general.

To improve the exposure of an order we must change the duration of the order according to a prediction of the return. Since it is extremely difficult to correctly predict daily returns a commonly used alternative is to look at the sign of the return. Figure 2D shows the relation of transaction costs with the exposure term where only the sign of the return is used. A large amount of information is now lost on the strength of the correlation between transaction costs and the exposure term, however, there is still a distinct downward correlation visible such as in the third plot. This shows that ex-ante information about the sign of the return might still be valuable in improving market exposure. These insights will be used in modelling transaction costs in Section 3.

In Figure 3 the distribution of transaction costs is decomposed in four histograms, labeled A to D. Each histogram contains two distribution, resulting in eight distributions in total. The histograms on the left hand side show the distributions of the transaction costs for orders on a day with negative return and the histograms on the right hand side show the distributions of the transaction costs for orders on a day with positive returns. The histograms on the top row show the distributions of the transaction costs of buy orders and the histograms on the bottom row show the distributions of the transaction costs for sell orders. All histograms show the distributions for transaction costs of trades with long and short duration, e.g. longer and shorter than half a day respectively. Furthermore, the solid lines show the means for the solid distributions and the dashed lines show the means for the dotted distributions. All distributions are roughly equally populated and

are essentially the distribution of transaction costs decomposed for market exposure, order side, price change and duration.

Figure 3: Histograms of transaction costs given return, duration and side. This figure shows the histograms of transaction costs, expressed in basis points, decomposed for return sign, trade side and duration. The histograms in the top row show the distributions of transaction costs for buy trades, the histograms in the bottom row show the distributions of transaction costs for sell trades. The histograms on the left-hand side show the distributions of transaction costs on days with negative return, the histograms on the right-hand side show the distributions of transaction costs on days with positive return. All distributions are decomposed into trades with duration longer or shorter than half a day.

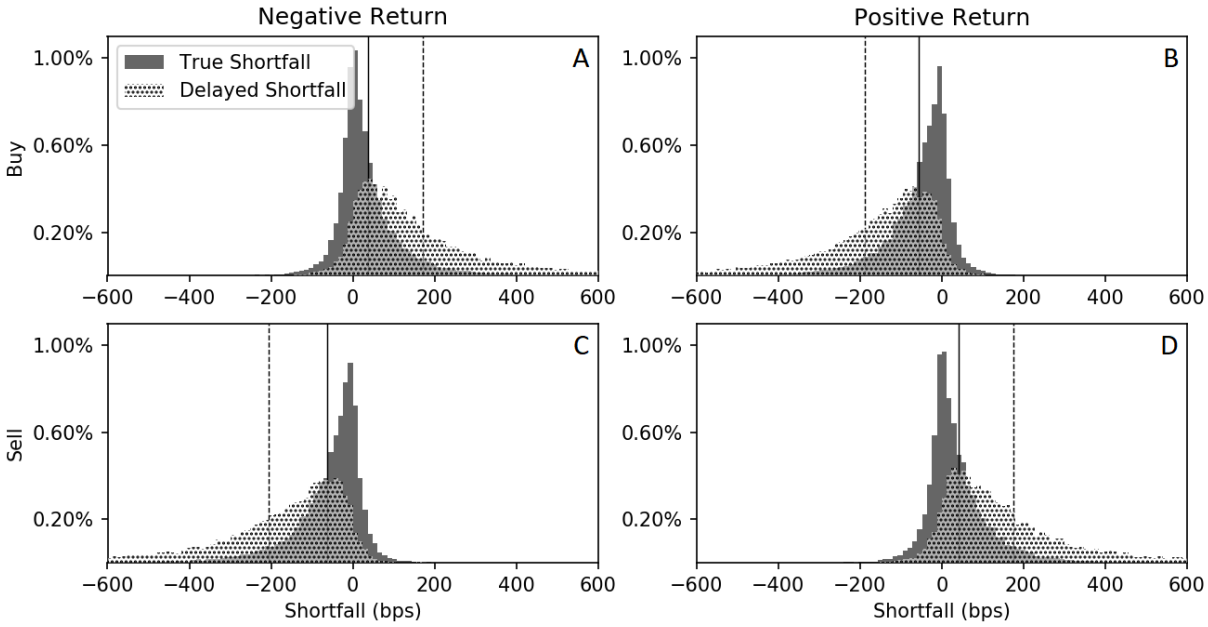


In each subplot we see that the duration of an order distinctly changes the distribution of the transaction costs in each subplot. As we have seen in the scatter plots this is to be expected, however, we now see that changing the duration given the circumstances can meaningfully shift the distribution of the transaction costs. In Figure 3A we see that transaction costs are mostly positive for buy orders on a day with negative return. Furthermore, the transaction costs are more positive for orders with a longer duration. As shown in Table 1, this is because the exposure of the order is positive and thus a longer duration is favorable for the transaction costs. This is similarly the case in Figure 3D. The opposite is the case in Figure 3B and 3C where a longer duration leads to more negative exposure of the trade to the underlying price change. Note that the distributions are not strictly positive or negative since the returns are over the entire day and merely serve as a proxy for the price change during the trade. This shows that duration has a significant effect on the transaction costs, depending on price direction and order side.

Another way to improve transaction costs is by changing the timing of a trade. Figure 4 shows the distribution of transaction costs, decomposed for price direction and order side, in four subplots labeled A to D. However, now the solid distribution in each subplot

is the true transaction cost distribution and the dotted distribution is the distribution of transaction costs with added timing costs. Each subplot is thus a layover comparison of the distribution of transaction costs with and without timing costs. Note that the solid distributions in Figure 4A is the aggregate distribution of the dotted and solid distribution of Figure 3A, since the distributions in Figure 4 are not split for duration. This is also the case for corresponding distribution of Figures 4B, 4C and 4D.

Figure 4: Histograms of transaction costs given return, duration and side. This figure shows the histograms of transaction costs and timing costs, expressed in basis points, decomposed for return sign, trade side and duration. The solid distribution in each histogram is the distribution of transaction costs, the dotted distribution in each histogram is the distribution of timing costs aggregated with transaction costs. The histograms in the top row show the distributions of transaction costs and timing costs for buy trades, the histograms in the bottom row show the distributions of transaction costs and timing costs for sell trades. The histograms on the left-hand side show the distributions of transaction costs and timing costs on days with negative return, the histograms on the right-hand side show the distributions of transaction costs and timing costs on days with positive return.



As in Figure 3, we see that timing cost has a major impact on transaction costs. The tails of the dotted distributions are much fatter than the original transaction costs. Delaying trades with positive exposure, shown in Figure 4A and 4D, thus yields orders with more favorable transaction costs — almost a 200bps improvement on average — due to improvements in timing cost. These results are dependent however on the accuracy of the prediction of the exposure. Since the distributions of the transaction costs of the delayed order are more dispersed, delaying an order introduces considerable risk. Missclassification of the market exposure might lead to delay in trades that should not be delayed and cause a deterioration of transaction costs of almost 200bps on average. Furthermore, the distributions in Figure 4 are slightly misleading, since the transaction costs on the delayed trade are taken to be the same. However, market conditions during

the delayed trade could be different, and, as we have seen, the return on that day is important for the true transaction costs.

3 Modelling Transaction Costs

In this Section, I describe several models to explain transaction costs given its different components. This will give an understanding of the driving factors of transaction costs and will serve as estimation models for accurately predicting transaction costs on trades with changed duration and on delayed trades. Furthermore, I explain how inference is done on these models.

3.1 Transaction Cost Models

Since the proprietary model that Robeco uses for transaction cost estimation is not disclosed I use several variations of the direct estimation model described in Almgren et al. (2005). The base transaction cost model is as follows

$$\text{A:} \quad C = \alpha + \beta FDV^{\gamma} + \varepsilon, \tag{7}$$

where C is the transaction cost and FDV is the fraction of daily volume. This is a standard power function where α, β and γ are estimated. The estimation of the exponent allows for flexibility of the shape. This way we let the data dictate the dependency of transaction cost on the order size. An important difference with the Almgren et al. (2005) model is that I do not use the sign function to denote the buy or sell of an order. This is because I model transaction costs whereas Almgren et al. (2005) model impact. The impact of a sell on the price is generally negative and that of a buy generally positive, however, in terms of transaction costs this is not the case. In terms of transaction costs we would expect a sell to have a negative impact on the price and thus causing the average price we receive for the order to be lower than the benchmark, thus causing negative transaction costs. For a buy it is analogous, if the impact on the price from a buy is positive we expect the average price we pay for an order to be higher than the benchmark, thus making transaction costs negative. In this reasoning I assume that the impact of size on transaction costs is symmetrical for buy and sell orders.

The impact of size on transaction costs is shown to be significant and thus remains in all following models. A lesser used explanatory variable is spread. I add spread to model A as follows

$$\text{B:} \quad C = \alpha + \beta FDV^{\gamma} + \eta SPR + \varepsilon, \tag{8}$$

where SPR denotes the spread and η is estimated. To assess the impact of spread on transaction costs all following models have a version with and without spread.

To assess the effect of price change on transaction costs I add the return of the corresponding stock and day to the regression model as follows

$$\text{C:} \quad C = \alpha + \beta FDV^{\gamma} + \delta_1 \text{sgn}(X)R + \varepsilon, \tag{9}$$

and

$$\text{D:} \quad C = \alpha + \beta FDV^{\gamma} + \delta_2 \text{sgn}(XR) + \varepsilon, \tag{10}$$

where $\text{sgn}(\cdot)$ is the sign function, X is the size, magnitude and side of the order and R is the return of the corresponding stock and day, which serves as a proxy of the price change, and where δ_1 and δ_2 denote the coefficient for each market exposure term. Model C models the effect of the change of price on transaction costs whereas model D only models the effect of the direction of the price change on transaction costs. This will give us insight in the different explanatory parts of the price change. Spread is added to model C and D as follows

$$\text{E:} \quad C = \alpha + \beta FDV^{\gamma} + \delta_1 \text{sgn}(X)R + \eta SPR + \varepsilon, \tag{11}$$

and

$$\text{F:} \quad C = \alpha + \beta FDV^{\gamma} + \delta_2 \text{sgn}(XR) + \eta SPR + \varepsilon. \tag{12}$$

As shown in Section 2.4, the effect of price change on transaction costs relies on the duration at which the trade is exposed to the price change. The market exposure term of model C and D is adjusted with duration as follows

$$\text{G:} \quad C = \alpha + \beta FDV^{\gamma} + \delta_3 \tau \text{sgn}(X)R + \varepsilon, \tag{13}$$

and

$$\text{H:} \quad C = \alpha + \beta FDV^{\gamma} + \delta_4 \tau \text{sgn}(XR) + \varepsilon, \tag{14}$$

where τ is the duration of the trade in days and where δ_3 and δ_4 are the coefficient of each market exposure term. These models are also augmented with the spread as follows

$$\text{I:} \quad C = \alpha + \beta FDV^{\gamma} + \delta_3 \tau \text{sgn}(X)R + \eta SPR + \varepsilon, \tag{15}$$

and

$$\text{J:} \quad C = \alpha + \beta FDV^{\gamma} + \delta_4 \tau \text{sgn}(XR) + \eta SPR + \varepsilon. \tag{16}$$

3.1.1 Regression

For estimating the transaction cost models I minimize the least absolute deviations since the transaction cost data has a highly non-normal distribution with fat tails. Least absolute deviations estimation is defined as

$$\min_{\theta} \frac{1}{M} \sum_{i=1}^{M} |y_i - f(\mathbf{x}_i; \theta)|, \tag{17}$$

where M is the number of observations, $f(\cdot)$ is the regression model, $\mathbf{x}_i$ is the set of regressors for observation i, y_i is the transaction cost for observation i and θ is the parameter set.

I opt for this estimation method since the presence of fat tails in the distribution of the transaction costs suggests there are outliers present, even after filtering. Least absolute deviations weighs all large error terms less heavily than least squares. This makes this estimation method more robust to outliers, since outliers might cause least squares to become biased as shown in a simulation study by Dielman (1986). A problem one might run into with least absolute deviations is non-unique solutions. However, as shown by Fisher (1961), this is only the case if the fitted curve or hyperplane does not pass through at least k data points, where k is the number of parameters of the curve fitting model. Given the amount of parameters in the models never exceeds five and the regression data contains tens of thousands of points, it is highly unlikely that a solution is non-unique.

3.1.2 Bootstrapping Standard Errors

In order to make inference about the regression models I apply bootstrapping to obtain robust standard errors. Bootstrapping is favorable in this case since the loss function of the least absolute deviations is not continuous at zero and thus not differentiable at zero. This makes the use of other methods, such as the sandwich estimator by Huber et al.

(1967), inapplicable.

The standard error is estimated by resampling l sets of m observations, $C^{(1)} \dots C^{(l)}$ from the original sample, C, which contains n observation and where $n > m$. Then the same LAD-regression models are applied on each set $C^{(1)}$ to $C^{(l)}$.[4] The parameter estimates are then collected in $\hat{\theta}^{(1)} \dots \hat{\theta}^{(l)}$ and consequently the standard deviations of the parameters in $\hat{\theta}^{(1)} \dots \hat{\theta}^{(l)}$ are calculated giving an estimate of the standard errors. Given the size of the sample this method is especially applicable to compute standard errors.

3.2 Performance Evaluation

I use three methods to assess the performance of the models. The first is a robust R^2 measure that shows the goodness of fit for least absolute deviation models with outlier prone data. The second method is walk-forward validation which is a statistical model validation method which yields an evaluation metric for the predictive power for each model. The third method is the a modification of the Chow-test which evaluates whether a model changes over time, signifying whether there are structural breaks in the data.

3.2.1 Robust R-squared

Given the presence of outliers in the transaction cost data and the use of least absolute deviations, the usual coefficient of determination,R^2, does not apply. To assess the goodness of fit of the regressions, I apply a robust coefficient of determination designed by McKean and Sievers (1987) specifically for least absolute deviation analysis. They define a LAD test of $H_0 : \theta = 0$ vs $H_A : \theta \neq 0$ using the natural metric $RAD = AD(\hat{\theta}_0) - AD(\hat{\theta})$ where $AD(\hat{\theta})$ is the sum of absolute deviations of a model with parameter set $\hat{\theta}$, and $AD(\hat{\theta}_0)$ is the sum of absolute deviations of a model with only a constant. Large values of RAD indicate H_A but this needs to be measured on an appropriate scale. The natural scale parameter for the LAD fit is

$$\tau = 1/(2SC), \tag{18}$$

where SC is estimated using

$$\widehat{SC}_\gamma = (2z_{\alpha/2})^{-1}(e_{(n-k+1)} - e_{(k)}), \tag{19}$$

[4]The choice of l and m are somewhat arbitrary as long as the subsample is smal enough to account for all variability in the sample and the amount of subsamples is large enough to find a consistent estimate. In this case I chose $l = 1000$ and $m = 10000$.

where $e_{(i)}$ are the ordered absolute deviations, $e_{(1)} \leq e_{(2)} \leq ... \leq e_{(n)}$, $z_{\alpha/2}$ is the upper tailed value from the standard normal distribution with $\gamma = (1-\alpha)100\%$, n is the number of observations and $k = \frac{n+1}{2} - z_{\alpha/2}(\frac{n}{4})^{1/2}$. SC_γ is the length of a nonparametric confidence interval which is shown by McKean and Schrader (1984) to provide accurate inference under least absolute deviation regression. The coefficient of determination is then computed as

$$R^2_{LAD} = RAD/(RAD + (n-p-1)(\hat{\tau}/2)) \tag{20}$$

where p is the number of parameters in the model.

3.2.2 Walk-forward Validation

The purpose of model validation is to test the ability to generalize to out of sample data and perform accurate predictions. Although the data I use does not follow a strict time-series structure, the underlying mechanisms that drive the data generating process might be subject to temporal patterns. For this reason I use walk-forward model validation. Other cross-validation methods use non-chronological subsampling which, in this case, would incorporate future information in model evaluation.

The testing data is divided into $k = 11$ subsamples after which the models are fitted on the training and validation data. The models then make predictions for the first subsample after which the Mean Absolute Prediction error (MAPE) is computed. I do this for all k subsamples. To check whether the models are robust over time, I use a rolling window and an expanding window for the walk-forward validation.

3.2.3 Testing for Structural Breaks

To test for the presence of structural breaks I use the F-test proposed by Furno (2014), which is a simple modification of the Chow-test (Chow, 1960) to test equality between sets of coefficients of two regressions.

The F-test tests $H_0 : \theta_1 = \theta_2$ vs $H_a : \theta_1 \neq \theta_2$ where θ_j is the parameter set of the regression model on the j-th subsample, where the subsamples are taken over time. The corresponding test statistic is given by

$$F = \frac{(A_0 - A_1 - A_2)/k}{(A_1 + A_2)/(n_1 + n_2 - 2k)}, \tag{21}$$

where A_0 is the sum of absolute deviations over the entire sample and A_j, with $j > 0$, the

sum of absolute deviation over the j-th subsample, k is the amount of parameters in the set θ_j and n_j the amount of observations in the j-th subsample. The F-statistic follows a $F(k, n_1 + n_2 - 2k)$ distribution under H_0.

4 Predicting Exposure

In this section I describe the models I use to predict return directions and consequently the exposure of trades. Furthermore, I describe how these models are trained and how the performance of these models is tested. To conclude, I describe how the predictions from these models are used to find improvements in transaction costs and timing costs.

4.1 Prediction Models

In this subsection I describe four models I use to predict the direction of returns. Although these models predict or classify the direciton of future the output of these models are essentially predictions of the market exposure for orders given that we know the side of an order ex-ante.

4.1.1 Trend Factor

The first model I use for this purpose is a method developed by Han et al. (2016). A fundamental problem in constructing trend-signals is the subjective and fixed choice of look-back horizons. We can cope with this problem by combining a set of several moving averages and dynamically incorporating their containing information in a trend forecast. I do this by applying a two-step procedure.

First, Han et al. (2016) conduct a cross-sectional regression on various moving average lags:

$$r_{i,t} = \beta_{0,t} + \sum_j \beta_{j,t} \tilde{A}_{i,t-1,L_j} + \epsilon_{i,t}, \tag{22}$$

where I use the daily return, $r_{i,t}$, of stock i at time t, $\beta_{0,t}$ is the intercept at time t and $\tilde{A}_{i,t-1,L_j}$ is the normalized moving average signal[5] for stock i up until time $t-1$ with lag

[5]The moving average signal $A_{i,t-1,L_j}$ is calculated as $\frac{P^{t-1}_{i,d-L_j+1} + P^{t-1}_{i,d-L_j+2} + \ldots + P^{t-1}_{i,d-1} + P^{t-1}_{i,d}}{L}$ and is normalized by its closing price $P^{t-1}_{i,d}$ to get the normalized moving average signal $\tilde{A}_{i,t-1,L_j}$.

L_j.[6]

Next, I forecast the return in the next period, $t+1$, by

$$\mathbb{E}_t\left[r_{i,t+1}\right] = \mathbb{E}_t\left[\beta_{0,t}\right] + \sum_j \mathbb{E}_t\left[\beta_{j,t+1}\right] \tilde{A}_{i,t,L_j}, \tag{23}$$

where $\mathbb{E}_t\left[\beta_{j,t+1}\right]$ is the estimated expected coefficient of the trend signal with lag L_j. We take this as the average over the past 12 months of the estimated loadings on the trend signals,[7] which we obtained from Equation 22. Hence, $\mathbb{E}_t\left[\beta_{j,t+1}\right] = \frac{1}{12}\sum_{m=1}^{12} \hat{\beta}_{j,t+1-m}$.

Han et al. (2016) use this procedure to forecast monthly returns, however, I apply it to daily returns. Then I select the top and bottom portfolios by selecting the 50% highest and lowest predictions of $r_{i,t+1}$ and I scale the returns to 1 and 0 if they belong to the top or bottom portfolios respectively. I use a 50% top and 50% bottom division so that a prediction is available for the full sample. This gives the predicted binary classifications of the return direction.

4.1.2 Logistic Regression

A simple modification of the regression framework for return prediction is the logistic regression for classifying positive or negative returns. This technique was coined by Berkson (1944) and subsequently popularized in the machine learning literature.

The regression model is denoted as follows

$$f(\mathbf{x}_i;\theta) \equiv y_i = \frac{1}{1+e^{-\theta'\mathbf{x}_i}} \tag{24}$$

where θ is the parameter set that is to be estimated and $\mathbf{x}_i$ is the regressor vector for observation i. The predictions that follow from Equation 24 fall in the continuous set $[0,1]$ and can be regarded as classification probabilities. In this thesis I use a set of 240 past returns as $\mathbf{x}_i$ and use binary classification for future returns, y_i, where negative returns are set to 0 and positive returns to 1. A predicted classification probability lower than 0.5 is seen as a prediction for negative return and a predicted classification probability higher than 0.5 is seen as a prediction for positive return.

[6] The lags are picked equal to 3, 5, 10, 20, 50, 100, 200, 400, 600, 800, and 1000 days, to represent the daily, weekly, monthly, quarterly, one-year, two-year, three-year and four-year price trend of the underlying stock.

[7] We follow Han et al. (2016) by doing so.

4.1.3 Neural Network

The third model I use to predict return directions is a neural network for binary classification which can be schematically shown as a directional acyclical graph.

Figure 5: Diagram of neural network. This is a diagram of the architecture of a fully connected neural network. $x_{i,t-240}$ to $x_{i,t-1}$ denote the inputs, the lagged returns. In this thesis I use 4 hidden layers with 40 nodes per layer. The output layer is a single node containing the softmax function.

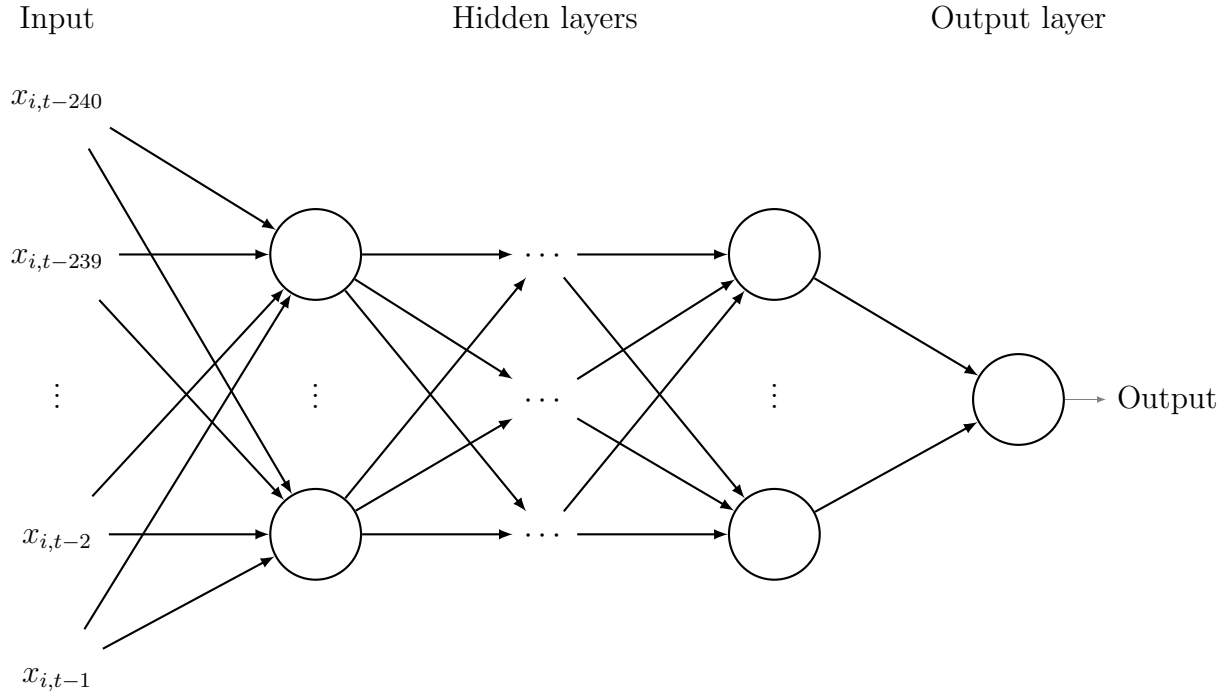


Figure 5 shows a diagram of a general fully connected neural network where the key components are the amount of layers, the amount of nodes per layer and the edges between the nodes. Note that all nodes of one layer are connected to all nodes of the next layer.

Mathematically, the output of each layer can be formulated in matrix form as

$$a^{(l)} = \sigma(W^{(l)}a^{(l-1)} + b^{(l)}) \tag{25}$$

where $a^{(l)}$ is the output vector of layer l and where $a^{(0)}$ is the input layer, or the regressor vector, $\mathbf{x}_i$. $b^{(l)}$ is a vector of biases or constants, which represent the state of each node when all weights would be zero, $W^{(l)}$ is the weight matrix containing the weights on the edges between layer $l-1$ and l. $W^{(l)}$ is of size $n^{(l)} \times n^{(l-1)}$ where $n^{(l)}$ represents the number of nodes in layer l. Each element i in row j in W represents the weight that connects node i in layer $l-1$ to node j in layer l. Equation 25 is a recursive formula, running from 0 to L, where $L-1$ is the number of hidden layers. This recursive formula can be summarized as $f(\mathbf{x}_i;\theta)$, where it takes input variables, $\mathbf{x}_i$, and transforms them with the weights and biases which are summarized in the parameter set θ.

The function $\sigma(\cdot)$ is called the activation function and is a transformation of the input variables. For the activation functions I use the rectified linear unit (ReLU), posed by Hahnloser and Seung (2001), which is defined as

$$\sigma(u) = \max(0, u). \tag{26}$$

This activation function has been shown in several works to perform well in learning and generalizing similar problems.[8] The activation function is the same for all layers except for the output layer. Since I apply the neural network to a binary classification problem the output function will be a softmax function, which is the same as in Equation 24. Essentially, this neural network can be seen as a logistic regression with interconnected layers of regressors on top. The neural network accounts for interaction of the variables and thus is a especially qualified in tackling return predictions with past return information.

4.1.4 Long Short-term Memory

To increase the capability of learning temporal sequences of the neural net I add a layer of Long Short-term Memory (LSTM) cells (Hochreiter & Schmidhuber, 1997) on top of the neural net. This is the fourth method I use to predict return directions.
Figure 6 shows a neural net with an LSTM layer. The nodes of the last hidden layer are each connected to a single respective LSTM cell. Each LSTM cell is connected to the next LSTM cell, and the last LSTM cell is connected to the output node.

An LSTM cell consists of a set of semi-recurrent units. These units control the flow of information through the cell and which information is retained in its memory. A diagram of an LSTM cell is shown in Figure 7 where the units are depicted as rectangular boxes, which are named accordingly.

[8] See e.g. Glorot, Bordes, and Bengio (2011) and Gu et al. (2019)

Figure 6: Neural network with LSTM layer. This is a diagram of the architecture of a neural network with layer of LSTM cells. Each node of the last layer of the neural network is connected to a single LSTM cell which in turn is connected to the following LSTM cell. The last LSTM cell is connected to the output node.

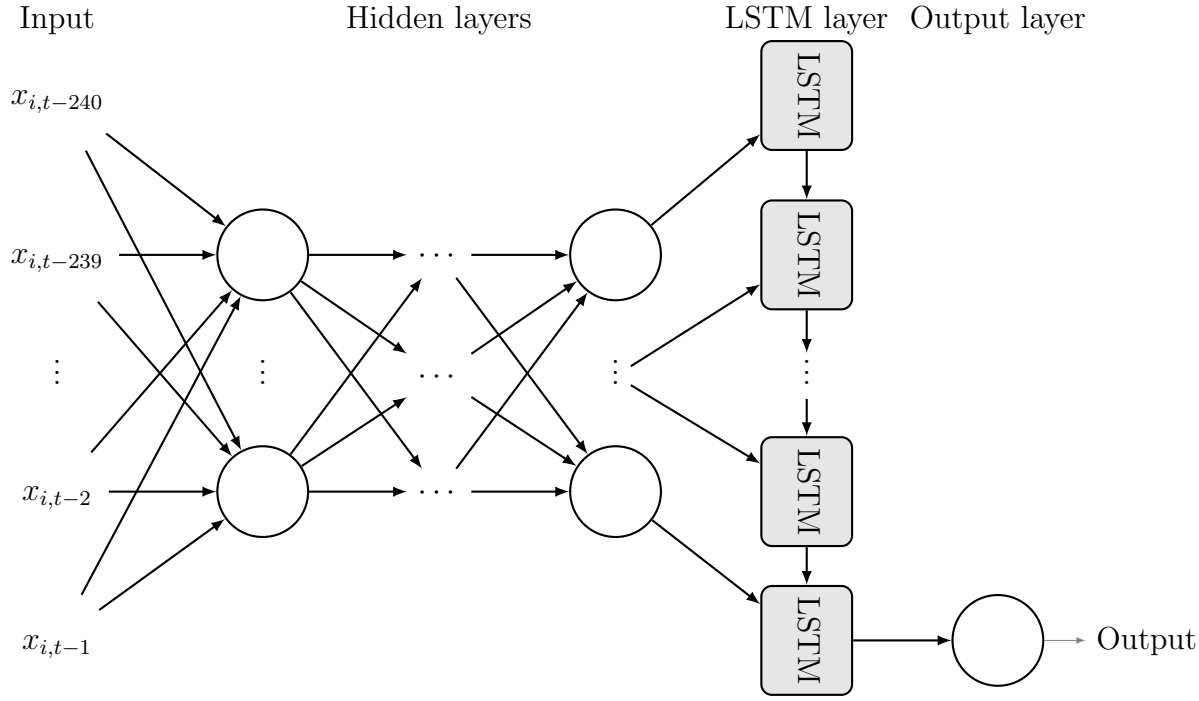
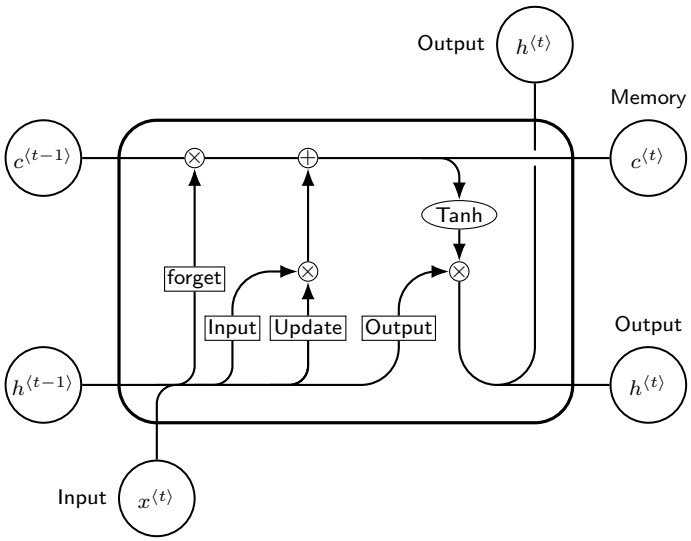


Figure 7: Long Short-term Memory cell. Figure 7 shows a diagram of a Long Short-term Memory cell. xt denotes the input of the LSTM cell, which is where the neural network node is connected. ct and ht denote the memory and the output respectively that are passed to the next LSTM cell. In the last LSTM cell of the LSTM layer only the output is passed to the output node.



The units are described as follows:

1. The **input** unit learns what information is stored in the memory:

$$input_t = \sigma(W_{x,i}x_t + W_{h,i}h_{t-1} + b_i) \tag{27}$$

where $\sigma(\cdot)$ is the softmax function, x_t is the input from node t of the last layer of the neural network, h_{t-1} is the output of the cell of the previous time step and the subscript i denotes the input unit.

2. The **update** unit which learns what proportion of input to add or substract from the memory:

$$update_t = \sigma(W_{x,u}x_t + W_{h,u}h_{t-1} + b_u) \tag{28}$$

where the subscript u denotes the update unit.

3. The **forget** unit learns how long to store the memory:

$$forget_t = \sigma(W_{x,f}x_t + W_{h,f}h_{t-1} + b_f) \tag{29}$$

where the subscript f denotes the forget unit.

4. The **output** unit which gives the memoryless output:

$$output_t = \sigma(W_{x,o}x_t + W_{h,o}h_{t-1} + b_o) \tag{30}$$

where the subscript o denotes the output unit.

The memory of the cell is updated as

$$c_t = forget_t \times c_{t-1} + input_t \times update_t, \tag{31}$$

where $forget_t \times c_{t-1}$ denotes what part of the memory should be forgotten, where $input_t \times update_t$ denotes what proportion of the input should be added to the memory, and where $\times$ denotes the crossproduct.

The output of the cell is computed as

$$h_t = output_t \times tanh(c_t). \tag{32}$$

Each cell outputs the memory and the memory-augmented output. The output and the memory are passed on to the next LSTM cell in the layer. This results in the last LSTM cell containing all relevant information in the sequence, or memory, and passing it to the output node.

4.2 Model Training

The optimization for the trend factor is done using simple linear regression and forecasting. However, the logistic regression, the neural network, and the LSTM are optimized differently, since an analytic answer can not be found for these methods.

Optimization of these models means optimizing the parameters such that a loss function is minimized. This is done by solving

$$\min_{\theta} \frac{1}{M} \sum_{i=1}^{M} \ell(f(\mathbf{x}_i; \theta), y_i) = \min_{\theta} \mathcal{L}(\mathbf{x}_i, y_i; \theta) \tag{33}$$

where M is the number of observations and $\ell(\cdot)$ is a function that measures the distance between the output of the neural network, $f(\cdot)$, and the corresponding target variable, y_i, for observation i.

For $\ell(\cdot)$ I use the binary cross-entropy loss function which is the standard for binary classification problems. The loss function is defined as

$$\ell(f(\mathbf{x}_i; \theta), y_i) = y_i log(f(\mathbf{x}_i; \theta)) + (1 - y_i) log(1 - f(\mathbf{x}_i; \theta)). \tag{34}$$

The output of the classification models $f(\cdot)$ can be regarded as the probability of the label being 0 and thus with this loss function the model is trained to minimize the probability of missclassification.

To solve the optimization I use back propagation with adaptive moment estimation (ADAM) (Kingma & Ba, 2014), which robustifies the optimization by adaptively controlling for the gradient slope in optimization with gradient descent. I also use L_1 regularization (Tibshirani, 1996) on all the weights of the neural network which robustifies the neural network for unkown or out-of-sample data. Furthermore, I use early stopping to avoid overfitting. The model is optimized on training data and evaluated on validation data. I pick the model for which the in-sample and out-of-sample accuracies are as high as possible and are closest together, meaning out-of-sample performance matches in-sample performance. For the initialization of the weights and biases I use a robust method developed by He, Zhang, Ren, and Sun (2015) specifically for rectified activation functions that are not differentiable at zero. They show that drawing weights from a zero-mean Gaussian distribution with standard deviation $\sqrt{\frac{2}{n_l}}$, where n_l is the number of nodes in layer l, the previous layer, leads to faster convergence than other common methods. They also initialize the biases at zero and the standard deviation of the weights of the first hidden layer at one. Furthermore, to make the input more homogenous and

thus facilitating generalization, the inputs are standardized with the cross-sectional mean and standard deviation, and the input subsamples are undersample to avoid bias in the training data. Details on the training of the neural networks and feature engineering are further discussed in Appendix B and Appendix C respectively.

As mentioned, I divide the return data into three parts: the training sample, containing all return data over the period January 2014 to October 2015, which is approximately 60% of return data, the validation sample containing all return data over the period November 2015 and May 2017, which is approximately 20% of return data, and the testing sample, containing all return data over period June 2017 to December 2018, which is also approximately 20% of return data. This three way split is imperative in machine learning. The training sample is used to train the models in batches and the validation sample is used to evaluate the out-of-sample performance. If the out-of-sample performance is not satisfactory the model training is continued with another batch of the training sample. The testing sample is used to evaluate the performance of the models.

4.3 Prediction Performance

For the assessment of the return sign predictions I describe several metrics used in classification, a modified binary classifier correlation coefficient and a modified Diebold-Mariano test for binary classification.

4.3.1 Prediction Metrics

To assess the performance of the return predictions I use the hitrate (HR), the sensitivity (SE) and the specificity (SP). Furthermore, I use the F1-score for which the precicion (PR) is needed. These metrics are calculated as follows

$$HR = \frac{\sum_{i=1}^{N}\sum_{t=1}^{T} I(\hat{y}_{i,t} = y_{i,t})}{NT} \tag{35}$$

$$SE = \frac{\sum_{i=1}^{N}\sum_{t=1}^{T} I(\hat{y}_{i,t} = y_{i,t} = 1)}{\sum_{i=1}^{N}\sum_{t=1}^{T} I(y_{i,t} = 1)} \tag{36}$$

$$SP = \frac{\sum_{i=1}^{N}\sum_{t=1}^{T} I(\hat{y}_{i,t} = y_{i,t} = 0)}{\sum_{i=1}^{N}\sum_{t=1}^{T} I(y_{i,t} = 0)} \tag{37}$$

$$PR = \frac{\sum_{i=1}^{N}\sum_{t=1}^{T} I(\hat{y}_{i,t} = y_{i,t} = 1)}{\sum_{i=1}^{N}\sum_{t=1}^{T} I(\hat{y}_{i,t} = 1)} \tag{38}$$

$$F1 = 2 \cdot \frac{PR \cdot SE}{PR + SE} \tag{39}$$

where $I(\cdot)$ is the indicator function which is set to 1 if the condition is fulfilled and 0 otherwise. The hitrate denotes the proportion of correctly classified predictions. The sensitivity and specificity denote the proportion of the positive returns and negative returns that are correctly classified respectively. The F1-score is the harmonic mean of precision and sensitivity and

As a base prediction model I am using the optimist forecast which predicts all returns to be positive. The hitrate of the optimist forecast gives the proportion of all returns that are positive. Consequently the sensitivity is 100% and the specificity is 0% since no negative returns are predicted. For all other prediction models there usually exists a trade-off between sensitivity and specificity.

4.3.2 Correlations

The correlation term for binary classifiers can not be expressed in the usual sense of the Pearson correlation coefficient since it is developed solely for continuous data. For this purpose I propose a simple binary correlation term, defined as

$$\rho = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[I[\hat{y}_{i,t}^{(1)} = \hat{y}_{i,t}^{(2)}] - I[\hat{y}_{i,t}^{(1)} \neq \hat{y}_{i,t}^{(2)}]\right] \tag{40}$$

where $\hat{y}_{i,t}^{(1)}$ and $\hat{y}_{i,t}^{(2)}$ are the binary predictions for model 1 and 2 respectively for stock i at time t. A correlation of 1 means the predictions of model 1 and 2 are exactly the same and a correlation of -1 means the predictions of model 1 and 2 are exactly the opposite. A correlation of 0 means the predictions of model 1 and 2 are the same on only half of

the predictions which can not be explained by more than chance.

4.3.3 Diebold-Mariano Test

To compare the performance of the return predictions of different models I use the approach of Becker and Leschinski (2018) which is a modification of the Diebold-Mariano test (Diebold & Mariano, 2002). Lets consider two series of forecasts $\{\hat{y}_{i,t}^{(1)}\}_{i=1}^{N}$ and $\{\hat{y}_{i,t}^{(2)}\}_{i=1}^{N}$ for model 1 and 2 at time t respectively. Let the associated prediction errors be denoted as $\{e_{i,t}^{(1)}\}_{i=1}^{N}$ and $\{e_{i,t}^{(2)}\}_{i=1}^{N}$, where $e_{i,t}^{(p)} = (y_{i,t}^{(p)} - \hat{y}_{i,t}^{(p)})^2$ for model p. The set of loss differentials is defined as $\{d_{i,t}\}_{i=1}^{N} = \{e_{i,t}^{(1)} - e_{2,t}^{(2)}\}_{t=1}^{T}$. The null hypothesis for the equal forecasting accuracy is then defined as

$$H_0 : E[d_t] = 0. \tag{41}$$

where $d_t = \sum_i d_{i,t}$. The Diebold-Mariano test statistic is computed as

$$DM = \frac{\bar{d}}{\sqrt{\hat{\gamma}_d(0)/T}} \tag{42}$$

where $\bar{d}$ is the average over the loss differences,

$$\bar{d} = \frac{1}{T}\sum_{t=1}^{T} d_t, \tag{43}$$

and where $\hat{\gamma}_d(0)$ is the autocovariance of the the error differences $d_{i,t}$ at lag 0, which is the variance. Diebold and Mariano (2002) argue that a $k-1$ dependence is a reasonable benchmark for a k-step ahead forecast error, therefore the sample autocovariance in Equation 42 is taken at lag 0. In this instance the Diebold-Mariano test is analogous to testing the difference of two hitrates. The DM test statistic is tested against two-sided alternative hypthosis assuming that it follows a t-distribution with $T-1$ degrees of freedom.

4.4 Changing Exposure and Timing the Market

As described in Section 1.3 we can now use the return classification predictions in conjunction with the order side to adjust the duration or the timing of trades.

To find a suitable duration for a trade I use quantile regressions to find empirically viable long and short durations for each trade given several of its characteristics. This is modelled as follows

$$\tau = \alpha + \beta FDV + \gamma SPR + \delta MC + \varepsilon, \tag{44}$$

where τ is the duration, MC is the market cap of the corresponding stock and α, β, γ and δ are to be estimated. This regressions is executed by optimizing the following loss function

$$\min_{\theta} \frac{1}{M} \left[\sum_{i \in \{i: y_i \geq f(\mathbf{x}_i;\theta)\}} \xi |y_i - f(\mathbf{x}_i;\theta)| + \sum_{i \in \{i: y_i < f(\mathbf{x}_i;\theta)\}} (1-\xi) |y_i - f(\mathbf{x}_i;\theta)| \right] \tag{45}$$

where M is the amount of observations, θ the set of parameters of Equation 44, $f(\cdot)$ is the quantile regression model from Equation 44, $\mathbf{x}_i$ the set of regressors for observation i, and ξ is the target quantile for which the regression is performed. For these quantiles I use 10% and 90%. A quantile higher than 50% puts heavier emphasis on the positive errors of the loss function and thus penalizing positive errors more than negative errors. This results in a fit that purposely overfits to positive errors and underfits to negative errors, and vice versa for a quantile lower than 50%. The regressors in this model are all known ex-ante, so given a prediction of the return classification and the side of an order we can find the 10% longest or shortest durations that are empirically plausible for any trade. If a trade is executed near the end of the day then the longest possible duration is taken to be the time until the end of the day. I make no inference on the duration model, I use it solely as a heuristic to find viable long and short durations.

To evaluate the improvements in transaction costs from the market timing I simply take the timing costs for trades that are delayed and estimate the transaction costs on the delayed day and add those together. This will give an estimate of the timing costs plus transaction costs and can be compared to the actual transaction costs.

5 Results

Here I show the results of this thesis. In Section 5.1 I show the results of the transaction cost models and their robustness, in Section 5.2 I show the performance of the predictors for return classification and in Section 5.3 I show the improvements these predictors can make on transaction costs by changing the duration of these trades or by delaying these trades according to the return predictions.

5.1 Transaction Cost Models

The regression results for the models described in Section 3.1 are presented in Table 4. The top row denotes the coefficients of each model, the robust R^2_{LAD} measure and the mean absolute error (MAE) for each model. Robust standard errors are presented in brackets underneath the coefficient values and all significant coefficient values with $p < 0.01$ are in bold print.

Table 4: Regression results for transaction cost models. The models in the leftmost column correspond to the models in Equations 7, 8, 9, 10, 11, 12, 13, 14, 15 and 16. The values in brackets are the robust standard errors and all significant coefficient values with $p < 0.01$ are in bold print. The right-most column shows the R^2 and MAE of each model.

Model	α	β	γ	δ_1	δ_2	δ_3	δ_4	η	R^2_{LAD}	MAE
A	**1,79**	**-70,56**	**0,42**						0,29	68,46
	(0,86)	(6,36)	(0,04)							
B	2,21	**-65,38**	**0,43**					**-2.025**	0,29	68,44
	(0,80)	(6,49)	(0,04)					(401,75)		
C	0,73	**-57,43**	**0,42**	**-3.563**					0,93	54,74
	(0,99)	(4,91)	(0,04)	(42,33)						
D	1,88	**-60,54**	**0,38**		**-28,92**				0,88	62,08
	(1,57)	(5,24)	(0,05)		(0,33)					
E	1,16	**-50,32**	**0,44**	**-3.565**				**-1.505**	0,93	54,72
	(0,93)	(5,48)	(0,05)	(47,56)				(340,42)		
F	2,21	**-56,25**	**0,39**		**-28,91**			**-1.569**	0,88	62,06
	(1,48)	(4,70)	(0,05)		(0,34)			(366,47)		
G	0,61	**-40,70**	**0,35**			**-7.981**			0,94	45,83
	(0,80)	(2,92)	(0,04)			(52,31)				
H	0,53	**-57,50**	**0,41**				**-75,09**		0,89	58,50
	(0,77)	(5,14)	(0,04)				(0,79)			
I	1,03	**-35,20**	**0,37**			**-7.988**		**-1.968**	0,94	45,80
	(0,66)	(3,01)	(0,04)			(47,14)		(288,53)		
J	1,12	**-50,80**	**0,42**				**-75,05**	**-1.959**	0,89	58,47
	(0,82)	(5,98)	(0,05)				(0,73)	(370,22)		

We see that the exponent, γ, of the size term is close to $0,5$ for all models, which is close to the square root rule as posed in Torre (1997). However, this result is clearly different to that of Almgren et al. (2005) or Zarinelli et al. (2015). The 3/5 power law from Almgren et al. (2005) is steeper overall and the logarithmic dependence from Zarinelli et al. (2015) is steeper for small values. This shows that the impact of size on transaction costs generally follows a concave shape but does not have a definitive shape for all order data. The concave dependency of transaction costs on order size means that transaction costs increase at a decreasing rate with increasing order size. For models D, F, G and I this is more pronounced since the exponent is smaller.

The slope coefficient for size is negative for all models which follows logically from the definition of transaction costs, the bigger the trade the higher the transaction costs generally will be. Furthermore, the constant is only significantly different from zero for model A. In all other models this effect is diminished by the presence of the other regressors.

As informally shown in Section 2.4 the slope coefficient for the market exposure terms is negative for all models. This means that if negative market exposure increases the transaction costs decrease with δ_j basis points, where j denotes the corresponding market exposure term. The difference between the slope coefficient for market exposure of the models C and D, E and F, G and H, and I and J is large due to the return being incorporated in the sign function. The return is incorporated in the sign function for model D, F, H and J and thus is either 1 or -1, however, for C, E, G and I the raw return is used which is of a vastly smaller magnitude.

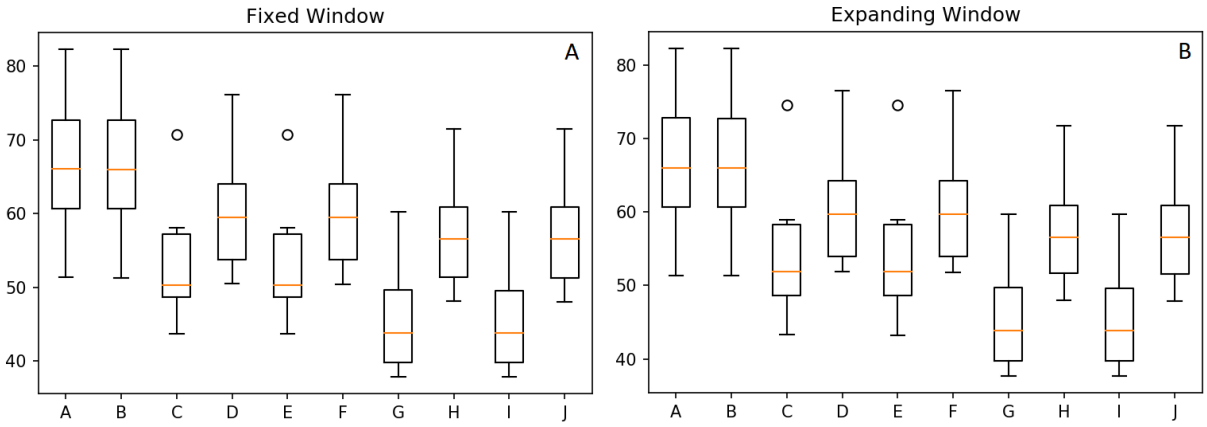
Furthermore, we see that including the entire return instead of just the sign of the return increases the goodness of fit with about 5%. This increase is visible between models C and D and models E and F. We also see that including the duration in the market exposure term increases the goodness of fit with about 1% between models C and G, D and H, E and I, and F and J.

We see for all models that coefficient for spread is significantly different from zero, however that it adds barely any explanatory value in all of the models in terms of R^2 and MAE. However, we see that the market exposure terms increase the explanatory power of the models significantly. Whereas the R^2 and MAE of the model with only size is 0,29 and approximately 68 respectively, the market exposure models increase the R^2 and decrease the MAE up to and over 0,90 and 45 respectively. This shows that, in any form, the market exposure has significant influence on the transaction costs of orders.

5.1.1 Robustness

To test the robustness of the models I split the sample data into eleven folds of approximately four months each and apply the walkforward validation and the chow-test on the subsamples. Figure 8 shows the MAPE's of the walkforward validation with a fixed window and expanding window for all models and Figure 9 shows the F-statistics for the chow-test between each consecutive subsample for all models.

Figure 8: Boxplots of the walk forward validation for fixed and expanding window. The left boxplot contains the MAPE's for the fixed window walk forward validation, the right boxplot contains the MAPE's for the expanding window walk forward validation.



In Figure 8 we see the same pattern of MAPE's as for the MAE in the regression results. Overall the MAPE decreases the more market exposure information is included in the model, however, including spread does not meaningfully increase the prediction accuracy. We see that model C and E have the lowest amount of fluctuation over time and are thus most robust over time. Interestingly for all models the MAPE's do not change with a larger estimation window. This shows that incorporating more information does not meaningfully improve the prediction accuracy. The difference of the MAPE's between fixed and expanding window walk forward validation are so small they are not visible for less than two decimals. This suggests that the functional dependence of all models is persistent in the data over time.

Figure 9: Boxplot for the F-statistics of the LAD Chow-test for each model. The degrees of freedom of the F-distributions are (3, 15112) for model A, (4, 15110) for models B, C, D, G, and H, and (5, 15118) for models E, F, I, and J. H_0 is rejected with $p < 0.01$ for all F-statistics of all models.

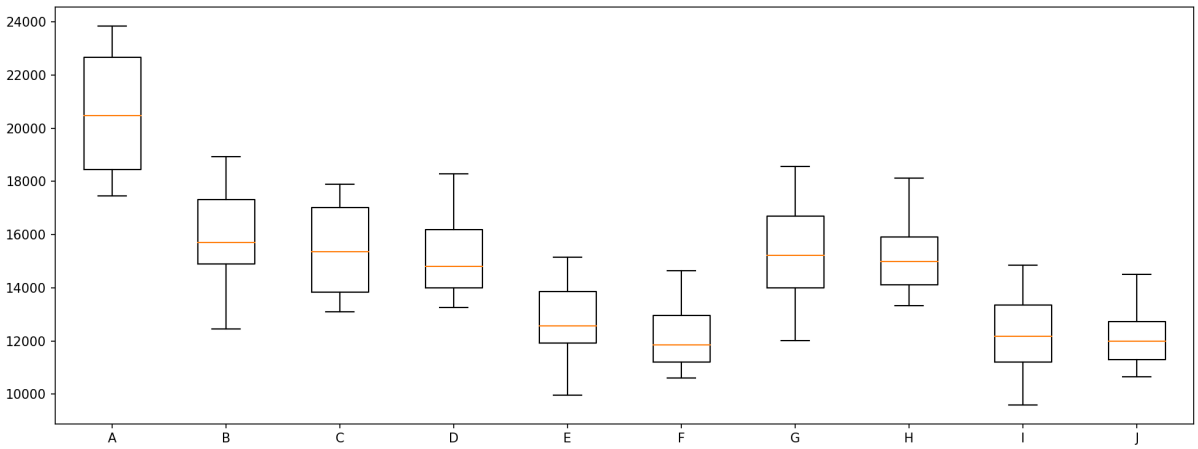


Figure 9 shows that there are no signs of structural breaks between the consecutive periods for any of the models. The boxplot for the F-statistics of the Chow test for each

model is far above the cutoff point for each respective degrees of freedom. This is a strong indication that the models hold up over time and on smaller subsamples of the data. This shows that all of these models are highly robust over time. Interestingly, including spread in each model lowers the robustness of the model since it lowers the degrees of freedom of the model and barely increases the predictive performance. This is also visible from the lower F-statistics for the models and their respective counterpart with spread.

5.2 Predicting Returns

Table 5 shows the performance of the return direction predictions for all methods. Furthermore, two simple predictors are added which are 'Optimist' which predicts all returns to be positive and 'Lazy' which predicts all returns to be so that the accompanying exposure is positive, thus incentivising delaying the trade. These naive predictors are used as a benchmark. For each predictor two panels are shown. The top panel shows the confusion matrix, where the columns indicate which direction the predictor predicted and where the rows indicate the true direction of the return, and the bottom panel shows some performance measures. The neural network model I have chosen has six hidden layers with 220 nodes per layer and the LSTM model I have chosen has ten memory nodes per LSTM cell. Details on the choice of the topology are given in Appendix D. The classification results for the training and validation sample are given in Appendix E.

We see in Table 5 that, as per the definition, the optimist only predicts positive returns. Here the rate of false positives, 0,496, and the rate of true positives, 0,504, are the distribution of negative and positive returns respectively since the accuracy is the sum of true positive rate and true negative rate. This gives an accuracy of 0,504 and a specificity and sensitivity of 0 and 1 respectively. Furthermore this predictor has the highest F1-score at 0,67 of all predictors which is driven by the high sensitivity. The lazy predictor has a fairly even distribution of predictions, also hitting mostly positive return predictions. This gives an accuracy of 0,526 and a F1-score of 0,546. These high metrics can probably explained by a high amount of sells happening on positive return days and a high amount of buys happening on negative return days. This causes timing costs to be favorable and thus for a trade to be delayed. This might be caused by the portfolio managers buying stock that are becoming cheaper and selling stocks which are gaining a premium, however the true nature can not be discerned. The linear predictor, or trend-factor, is not as powerful as for daily returns as for monthly returns. The accuracy is just slightly lower than that of a coin toss and mostly caused by misspecifying positive

Table 5: Classification performance of each return predictor. Two panels are shown for each predictor. The top panel shows the confusion matrix where the columns denote the predicted direction and the rows denote the trude direction of the returns. The bottom panel shows several performance metrics.

	Optimist		Lazy		Linear	
	Down	Up	Down	Up	Down	Up
Down	0	0,496	0,241	0,255	0,271	0,225
Up	0	0,504	0,219	0,285	0,28	0,224
Accuracy	0,504		0,526		0,495	
Specificity	0		0,486		0,546	
Sensitivity	1		0,565		0,444	
F1-score	0,67		0,546		0,469	
	Logistic		NN		LSTM	
	Down	Up	Down	Up	Down	Up
Down	0,094	0,468	0,206	0,356	0,327	0,235
Up	0,02	0,418	0,114	0,324	0,221	0,217
Accuracy	0,512		0,53		0,544	
Specificity	0,167		0,366		0,582	
Sensitivity	0,954		0,74		0,495	
F1-score	0,488		0,58		0,632	

returns. The high rate of true negatives and false positives shows that this predictor is biased towards negative returns, consequently causing higher specificity. The logistic predictor is biased towards predicting positive returns, even though the input data is standardized and the batches are resampled to avoid a biased sample. This leads to an accuracy of 0,512 caused by a low specificity of 0,167 and a high sensitivity of 0,954. The neural network has a lower prediction bias with a more evenly distributed confusion matrix. Still, it seems to predict mostly positive returns, however, correctly predicting more negative returns than the logistic predictor. This leads to a higher accuracy of 0,53 caused mostly by a high sensitivity of 0,74. The LSTM performs very well and has the most evenly distributed confusion matrix of the non-naive methods. It has the highest prediction accuracy with both a high specificity of 0,582 and a moderate sensitivity of 0,495. This shows that the LSTM is the best predictor with the lowest bias of all the predictors. These findings are in line with the findings of Fisher (1961), who find that LSTM's perform well on return prediction with accuracies of up to 55%, and the findings of Becker and Leschinski (2018), who find that logistic regression and neural networks perform similarly on return prediction with accuracies up to 52%.

Table 6 shows the distribution of the predicted returns. For each predictor the distribution of the returns that are correctly classified and the returns that are missclassified are

given. This gives insight into the magnitude of the returns and the bias of the predictor.

Table 6: Distribution of predicted returns. This table denotes for each predictor the distribution of the returns that are correcty classified and the distribution of the returns that are incorrectly classified. The top columns for each predictor show several statistics and the bottom columns show the quantiles for these distributions.

	Optimist		Lazy		Linear	
	Hit	Miss	Hit	Miss	Hit	Miss
Mean	1,33%	-1,37%	0,06%	-0,09%	-0,19%	0,16%
Std.dev.	1,48%	1,50%	1,98%	2,04%	2,05%	1,96%
Skewness	3,94	-3,85	-0,17	0,12	-0,02	-0,01
Kurtosis	34,38	34,12	11,26	17,13	17,71	10,50
Quantiles						
95%	3,94%	-0,07%	3,02%	2,81%	2,69%	3,12%
75%	1,74%	-0,41%	1,03%	0,81%	0,77%	1,06%
50%	0,92%	-0,95%	0,13%	-0,10%	-0,15%	0,16%
25%	0,41%	-1,84%	-0,89%	-0,98%	-1,11%	-0,76%
5%	0,08%	-4,08%	-3,02%	-3,04%	-3,24%	-2,81%
	Logistic		NN		LSTM	
	Hit	Miss	Hit	Miss	Hit	Miss
Mean	0,17%	-0,14%	-0,21%	0,27%	-0,36%	0,67%
Std.dev.	1,98%	2,00%	1,98%	1,98%	1,97%	1,86%
Skewness	0,02	-0,07	-0,08	0,03	-0,21	0,45
Kurtosis	11,84	16,53	11,67	17,35	14,80	16,50
Quantiles						
95%	3,02%	2,77%	2,69%	3,14%	2,48%	3,54%
75%	1,05%	0,82%	0,74%	1,14%	0,56%	1,40%
50%	0,20%	-0,13%	-0,19%	0,30%	-0,33%	0,57%
25%	-0,73%	-1,06%	-1,11%	-0,61%	-1,22%	0,02%
5%	-2,80%	-3,16%	-3,21%	-2,70%	-3,36%	-2,05%

The optimist predictor hits all positive returns and misses all negative returns. Consequently the distribution of positive and negative returns are displayed under the hit and miss columns of the optimist predictor respectively. We see that on average the magnitude of the positive returns is lower than that of the negative returns, however, the rest of the distribution is almost mirrored. The lazy predictor is slightly biased towards positive returns, however the mean for both hits and misses close to zero indicating a low bias. Furthermore, the distributions of the returns are very similar for both hits and misses. This is a desirable trait in a predictor, indicating that the bias of this predictor is low. The linear and logistic predictors have almost opposite distribution for both hits and misses. The linear predictor has a slight bias towards negative returns and the logistic predictor has a slight bias towards positve returns. Furthermore, we see that the distribution of the

returns of the logistic predictions is not as heavily biased as the confusion matrix in Table 5 would lead us to expect. We see that the Neural Network and the LSTM, the two best predictors as seen in Table 5, have similar distributions of predicted returns. We see that both are biased towards negative returns with the LSTM having a heavier bias than the Neural Network. Given the confusion matrix of the Neural Network, with mostly correct classifications on positive returns, this suggest that the magnitude of the negative returns correctly classified are greater than the magnitude of the correctly classified positive returns. This is a favorable characteristic of the predictor. High accuracy is important, however, high magnitude in correctly classified returns is also important since the higher the magnitude of the return the bigger the impact is on transaction costs. Thus correctly predicting high magnitude returns will have a high impact on transaction cost reduction.

5.2.1 Correlation of Predictors

Table 7 shows the binary correlation coefficients between the predictors. A star indicates whether the predictors have significantly different predictions, tested with the DM-test at p-value of 0.01. Predictors with low correlation could be used in ensembled to increase overall performance of the predictions. Furthermore, the correlations imply whether the predictors use the same information to predict returns.

Table 7: Correlations between predictors. This table shows the correlation between all predictors. A star denotes whether two predictors have significantly different predictions according to the DM-statistic.

	Optimist	Lazy	Linear	Logistic	NN	LSTM
Optimist						
Lazy	0,0797*					
Linear	-0,1027*	0,0145*				
Logistic	0,7896*	0,0637*	-0,0792*			
NN	0,1130*	0,0263*	0,0529*	-0,3126*		
LSTM	-0,0264*	-0,0295*	0,0190*	-0,1036*	0,1245*	

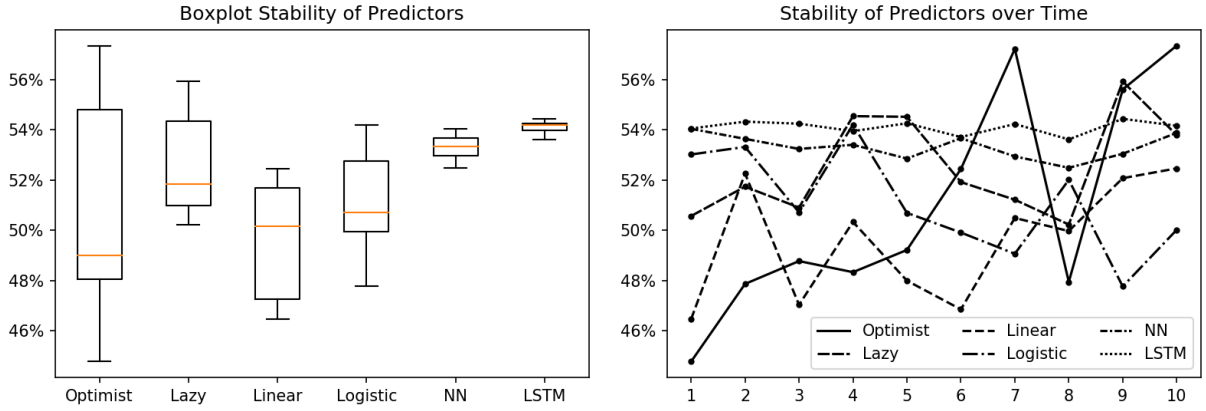
We see that the optimist predictor has a high positive correlation with the logistic predictor. This shows that the logistic predictor, although it has decent performance, uses only little more information than the naive classifier. The other predictors have a low correlation with the optimist predictor showing they use more information than the naive classifier. The lazy predictor has low correlation with all the other predictors, which follows logically from the definition of the lazy predictor since its a more sophisticated naive classifier. The linear predictor has fairly low correlations across the board but given its low performance has little value to add to the other predictors. The logistic predictor, the neural network and the LSTM have low to medium correlations with fairly strong

performance suggesting these predictors could be used jointly. Furthermore, we see that all methods have significantly different predictions showing that all predictors are unique.

5.2.2 Stability of Predictors

Figure 10 shows the stability of the return predictors. The left figure shows the boxplot of the accuracies of the predictors in ten consecutive subsamples in the test sample. The right figure shows the stability over time of the predictors in these ten subsamples. The stability of the predictors over time in the right plot is reflected in the boxplot on the left.

Figure 10: Figure of the stability of the return predictors. The left figure shows the boxplot of the accuracies of the predictors in ten consecutive subsamples of the test sample. The right figure shows the stability over time of the predictors in the consecutive subsamples.



We see that the stability of the predictors increases with its performance. In the right plot we see that the optimist predictor, the solid line, tracks the percentage of positive returns over time. This fluctuates strongly over time and is reflected in the stability in the boxplot. The stability of the lazy predictor (densely dashed line), the linear predictor (loosely dashed line), and the logistic predictor (loosely dash-dotted line), are roughly the same as we can see from the boxplot. Furthermore, we see that, from these three, the lazy predictor performs most accurately over time. The neural network and the LSTM have a high stability around their respectively high accuracies. This is reflected in the stability over time where we see little fluctuations of accuracy over time. Furthermore, judging from the boxplot, we see that the LSTM performs slightly better in terms of stability. Interestingly, the accuracy of the predictors does not decrease as the testing subsample progresses further from the training and validation sets. This is especially the case for the neural network and the LSTM. This shows that these models capture information that is persistent over time and shows that these models are robust to future information.

5.3 Improved Transaction Costs

In this section I show the results in transaction costs and timing costs from changing the exposure of trades and from delaying trades respectively. To reiterate, the predictors from Section 4.1 give a return direction prediction for the stocks that are to be traded. Together with the side of the trade this gives a prediction of the sign of the market exposure of the trade. If the prediction of the market exposure is negative we minimize the exposure by replacing the actual duration with a low duration. If the prediction of the market exposure is positive we maximize the exposure of the trade by replacing the actual duration with a high duration, or we delay the trade. This should improve the transaction costs. We can estimate this change in transaction costs using the models from Section 3.1 and entering the long or short duration. If the duration corresponds with the exposure the transaction costs improve, however if it does not correspond the transaction costs deteriorate. Details on the quantile regression and estimated duration for each trade are given in Appendix F.

5.4 Changing Duration

Table 8 shows the transaction costs for all different strategies. The true transaction costs are given in the column denoted by Truth, the column Min. denotes the transaction costs if the duration for all trades was minimized, the column Max. denotes the transaction costs if the duration for all trades was maximized and the column Best denotes the transaction costs in the case all trades had perfect market exposure. All transaction costs are expressed in basis points.

The first thing to note is that by minimizing the duration for all trades we would have improved the transaction costs by about two basis points. This is probably due to the skewness towards buys in this period as well as a skewness towards positive returns within this period, causing on average negative market exposure for all trades. This also explains the increased transaction costs when all trades have increased duration. Another interesting observation is the distribution of the minimized and maximized duration transaction costs. We see that the transaction costs with minimized duration have a much tighter distribution with a lower standard deviation and tail percentiles that are much closer to zero. On the other hand the transaction costs with maximized duration have a much wider distribution with high standard deviation and long tails on both sides. This follows naturally from the definition of market exposure as we have seen in Section 2.4. Moreover, we see that in the best case we can gain approximately 27 basis points per trade

Table 8: Distribution of improved transaction costs for changed duration. This table shows the distribution of the transaction costs in case the duration of the trade is changed according to several heuristics or the return predictors. The top panel shows several statistics and the bottom panel shows the quantiles of the distribution. The columns denote the heuristics 'Truth', which are the transaction costs of the trades without changing the duration, 'Min.', which takes the lowest possible duration for all trades, 'Max.', which takes the highest possible duration for all trades, 'Best', which applies the optimal duration to each trade. The other columns denote the aforementioned predictors.

	Truth	Min.	Max.	Best.	Optimist
Mean	-11,71	-9,43	-14,28	26,65	-11,79
Std.dev.	116,20	73,32	144,10	105,54	114,15
Skewness	0,16	-1,55	-0,04	2,62	-0,48
Kurtosis	18,27	45,96	17,77	35,83	25,79
Quantiles					
95%	151,72	84,89	186,33	190,66	136,57
75%	32,18	21,15	47,82	58,83	30,01
50%	-6,40	-6,00	-9,28	14,09	-7,53
25%	-54,10	-36,15	-71,90	-19,50	-49,07
5%	-189,11	-113,95	-228,62	-96,30	-175,05

	Linear	Logistic	NN	LSTM
Mean	-12,02	-8,62	-7,43	-0,87
Std.dev.	112,75	114,58	114,71	114,61
Skewness	-0,38	0,12	-0,27	0,51
Kurtosis	30,19	28,51	26,03	31,39
Quantiles				
95%	136,32	144,76	146,67	155,64
75%	29,74	33,76	34,01	40,10
50%	-7,59	-5,37	-4,31	-0,52
25%	-50,37	-48,15	-46,67	-41,73
5%	-172,18	-172,02	-168,28	-157,40

by improving the market exposure of trades. This is an improvement of approximately 38 basis points on average per trade. As we see from the distribution this improvement is caused by a shift in the distribution to the positive side with a fat tail on the positive side and a short tail on the negative side, however negative transaction costs are still inevitable. The optimist predicts only positive returns and hits just over 50% of its predictions. Together with a skewness towards buys in the sample this would lead on average to minimized market exposure for most trades. However, since the hitrate is only slightly higher than 50% and as we have seen in Table 6 that the magnitude of negative returns is larger than that of positive returns, this lead to transaction costs that are slightly worse on average than the truth. We also see this from the negative skewness and the distribution both with more emphasis on the negative side. The linear predictor has an accuracy of slightly less than 50% and therefore causes transaction costs to be

slightly worse. This is also visible from the negative skewness and, although the distribution is more tight, this is probably caused by the distribution being hemmed in by the artifically set boundaries of the durations. The logistic predictor is where we begin to see proper improvements of the transaction costs. With an improvement of more than three basis points the overall distribution seems to be similar to the truth, be it more tight. However, we see that the kurtosis of the logistic predictor is much higher which indicates fatter tails probably causing the better transaction costs on average. The Neural Network has a slightly better improvement at more than four basis points from the truth with a longer tail on the positive side and a shorter tail on the negative side. The LSTM has a significant improvement with close to zero transaction costs per trade on average. It has the most favorable distribution of transaction costs of all the predictors with the longest positive tail and the shortest negative tail, positive skewness and high kurtosis. Overall this shows that changing the exposure of trades using modern prediction techniques can improve transaction costs. Furthermore, it shows clearly that better prediction accuracy is repaid in improved transaction costs.

5.4.1 Delaying Trades

Table 9 contains the distributions of the change in timing costs that would be induced by delaying trades according to the return predictions. To estimate the transaction costs, all variables are taken on the delayed day. The table does not have a Truth column since no timing costs are taken into account when trading as of yet. Furthermore, this table, as opposed to Table 8, does contain a column for the Lazy predictor since this predictor by definition implies that all trades are delayed which has no significance when changing the duration. All values except the standard deviation, skewness and kurtosis are denoted in percentages. For comparison 1.00% is the same as 0,01.

The first column denotes the distribution of timing costs if all trades with positive timing costs are delayed. This shows that the highest achievable gain in timing costs is 1,43% on average per trade. Logically the entire distribution resides on the positive side with a long tail positive tail and a short negative tail. Furthermore, we see that the optimist, lazy and linear predictor all have very similar distributions for gain in timing costs. Delaying trades using either of these predictors will cause a loss of 0,09% per trade due to timing costs. Although the predictive performance of these predictors varies, this shows that approximating timing costs with returns does not necessarily yield similar results in timing costs as in binary classification. The timing costs incorporate the price

Table 9: Distribution of improved transaction costs for delayed trades. This table shows the distribution of the improvement of timing cost according to several heuristics or the return predictors. The top panel shows several statistics and the bottom panel shows the quantiles of the distribution. The columns denote the heuristics 'Best', which is the best possible timing cost improvement that is possible, 'Optimist', which is the improvement of timing cost if all returns are predicted to be positive, and 'Lazy.', which is the change in timing cost if all trades are delayed. The subsequent columns denote the distribution of timing costs given the market exposure predictions of the predictors.

	Best possible	Optimist	Lazy	Linear	Logistic	NN	LSTM
Mean	1,43%	-0,09%	-0,09%	-0,09%	0,00%	0,06%	0,32%
Std.dev.	0,019	0,021	0,020	0,020	0,020	0,020	0,020
Skewness	11,328	0,344	0,367	0,696	0,161	0,461	0,899
Kurtosis	330,045	13,030	13,872	13,384	14,687	14,648	17,608
Quantiles							
95%	4,27%	2,82%	2,84%	2,78%	2,93%	3,00%	3,27%
75%	1,87%	0,87%	0,83%	0,82%	0,92%	0,94%	1,15%
50%	0,96%	-0,05%	-0,08%	-0,10%	0,03%	0,08%	0,29%
25%	0,42%	-1,06%	-1,02%	-1,03%	-0,93%	-0,87%	-0,60%
5%	0,08%	-3,21%	-3,06%	-3,02%	-2,97%	-2,89%	-2,59%

change between the moment an investment decision is made and the next open price. This means that part of the daily price change has already occurred and thus a portion of the informational edge of the predictors has already dissolved causing the predictors to not necessarily perform as well on timing costs as on market exposure. We see this with the logistic predictor which, although it performs well in the return classification, has a zero percent improvement in timing costs. The neural network, with an accuracy of 53%, causes an improvement in timing costs with 0,06% per trade. We see that the distribution of timing costs per trade is shifted to the positive side compared with the logistic predictor. This is even more so the case with the LSTM predictor which causes an improvement of 0,32% timing costs per trade. All predictors have a standard deviation of about 0,02 or 2%, so although the neural network and the LSTM cause positive timing costs the risk reward is lower than one and might thus not be attractive from a risk management perspective.

6 Conclusion and Discussion

In this research I combine the findings of Perold (1988), Israelov and Katz (2011) and Kissell and Malamut (2007), together with a broad range of literature into machine learning for daily return prediction, into a novel approach that can be applied to any existing long term investment strategy with the use of high frequency trading signals.

Since institutional trading takes time orders of any significant magnitude are usually

executed in parts and are thus subjected to changing market prices, which I call the market exposure of a trade. Using high frequency information, such as a daily prediction of the return direction, a trader can increase or decrease the duration of a trade and thus the market exposure. This leads to a more favorable price on average for the trade and consolidates the goals of the portfolio manager and the trader. I show that the market exposure drives a large part of transaction costs of existing trades and that this finding is robust over time. Furthermore, I show that the impact of market exposure on transaction costs is linear with market exposure and that the finding of concave impact of order size on transaction costs of earlier research still holds in the presence of market exposure.

For the return direction prediction I extend the research of Han et al. (2016) who develop a trend factor by regressing moving averages to future returns and using this trend factor to predict monthly return directions. I apply the same technique to daily returns and show that the results are not in line with the original research. This indicates that the information that is picked up in the monthly trend factor contains mostly low frequency information that is only useful in the long term. Furthermore I follow the research of Qiu and Song (2016), Bao et al. (2017), Selvin et al. (2017), Becker and Leschinski (2018) and Fischer and Krauss (2018) by applying several popular machine learning techniques, namely logistic regression, neural networks and LSTM's, to daily return prediction. I show that logistic regression performs better than the daily trend factor in binary classification of return directions by using up to 240 lagged daily returns. Furthermore I show that neural networks and LSTM's respectively increase the classification performance indicating that cross-temporal interactions of the lagged returns and recurring patterns in the returns carry relevant information for classifying daily returns. This is in line with existing research on machine learning techniques for return direction classification.

Subsequently I show that the return predictions can be used to change the duration of trades and thus the market exposure of these trades. I show that better classification performance leads to better performance in transaction cost improvement. Up to 10 basis points per trade can be saved by changing the market exposure of trades.

I follow the definition of Perold (1988) that implementation shortfall of portfolio implementation is the sum of transaction costs and opportunity costs. Since market exposure has an impact on transaction costs we can extend the logic of market exposure to timing costs. Return direction predictions give us the option to trade faster or slower to improve market exposure of a trade, however, instead of trading slower we could also trade later. The return prediction thus essentially becomes a prediction of timing costs. I show that the same high frequency trading signals can be used to delay trades and improve the

timing costs of trades. Since trades are not executed at the start of the day the return predictions do not necessarily fully coincide with the magnitude of the timing costs and I show that the performance of some return classifiers is strongly diminished by this. However, the neural network and LSTM still perform well and achieve up to a 32 basis points improvement in timing costs per trade on average.

Overall this shows that this approach successfully implements short term information in the favor of the portfolio implementation process. Existing machine learning techniques are shown to be effective in return classification and subsequently in the use of timing cost prediction. Furthermore this approach is readily implementable, however, the predictions for return direction are a coarse approximation of the timing costs given that trades might not happen well after the biggest change in price over the day has already transpired. Therefore more fine tuned predictions or heuristics might be preferable in handling those trades.

References

Almgren, R. (2003). Optimal execution with nonlinear impact functions and trading-enhanced risk. *Applied mathematical finance*, *10*(1), 1–18.

Almgren, R., & Chriss, N. (2001). Optimal execution of portfolio transactions. *Journal of Risk*, *3*, 5–40.

Almgren, R., Thum, C., Hauptmann, E., & Li, H. (2005). Direct estimation of equity market impact. *Risk*, *18*(7), 58–62.

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, *12*(7).

Becker, J., & Leschinski, C. (2018). Directional predictability of daily stock returns. *Hannover Economic Papers (HEP)*, *624*.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*(Feb), 281–305.

Berkson, J. (1944). Application of the logistic function to bio-assay. *Journal of the American Statistical Association*, *39*(227), 357–365.

Bertsimas, D., Hummel, P., & Lo, A. W. (2000). Optimal control of execution costs for portfolios. *Computing in Science & Engineering*, *1*(6), 40.

Bertsimas, D., & Lo, A. W. (1998). Optimal control of execution costs. *Journal of Financial Markets*, *1*(1), 1–50.

Chan, L. K., & Lakonishok, J. (1995). The behavior of stock prices around institutional trades. *The Journal of Finance*, *50*(4), 1147–1174.

Chow, G. C. (1960). Tests of equality between sets of coefficients in two linear regressions. *Econometrica: Journal of the Econometric Society*, 591–605.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, *2*(4), 303–314.

Diebold, F. X., & Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business & economic statistics*, *20*(1), 134–144.

Dielman, T. E. (1986). A comparison of forecasts from least absolute value and least squares regression. *Journal of Forecasting*, *5*(3), 189–195.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, *270*(2), 654–669.

Fisher, W. D. (1961). A note on curve fitting with minimum deviations by linear programming. *Journal of the American Statistical Association*, *56*(294), 359–362.

Furno, M. (2014). Quantile regression estimates and the analysis of structural breaks. *Quantitative Finance*, *14*(12), 2185–2192.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. *AISTATS 2011*, 315–323.

Gu, S., Kelly, B., & Xiu, D. (2019). Empirical asset pricing via machine learning. *National Bureau of Economic Research*(No. w25398).

Hahnloser, R. H., & Seung, S. H. (2001). Permitted and forbidden sets in symmetric threshold-linear networks. In *Advances in neural information processing systems* (pp. 217–223).

Han, Y., Zhou, G., & Zhu, Y. (2016). A trend factor: Any economic gains from using information over investment horizons? *Journal of Financial Economics*, *122*(2), 352–375.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the ieee international conference on computer vision* (pp. 1026–1034).

Heston, S. L., Korajczyk, R. A., Sadka, R., & Thorson, L. D. (2011). Are you trading predictably? *Financial Analysts Journal*, *67*(2), 36–44.

Hinton, G., Srivastava, N., & Swersky, K. (2012). Overview of mini-batch gradient descent. *Neural Networks for Machine Learning*, *575*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), 359–366.

Huber, P. J., et al. (1967). The behavior of maximum likelihood estimates under non-standard conditions. In *Proceedings of the fifth berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 221–233).

Israelov, R., & Katz, M. (2011). To trade or not to trade? informed trading with short-term signals for long-term investors. *Financial Analysts Journal*, *67*(5), 23–36.

Keim, D. B., & Madhavan, A. (1997). Transactions costs and investment style: an inter-exchange analysis of institutional equity trades. *Journal of Financial Economics*, *46*(3), 265–292.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kissell, R. (2006). The expanded implementation shortfall: Understanding transaction cost components. *The Journal of Trading*, *1*(3), 6–16.

Kissell, R., & Malamut, R. (2007). Investing and trading consistency: Does vwap compromise the stock selection process? *The Journal of Trading*, *2*(4), 12–22.

Lillo, F., Farmer, J. D., & Mantegna, R. N. (2003). Econophysics: Master curve for price-impact function. *Nature*, *421*(6919), 129.

Linton, O., & Whang, Y.-J. (2007). The quantilogram: With an application to evaluating directional predictability. *Journal of Econometrics*, *141*(1), 250–282.

Loeb, T. F. (1983). Trading cost: the critical link between investment information and results. *Financial Analysts Journal*, *39*(3), 39–44.

Mackintosh, P., & Ewen, G. (2008). Market commentary: Estimating execution costs. *Credit Suisse White paper*.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

McKean, J. W., & Schrader, R. M. (1984). A comparison of methods for studentizing the sample median. *Communications in Statistics-Simulation and Computation*, *13*(6), 751–773.

McKean, J. W., & Sievers, G. L. (1987). Coefficients of determination for least absolute deviation analysis. *Statistics & Probability Letters*, *5*(1), 49–54.

Perold, A. F. (1988). The implementation shortfall: Paper versus reality. *Journal of Portfolio Management*, *14*(3), 4.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, *12*(1), 145–151.

Qiu, M., & Song, Y. (2016). Predicting the direction of stock market index movement using an optimized artificial neural network model. *PloS one*, *11*(5).

Selvin, S., Vinayakumar, R., Gopalakrishnan, E., Menon, V. K., & Soman, K. (2017). Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)* (pp. 1643–1647).

Stanzl, W., & Huberman, G. (2000). Arbitrage-free price-update and price-impact functions. *Yale ICF and Yale SOM Working Paper No. 00-20*.

Sutton, R. (1986). Two problems with back propagation and other steepest descent learning procedures for networks. In *Proceedings of the eighth annual conference of the cognitive science society, 1986* (pp. 823–832).

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288.

Torre, N. (1997). Barra market impact model handbook. *BARRA Inc., Berkeley*.

Wagner, W. H., & Banks, M. (1992). Increasing portfolio effectiveness via transaction cost management. *Journal of Portfolio Management*, *19*(1), 6.

Wagner, W. H., & Edwards, M. (1993). Best execution. *Financial Analysts Journal*, *49*(1), 65–71.

Zarinelli, E., Treccani, M., Farmer, J. D., & Lillo, F. (2015). Beyond the square root: Evidence for logarithmic dependence of market impact on size and participation rate. *Market Microstructure and Liquidity*, *1*(02), 1550004.

A List of Countries

Table 10: List of countries to which the traded stocks are denominated.

Arab Emirates	Denmark	Ireland	Korea, Republic of	Sweden
Australia	Egypt	Israel	Kuwait	Switzerland
Austria	Finland	Italy	Philippines	Taiwan
Belgium	France	Japan	Poland	Thailand
Brasil	Germany	Malaysia	Portugal	Turkey
Canada	Greece	Mexico	Qatar	United Kingdom
Chile	Hong Kong	New Zealand	Russia	United States
China	Hungary	the Netherlands	Singapore	
Colombia	India	Norway	South Africa	
Czechia	Indonesia	Pakistan	Spain	

B Details on Parameter Optimization

Equation 25 is a recursive formula, running from 0 to L, where $L-1$ is the number of hidden layers. This recursive formula can be summarized as $f(\mathbf{x}_i;\theta)$, where it takes input variables, $\mathbf{x}_i$, and transforms them with the weights and biases which are summarized in the parameter set θ. Optimization of the neural network means optimizing the weights and biases of the network such that a loss function is minimized. This is done by solving

$$\min_{\theta} \frac{1}{M} \sum_{i=1}^{M} \ell(f(\mathbf{x}_i;\theta), y_i) = \min_{\theta} \mathcal{L}(\mathbf{x}_i, y_i; \theta) \tag{46}$$

where M is the number of observations and $\ell(\cdot)$ is a function that measures the distance between the output of the neural network, $f(\cdot)$, and the corresponding target variable, y_i, for observation i and $\mathcal{L}(\cdot)$ is the resulting loss function.

For $\ell(\cdot)$ I use the binary cross-entropy loss function which is the standard for binary classification problems. The loss function is defined as

$$\ell(f(\mathbf{x}_i;\theta), y_i) = y_i log(f(\mathbf{x}_i;\theta)) + (1-y_i) log(1-f(\mathbf{x}_i;\theta)) \tag{47}$$

The output of the classification models $f(\cdot)$ can be regarded as the probability of the label being 0 and thus with this loss function the model is trained to minimize the probability of missclassification.

Since the neural network is highly dimensional it is not possible to analytically solve it for the parameters. For this reason I use the backpropagation algorithm in conjunction with the gradient descent method.

The loss function is dependent on the output layer and the target variable. By calculating the partial derivative of the loss function to the parameters of the output layer, layer L, we obtain the sensitivity of the output to a change in parameters of layer L. Since layer L depends on layer $L-1$ and so on we can calculate the partial derivative of the loss function to the parameters of each layer as

$$\frac{\partial \mathcal{L}}{\partial w^{(l-1)}} \frac{\partial w^{(l-1)}}{\partial w^{(l)}} \qquad \frac{\partial \mathcal{L}}{\partial b^{(l-1)}} \frac{\partial b^{(l-1)}}{\partial b^{(l)}} \tag{48}$$

where each partial derivative depends on the partial derivative of the next layer. This is a recursive formula and is where the term backpropagation comes from.

Since the value of the loss function depends on the target variable each partial derivative is different for each target variable. Therefore, we can take the partial derivative to be the average of the partial derivatives for all target variables,

$$\frac{\partial \mathcal{L}}{\partial w^{(l)}} = \frac{1}{M} \sum_{i=1}^{M} \frac{\partial \mathcal{L}_i}{\partial w^{(l)}}. \qquad \frac{\partial \mathcal{L}}{\partial b^{(l)}} = \frac{1}{M} \sum_{i=1}^{M} \frac{\partial \mathcal{L}_i}{\partial b^{(l)}}. \tag{49}$$

In this research I calculate the partial derivatives over a fraction, or batch, of the training sample instead of over the entire sample. This reduces the probability that the partial derivatives at each target variable within the batch are closely related and thus this method reduces redundant calculations. This method is called mini-batch gradient descent (Hinton, Srivastava, & Swersky, 2012) and saves computational power.

The partial derivatives make up the gradient vector as follows

$$\nabla \mathcal{L} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w^{(1)}}, \frac{\partial \mathcal{L}}{\partial b^{(1)}} \\ \frac{\partial \mathcal{L}}{\partial w^{(2)}}, \frac{\partial \mathcal{L}}{\partial b^{(2)}} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial w^{(L)}}, \frac{\partial \mathcal{L}}{\partial b^{(L)}} \end{bmatrix} \tag{50}$$

where each row denotes the gradient of the parameters in the corresponding layer. By subtracting these gradients from the corresponding parameters we can nudge the parameters in the direction that yields a prediction closer to the target variables, which minimizes the loss function. This is called gradient descent and yields a new set of parameters as follows

$$\theta^* = \theta - \eta \nabla \mathcal{L}, \tag{51}$$

where θ^* is the new parameter set and η is the step size, or learning rate. We can apply

this backpropagation process repeatedly with a new parameter set each iteration, called an epoch, until a learning criterion is reached and the parameter set is sufficiently optimized.

Gradient descent has trouble navigating parts of the problem surface where the surface curves much more strongly in one dimension than another (Sutton, 1986). For this reason I use a momentum term in Equation 51 which helps the gradient accelerate in the relevant direction (Qian, 1999). The gradient update step becomes

$$\begin{aligned} \upsilon_t &= \gamma \upsilon_{t-1} + \eta \nabla \mathcal{L} \\ \theta^* &= \theta - \upsilon_t \end{aligned} \tag{52}$$

where the previous update step is taken into account in the term υ_{t-1} multiplied by momentum parameter γ. By including the previous gradient update step we can force the new gradient update step to follow the same direction as the previous step, and smooth out the trajectory towards the optimum.

For further robustification of the model I also use L_1 regularization (Tibshirani, 1996). L_1 regularization is the addition of an absolute loss term to the loss function as follows

$$\mathcal{L} = \mathcal{L}_0 + \frac{\lambda}{2n} \sum_{w \in W} |w| \tag{53}$$

where w are the weights in the weight matrix of the neural network, W, $\mathcal{L}_0$ is the unregularized loss function and λ is the shrinkage parameter. By including this term the weights are also minimized to a certain degree when the loss function is minimized. Large weights can cause minimal changes in input to have a drastic impact on the output variable, thus by minimizing the weights we can robustify the model to outliers and out of sample data.

The problem with fitting a highly dimensional model is the increased chance of overfitting on the training data. When overfitting occurs, the model has very strong in-sample predictive power, however lacks out of sample power. This is because the model 'learns' all the patterns in-sample so well that it can not generalize them to out-of-sample data. For this reason the data is split into three parts, the training sample, the validation sample, and the test sample. The neural network is trained on the training sample and evaluated on the validation sample. If the performance on the validation sample is sufficient we say that it generalizes well. The test sample is used for subsequent analysis.

The optimization of the neural network incurs several parameters that guide the optimization, such as the learning rate, the momentum parameter, the mini-batch size and the shrinkage parameter. These are called hyperparameters and the choice of hyperparameters can influence the performance of the neural network. In this research I use random

search for the optimization of hyperparameters. Bergstra and Bengio (2012) show that only a few hyperparameters truly matter for the performance of optimization, however, that different hyperparameters matter for different datasets. This phenomenon makes systematic hyperparameter optimization, such as grid search, a poor choice. Bergstra and Bengio (2012) show that random search copes with this phenomenon and vastly increases computational speed.

For the initialization of the weights and biases of the neural network I use a robust method developed by He et al. (2015) specifically for rectified activation functions that are not differentiable at zero. They show that drawing weights from a zero-mean Gaussian distribution with standard deviation $\sqrt{\frac{2}{n_l}}$, where n_l is the number of nodes in layer l, the previous layer, leads to faster convergence than other common methods. They also initialize the biases at zero and the standard deviation of the weights of the first hidden layer at one.

C Feature Engineering

As shown in Section 2.3, there is a slight bias in the amount of positive and negative returns in the full sample. This is also the case in the training sample. This bias can cause the neural network to learn only the bias as a meaningful signal and ignore any other information and can therefore cause the neural network to also become biased. As a solution to this I propose to use undersampling. Given the large amount of return data this should not incur any meaningful change in the training of the neural network other than adjusting for the bias in the sample. When undersampling, a random portion of the biased sample is left out in training so that the training sample is not biased. This means that on average each mini-batch is unbiased.

Another important step to ensure the neural network is robustified for future information is standardization. When standardizing, the cross-sectional mean is substracted from input sample and the input sample is subsequently divided by the cross-sectional standard deviation. This ensures that the input sample always has mean zero and is thus also not biased, and that it falls within a certain range. This makes it easier for the neural network to recognize patterns. I opt for standardization since normalization might not work in the presence of large outliers. As shown in Section 2.3, the magnitude of the outliers in the return sample is very high. When normalizing, the input sample is compressed into the range 0 to 1. This means that in the presence of outliers the non-outlier sample is squeezed extremely closely together. This causes the loss of information and makes it

difficult for the neural network to pick up a meaningful signal.

D Neural Network Topology Selection

The topology of a neural network is defined as the amount of layers and nodes and how these are connected. Since I use a fully connected neural network the only two topology features to select are the amount of layers and the amount of nodes. Note that this is irrelevant for the logistic regression since the input is directed to one node, the output node.

Table 11: Table of accuracies for neural networks with different topologies. The rows indicate the number of layers and the columns indicate the number nodes per layer. For each topology the accuracy in the training and the validation sample is given.

	20	40	60	80	100	120	140	160	180	200	220	240
1	0,543	0,558	0,599	0,611	0,613	0,612	0,561	0,601	0,611	0,561	0,553	0,611
	0,534	0,534	0,535	0,534	0,536	0,536	0,533	0,533	0,534	0,533	0,533	0,536
2	0,577	0,559	0,589	0,569	0,591	0,571	0,603	0,584	0,587	0,576	0,582	0,575
	0,547	0,549	0,543	0,549	0,542	0,540	0,544	0,545	0,546	0,544	0,544	0,546
3	0,566	0,586	0,567	0,562	0,568	0,568	0,554	0,551	0,561	0,567	0,561	0,559
	0,550	0,545	0,549	0,549	0,550	0,550	0,550	0,551	0,548	0,550	0,549	0,550
4	0,565	0,549	0,552	0,550	0,552	0,559	0,552	0,553	0,551	0,552	0,549	0,546
	0,546	0,552	0,553	0,548	0,551	0,550	0,555	0,554	0,552	0,552	0,552	0,555
5	0,558	0,549	0,558	0,552	0,534	0,553	0,532	0,552	0,563	0,552	0,558	0,552
	0,551	0,554	0,548	0,552	0,550	0,550	0,551	0,555	0,552	0,554	0,551	0,551
6	0,536	0,550	0,550	0,556	0,556	0,554	0,564	0,552	0,554	0,551	0,553	0,552
	0,551	0,553	0,553	0,551	0,550	0,551	0,552	0,546	0,554	0,554	0,555	0,554
7	0,540	0,557	0,554	0,557	0,554	0,540	0,554	0,553	0,554	0,553	0,554	0,552
	0,555	0,551	0,546	0,551	0,552	0,555	0,550	0,548	0,549	0,548	0,551	0,551
8	0,551	0,566	0,555	0,542	0,552	0,554	0,556	0,556	0,554	0,555	0,553	0,555
	0,556	0,545	0,549	0,552	0,549	0,550	0,552	0,544	0,547	0,546	0,548	0,550
9	0,557	0,553	0,556	0,555	0,557	0,554	0,567	0,542	0,552	0,555	0,555	0,554
	0,548	0,5506	0,552	0,550	0,549	0,545	0,549	0,548	0,553	0,547	0,553	0,550
10	0,556	0,556	0,554	0,556	0,555	0,557	0,544	0,558	0,554	0,553	0,554	0,554
	0,551	0,548	0,553	0,546	0,549	0,552	0,546	0,544	0,551	0,552	0,545	0,549

In Table 11 we see the accuracies for the neural networks with different topologies. The rows indicate the amount of hidden layers and the columns the amount of nodes per layer. The gray cells denote the topologies for which the neural network has the highes validation accuracy. This is important since this implies a better capability of the neural network to generalize to out-of-sample data. Overall the accuracies lie close together, however, we see that with few hidden layers there is some overfitting on the training data and thus lower validation accuracy. For this research I have chosen 6 hidden layers with

220 nodes per layer. I have opted for this topology since the accuracies of the training sample and validation sample are high and lie close together. Furthermore, I have opted for a topology with a high amount of nodes per layer, this is to preserve the amount of information that can flow through the neural network. This might increase flexibility when adding the LSTM layer.

To conserve computational time I first optimize the topology of the neural network and then add the LSTM layer to form the LSTM model. The LSTM is then fitted again since the LSTM cells pick up different information. Table 12 shows the training and validation accuracies of the LSTM model with different amount of hidden nodes per LSTM cell.

Table 12: Accuracies in the training and validation sample for the LSTM models with different amount of memory nodes per LSTM cell. The columns denote the amount of nodes within the memory of the LSTM model. The rows denote the training and validation sample.

Nodes	5	10	15	20	25	30	35	40
Train	0,56	0,566	0,567	0,609	0,611	0,608	0,614	0,614
Validation	0,555	0,564	0,562	0,543	0,544	0,543	0,535	0,541

We see that the models with 5 to 15 memory nodes per LSTM cell have the best accuracy in the validation sample. The models with more nodes all seem to overfit to the training data with a higher accuracy on the training data and a lower accuracy on the validation data. Since the accuracy for the model with 10 memory nodes per LSTM cell has the highest validation accuracy with the closes training and validation accuracy I continue the analysis with this model.

E Training and Validation Performance

F Quantile Regression

Table 15: Quantile regression results of duration on a constant, FDV, spread and market cap. The left-most column shows the quantiles of the regression. The other columns show the regressors, the respective coefficients and the standard errors, shown in brackets. All coefficients for which the two-sided t-test, with $h_0 : \theta = 0$, has p-value < 0.001 is shown in bold.

Quantile	Intercept	FDV	Spread	Market Cap
90%	**0,7211**	**0,0355**	**0,9479**	**0,0672**
	(0,001)	(0,003)	(0,021)	(0,008)
10%	**0,0034**	**0,2529**	**0,2106**	**-0,0092**
	(0,001)	(0,002)	(0,018)	(0,005)

Table 13: Classification performance of each return predictor in the training sample. Two panels are shown for each predictor. The top panel shows the confusion matrix where the columns denote the predicted direction and the rows denote the trude direction of the returns. The bottom panel shows several performance metrics.

	Optimist		Lazy		Linear	
	Down	Up	Down	Up	Down	Up
Down	0	0,518	0,261	0,256	0,275	0,243
Up	0	0,482	0,216	0,267	0,250	0,232
Accuracy	0,482		0,528		0,507	
Specificity	0		0,504		0,531	
Sensitivity	1		0,553		0,481	
F1-score	0,651		0,531		0,485	

	Logistic		NN		LSTM	
	Down	Up	Down	Up	Down	Up
Down	0,075	0,475	0,323	0,227	0,230	0,320
Up	0,016	0,434	0,225	0,225	0,114	0,336
Accuracy	0,508		0,548		0,566	
Specificity	0,136		0,588		0,417	
Sensitivity	0,964		0,499		0,747	
F1-score	0,638		0,498		0,607	

Table 14: Classification performance of each return predictor in the validation sample. Two panels are shown for each predictor. The top panel shows the confusion matrix where the columns denote the predicted direction and the rows denote the trude direction of the returns. The bottom panel shows several performance metrics.

	Optimist		Lazy		Linear	
	Down	Up	Down	Up	Down	Up
Down	0	0,467	0,241	0,226	0,216	0,251
Up	0	0,533	0,236	0,297	0,232	0,301
Accuracy	0,533		0,538		0,517	
Specificity	0		0,516		0,462	
Sensitivity	1		0,558		0,564	
F1-score	0,695		0,563		0,555	

	Logistic		NN		LSTM	
	Down	Up	Down	Up	Down	Up
Down	0,086	0,463	0,325	0,224	0,236	0,314
Up	0,019	0,431	0,233	0,218	0,122	0,329
Accuracy	0,518		0,543		0,564	
Specificity	0,157		0,592		0,429	
Sensitivity	0,958		0,484		0,729	
F1-score	0,641		0,488		0,601	

Table 16: This table shows the distribution of several duration measures. The first column shows the distribution of the true duration of the trades, the second column shows the distribution of the fitted bottom quantile duration for each order, the third column shows the distribution of the fitted top quantile duration of each order, the fourth column shows the distribution of the difference in duration between the true duration and the fitted bottom quantile duration and the last column shows the distribution of the difference in duration between the true duration and the fitte top quantile duration.

	Truth	10%	90%	Diff. 10%	Diff. 90%
Mean	0,5287	0,0309	0,7454	-0,4977	0,2168
Std.dev.	0,2524	0,0432	0,0234	0,2528	0,2513
Skewness	-0,7877	2,6612	3,715	0,7568	0,7918
Kurtosis	-0,8373	7,6844	27,0953	-0,8377	-0,8231
Quantiles					
95%	0,7572	0,1275	0,7874	-0,0095	0,7123
75%	0,7225	0,0343	0,7518	-0,2851	0,4319
50%	0,6666	0,0133	0,7372	-0,6316	0,0732
25%	0,3078	0,0067	0,7311	-0,7108	0,0141
5%	0,0296	0,0035	0,7275	-0,7314	-0,0008