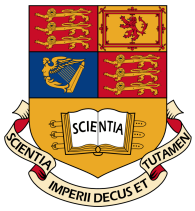
Predicting US stock returns using closing auction imbalance data

Cléa Morand (CID: 01805978)

Department of Mathematics
Imperial College London
London SW7 2AZ
United Kingdom

September 2020



Imperial College London

Thesis submitted as part of the requirements for the award of the MSc in Mathematics and Finance, Imperial College London, 2019-2020

Declaration

This work contained in this thesis is my own work unless otherwise is stated.

Signature and date:

In loving memory of my dad, Raphaël

Acknowledgements

I would like to express my sincere gratitude to Professor Johannes Muhle-Karbe, my research supervisor, for his multiple advice and guidance during this project.

I also would like to thank Deutsche Bank and particularly the Quantitative Researcher Electronic Equities team. Thank you Andrew, Benyamin, François and Yuan for your precious help during this project. The completion of this master thesis would have been impossible without your insightful comments, your experience and your kindness. I also express my sincere gratitude to Guhao Wu, for his guidance through this project, and his understanding.

Thank you to my family, especially to my mum, Arthur, Oscar and Guillaume for your support during my studies and this master thesis.

Abstract

This thesis introduces an approach to forecast US stocks price forward-return during the 10 minutes before the closing auction, using as explanatory variables the data published by venues during these last ten minutes, displaying the volume imbalance between sell and buy orders. To predict these forward returns, we used linear regression with imbalance data as predictors, on NASDAQ and NYSE venues, and for stocks in two indexes: S&P 500 and RUSSELL 2000. Our study showed some encouraging results. The predictability of forward returns for S&P stocks on NASDAQ venue is good, and increasing when we approach MOC and LOC cut-off time. Results show that we can earn 30 % of the spread with the model. Some interesting results are also displayed with on NYSE venue and for RUSSELL Stocks. But there are also situations where the R^2 is not significantly different from zero. This study could help improving stock price-return prediction during the 10 minutes before the US closing auction, between 3:50pm and 4pm.

Contents

List of Figures

Introduction

The evolution of financial markets and trading has always been closely related to technology advancements. Since the 1970s, the markets have been migrating to electronic trading platforms. Nasdaq started in 1971, but didn't really begin as an electronic trading system, there was only an automated quotation system and trades were then handled over the phone. Eventually, Nasdaq added other features like automated trading systems. Today, given the benefits of the electronic systems and the clients' preference for them, a very large percentage of the world's exchanges have converted to this method.

Exchanges offer different mechanisms for matching buyers and sellers, such as the continuous limit order book, which is the prevailing execution method used during the trading day. Apart from the continuous trading model, many exchanges utilize opening and closing auctions to set opening and closing prices. Closing auctions are held at the end of the trading day to determine an asset's closing price, before the market closes and reopens the following morning. There are many different order types available at the close, including market-on close (MOC), limit-on-close (LOC), and imbalance offset orders.We are going to explain more in depth the role of each of these orders in this thesis. MOC and LOC orders play the most important role in determining closing prices.

Since the early 2000s, there has been significant growth in MOC activity around the globe, a trend that has been particularly pronounced since 2016. Over the month of January 2016, global MOC activity represented on average less than 7% of total volume throughout the trading day . In contrast, as of January 2020, the closing auction represented 13.4% of the full trading day's activity. [5] This increasing volume makes the 10 minutes before the closing auction, between 3:50pm and 4pm, particularly important for US closing auctions.

In this thesis, we will try to predict forward price returns, during the ten minutes before the US closing auction (so between 3:50pm and 4pm), using the data published by venues during these last ten minutes, displaying the volume imbalance between sell and buy orders. We will use machine learning techniques to do so, mostly linear regression. The thesis will be structured as follows. We will first introduce the context of closing aucion more in depth,

how the closing auctions work for NASDAQ and NYSE venues, the two venues we are going to focus on in this thesis. We will also introduce the different types of orders in closing auctions. Then we will move on to the theory and assumptions of linear regression and explain the target of this linear regression, and the features we are going to use. Then, in section 3, we will explore the data for S&P 500 stocks, and try to apply the model to predict price forward returns. We will then discuss the results, first for NASDAQ venue, then for NYSE venue, because these two venues have a different behaviour. Then in last section, we are going to discuss the results for RUSSELL 2000 stocks. The Russell 2000 Index is a small-cap stock market index of the smallest 2000 stocks in the Russell 3000 Index. It has been embraced as the small cap index of choice for measuring the small cap market segment. This will enable us to compare the results for S&P stocks and RUSSELL stocks and see if there is different behaviour between large-cap stocks and small-cap stocks.

1 Context

1.1 Closing auctions

The closing auction is the last event of the trading day across all major exchanges and is designed to determine the closing price for each stock. The closing price is crucial, as it is the most widely published reference price for all equity-linked products.

In recent years, daily US equity trading volume has shifted more towards the closing auction. For S&P 500 stocks, the closing auction volume increased substantially, from 4 % of the daily volume eight years ago to more than 10 % of the daily volume now. The last half-an-hour accounts for more than 25 % of the total daily volume today. [4]

The two largest primary exchanges for closing auctions are NYSE and NSDQ. I only worked on these two venues during my project. Each exchange has its own closing auction procedures and methods for disseminating imbalance messages, and its own methodology to determine the eligibility and priority of orders participating in the closing auction. The information fields in imbalance messages include reference price, paired quantity, imbalance quantity, near and far indicative clearing price.

During the trading day, a stock can be traded on any exchange. But at the close, it reverts to the exchange where it is listed. This means that stocks listed on the NYSE will have dramatic closing auction volume on the NYSE at closing, and the process is similar for NASDAQ-listed stocks

The 10 minutes before the closing auction, between 3:50pm and 4pm, are particularly important for US closing auctions because of the growing volume of trading during this period (this is not the case in EMEA). Let me remind you the behavior of NASDAQ and NYSE venues during these 10 minutes before closing auction, and the different types of market orders during this period.

1.1.1 NASDAQ

MOC : (Market On Close) A market-on-close order is simply a market order that is scheduled to trade at the close, at the most recent trading price.

LOC : (Limit On Close). It is also executed on the close, but only if the close is equal or lower (for a buy order) or equal or higher (for a sell order) than the limit price.
IO : (Imbalance Only) Imbalance only (IO) orders are limit orders that provide liquidity during the opening cross and closing cross on the Nasdaq stock exchange. The orders are given to offset an imbalance in the opening or closing cross.

The NSDQ cutoff time for MOC/LOC order entry is 15:55. After 15:55, Imbalance-Only (IO) orders can be submitted before the close and late LOC orders can be submitted until 15:58, but IO and late LOC Orders cannot be updated or cancelled. Imbalance data is published every 10 seconds before 15:55 and every second after 15:55. [4]

1.1.2 NYSE

One key feature for the NYSE close is d-Quotes, D Order , which is short for Discretionary Order. The NYSE D-quote provides similar access to the NYSE closing auction, but without the restrictions applied to traditional MOC/LOC orders. Specifically, traders using MOC/LOC orders must send them in no later than 3:50PM, the NYSE cutoff time. After 3:50PM, MOC/LOC orders can be sent only if there is a sizeable imbalance, in which case they are allowed to send offsetting orders only. For example, if a large buy imbalance exists after 3:50PM, traders may submit on-close sell orders, but not buy orders. Also, after the cutoff, MOC/LOC orders cannot be canceled, except for certain error scenarios.

Contrast this with the D-quote, which has none of these restrictions. D-quotes can be sent up until 3:59:50 PM, ten seconds before the close, regardless of the direction of the trade or whether an imbalance exists. So, D-quotes can be used even if they add to a large existing imbalance, create a large imbalance that otherwise wouldn't exist, or even flip the direction of the imbalance (from buy to sell, for example). And what's more, D-quotes can be canceled up until 3:59:50 PM. The wrinkle that prevents D-quotes from being a perfect substitute for an unrestricted MOC/LOC order is that a floor broker must submit the D-quote into the auction. Specifically, buyside orders must be routed to an NYSE floor broker, who then manually releases them to the closing auction. [12]

1.2 Data

We want to predict price forward price returns at different horizons, using imbalance data, during the 10 minutes before the closing auction in the US. between What is imbalance data? Order imbalance is a situation resulting from an excess of buy or sell orders for a specific security on a trading exchange. Here, imbalance is the excess of buy or sell in the imbalance feed for market orders on closing auction.

The data I used for this thesis project comes from an Order Imbalances feed, provided to me by Deutsche Bank, that displays real-time publication of buy and sell imbalances sent at specified intervals during auctions throughout the trading day for all listed securities. It also provides the indicative auction price, which is the price at which the auction would occur if no new orders were received and the auction were held now, the price at which the maximum volume of orders can be executed at the time of an auction.

2 Linear Regression

2.1 Target

The goal of this project is to predict price forward return, at different horizons. We define n-seconds price forward return ($r_{f,t,n}$) as :

$$r_{f,t,n} = \frac{log(\frac{p_{t+n}}{p_t})}{\tilde{s}_t}$$

where p_t is midquote at time t
and $\tilde{s}_t$ is the stock's medium spread over the 20 trading days before t, in basis point (bps). (which means the spread is computed as $2 * \frac{ask-bid}{ask+bid}$)

Here, the forward returns are normalised by the medium spread over the last 20 trading days. It means that instead of computing expected returns as percentage of the price, I will compute them as a percentage of the spread (because both numerator and denominator of the return fraction are in bps).

2.2 Method used

The method that seemed the most adapted to predict forward returns was linear regression.
Let's recall quickly what OLS regression is.
Simple Linear Regression is a statistical model, widely used in ML regression tasks, based on the idea the variable y can be explained by the following formula: Suppose the data consists of n observations (x_i,y_i). Each observation i includes a scalar response y_i (target) and a column vector x_i of values of p parameters (features) x_{ij} for $j = 1, ..., p$. In a linear regression model, the response variable, y_i, is a linear function of the features:

$$y_i = \alpha_i + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon_i$$

Where ϵ_i is the error term. We make the following assumptions about these error terms :

- E(ϵ_i) = 0 for all i = 1, . . . , n
- Var(ϵ_i) = σ^2 for all i = 1, . . . , n

- $\text{Cov}(\epsilon_i, \epsilon_j) = 0$ for all i $\neq j$

Moreover, α, β_j are the true (but unobserved) parameters of the regression. The parameter β_j represents the variation of the dependent variable j when the independent variable has a unitary variation.

Now, the idea of Ordinary Least Square Linear Regression is finding those parameters α and β_j for which the error term is minimized. To be more precise, the model will minimize the squared errors. We compute the coefficients by minimizing the sum of squared residuals (RSS):

$$RSS = \sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - ... - \hat{\beta}_p x_{ip})^2$$

A linear regression model is built with several underlying assumptions about the model and the variables being modelled. The first assumption is that the relationship between the predictors and the target is linear. The second one is the assumption that the relationship is additive.
An assumption common to many techniques, including models built by multiple linear regression, is that there should exist no multicollinearity in the model, which occurs when predictors used in a model are strongly correlated with each other. Models containing multicollinearity in predictors could lead to erratic changes in coefficient estimates through only small alterations in the model or the data. To check for this, we will check correlation between features and we will use decorrelation methods detailed later.
In python, I used the sklearn package.

2.3 Features

I first defined different features in order to predict price forward returns. Let's recall the definition of imbalance: Order imbalance is a situation resulting from an excess of buy or sell orders. So I needed to create features to best reflect this imbalance. I tried several of them :

$$imb_{t,1} = \frac{sellv_t - buyv_t}{\overline{vol_{close}}}$$

$$imb_{t,2} = \frac{log(\frac{sellv_t}{buyv_t})}{\overline{vol_{close}}}$$

$$imb_{t,3} = \frac{sellv_t - buyv_t}{sellv_t + buyv_t}$$

$$imb_{t,4} = sign(sellv_t - buyv_t) * \frac{log(|sellv_t - buyv_t|)}{\overline{vol_{close}}}$$

where $\overline{vol_{close}}$ is the average closing volume over the 20 past days . Again, we need to normalise the imbalance by the average volume on close in order to have the same order of magnitude for the different stocks.

$sellv_t$ is the total sell volume in imbalance feed at time t
$buyv_t$ is the total buy volume in imbalance feed at time t

I also needed features to reflect the change of the imbalance over a certain period. Let's call imbalance delta ($r_{imb,t,n}$) the difference of imbalance between t-n and t.

$$r_{imb,t,n} = imb_{t,1} - imb_{t-n,1}$$

The imbalance data feed also provides the indicative auction price $p_{auction}$, which is the price at which the auction would occur if no new orders were received and the auction was held now. So I created a feature that is the difference between this indicative auction price and the actual midquote.

$$diff_{price,t} = \frac{log(\frac{p_t}{p_{auction,t}})}{\tilde{s}}$$

where p_t is the midquote at time t
and $p_{auction}$ is the indicative auction price at time t
and $\tilde{s}$ is the the median spread over the last 20 days

Finally, I also defined backward returns as :

$$r_{b,t,n} = \frac{log(\frac{p_t}{p_{t-n}})}{\tilde{s}}$$

to take into account changes in midquote during the 10 minutes before the closing auction.

3 S&P 500 Stocks

3.1 NASDAQ

3.1.1 Data

Let's first focus on the time between 3:50 and 3:55pm. The dataset includes all trading days between 01/04/2020 and 01/08/2020, and there is a datapoint every 10 seconds between 3:50 and 3:55pm, which is the frequency at which imbalance data is published.
At the close, a stock reverts to the exchange where it is listed. Here, we are focusing on S&P 500 stocks listed on NASDAQ venue. For the time period studied here, there were 140 such stocks.
The training set is 2 months long, and then there is a validation test, used to find the best features with features selection. More variables will always improve in-sample R2, too many parameters will worsen out of sample performance due to overfitting. That is why we need this validation set, followed by a 1-month-long test set.

Figure 1: Dataset separation

3.1.2 Features Selection

In this section, I will explain in depth the features selection process I went through during all the project. I used several methods for features selection.

First, I used a Gridsearch method. The algorithm I used for this was the following :

Algorithm 1: Features Selection

Take the set of all the features
for *every subset of feature* **do**
    Train the model with the features of the subset (in the training set);
    Compute the out of sample R^2 (in the validation set);
end
Choose the subset of features with the highest R squared
Result: Subset of features that gives the highest R^2

This method is possible thanks to the relatively low number of features.

After having applied this method, I used two other methods to confirm the choice of features.

- The correlation heatmap, to visualise Pearson's correlation between all features/ targets.
- Lasso Regression

Let's recall quickly what Lasso Regression is.
It consists in adding a penalty to the different parameters of the machine learning model to reduce the freedom of the model and in other words to avoid overfitting.
In linear model regularisation, the penalty is applied over the coefficients that multiply each of the predictors. From the different types of regularisation, Lasso or L1 has the property that shrinks some of the coefficients to zero. Therefore, that feature can be removed from the model.
We use Lasso regression (L1) and not Ridge (L2) because Ridge Regression doesn't set the coefficients to zero and so is less efficient for a feature selection purpose.
To find the coefficients, instead of minimizing

$$\sum_{i}^{n}(y_i - \sum_{j} x_{ij}\beta_j)^2$$

We now have to minimize :

$$\sum_{i}^{n}(y_i - \sum_{j} x_{ij}\beta_j)^2 + \alpha \sum_{j} |\beta_j|$$

It will set more and more coefficients to zero as α grows and can be used as a features selection.

Let's now see the results of this feature selection among all the features. Let's recall the different features before selection:

- Different formulations of imbalance ($imb_{t,1}, imb_{t,2}, imb_{t,3}, imb_{t,4}$)
- Imbalance return at different horizons ($r_{imb,t,10}, r_{imb,t,20}, r_{imb,t,30}, r_{imb,t,60}$)
- Backward return at different horizons ($r_{b,t,4}, r_{b,t,8}, r_{b,t,16}, r_{b,t,32}$)

Here, we don't have the $diff_{price,t}$ feature because on NASDAQ venue, indicative auction price is only published after 3:55 pm .

I ran the Gridsearch feature selection for every forward return horizon. The results were the following :
For every forward return horizon, two features are always selected : **the imbalance return** with a horizon of 10 seconds ($r_{imb,t,10}$) and **the imbalance ratio** ($imb_{t,1}$). Then for each forward return horizon, different backward returns are also selected, but they improve very slightly the R^2.

Horizon	Features selected
2 seconds	'b return 4second', 'ratio imb', 'imb return 10second'
4 seconds	'b return 4second', 'b return 16second', 'ratio imb', 'imb return 10second'
8 seconds	'b return 16second', 'ratio imb', 'imb return 10second'
16 seconds	'b return 16second', 'ratio imb', 'imb return 10second'
32 seconds	'b return 8second', 'b return 32second', 'ratio imb', 'imb return 10second'
60 seconds	'b return 32second', 'ratio imb', 'imb return 10second'

Let's now visualise the correlations table Figure 2

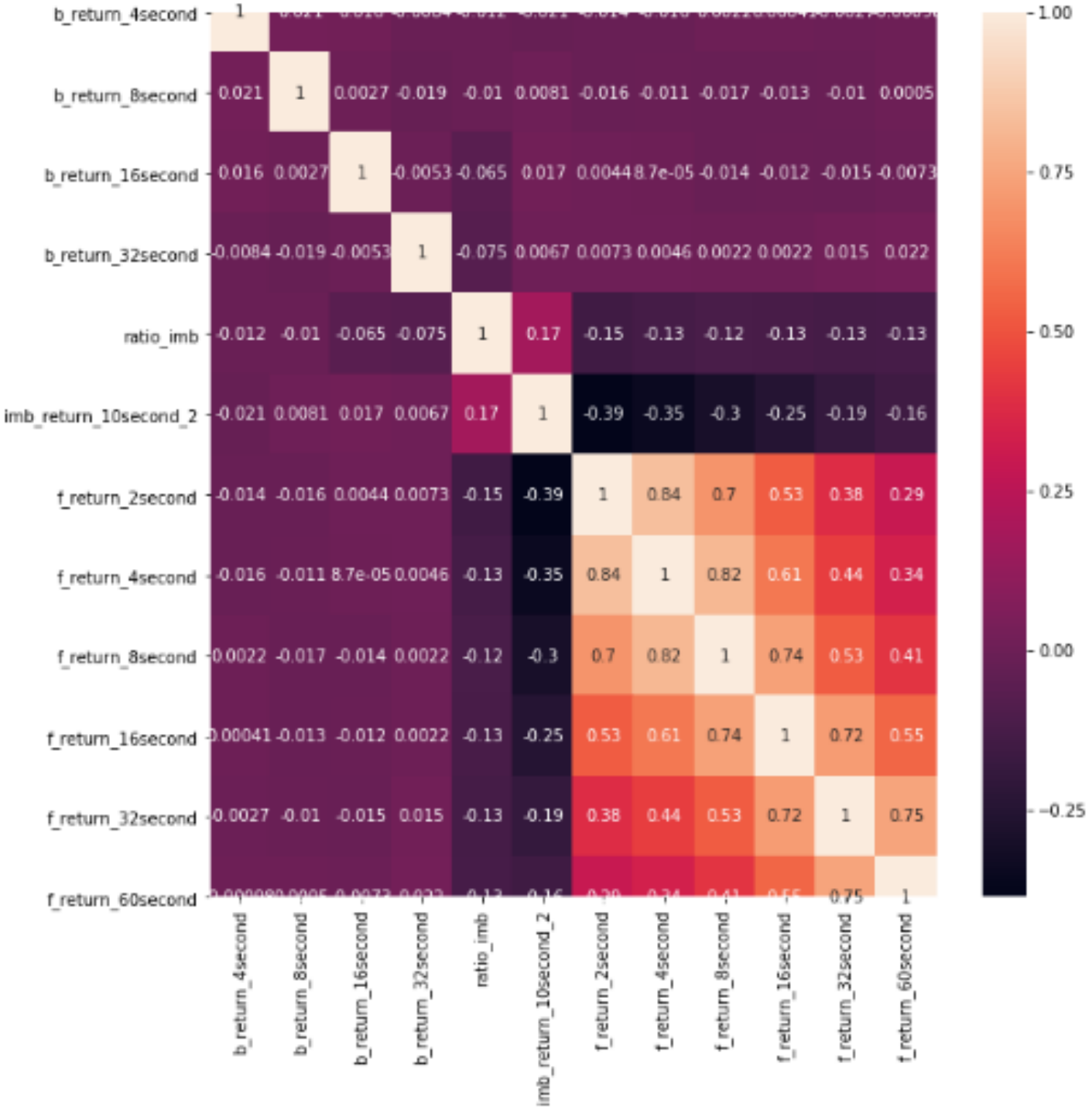


Figure 2: Correlation table

We can see that the correlation with forward returns is significantly different from zero only for the two features mentioned before: **the imbalance return** with a horizon of 10 seconds ($r_{imb,t,10}$) and **the imbalance ratio** ($imb_{t,1}$), and this for every horizon of forward return. (Correlations between forward returns are high because forward returns are not decorrelated).

Finally, Lasso Regression confirms the choice of these two features. The following plot displays the evolution of the coefficients of different features when alpha grows, alpha is the α in :

$$\sum_{i}^{n}(y_i - \sum_{j} x_{ij}\beta_j)^2 + \alpha \sum_{j} |\beta_j|$$

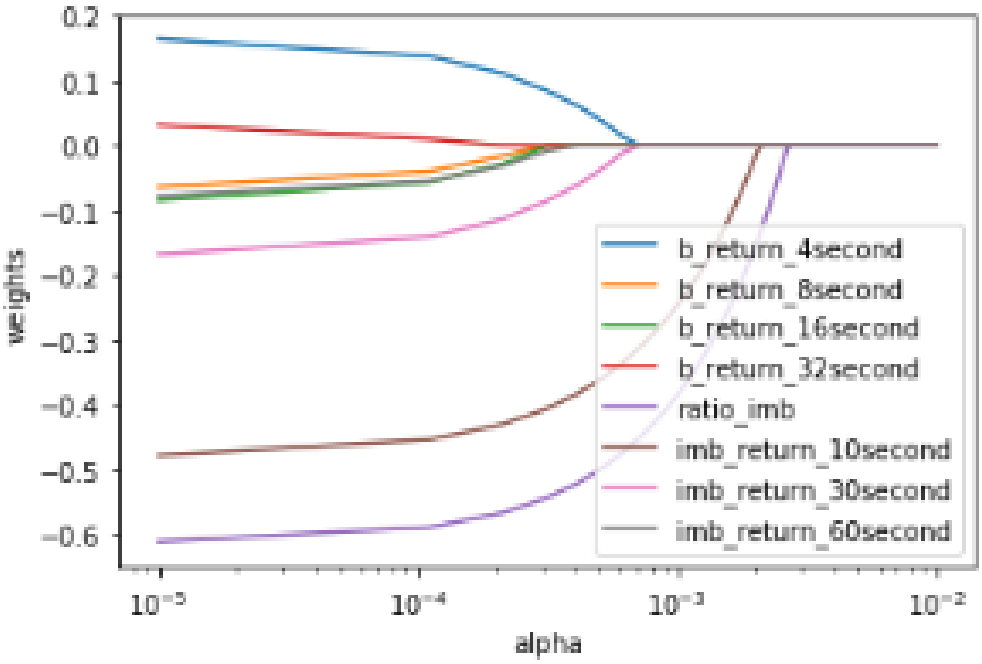


Figure 3: Lasso Regression

The coefficients that are set to zero in the last place are, again, **the imbalance return** with a horizon of 10 seconds ($r_{imb,t,10}$) and **the imbalance ratio** ($imb_{t,1}$).

3.1.3 Decorrelated Features

There should exist no multicollinearity in the model, which occurs when predictors used in a model are strongly correlated with each other. Models containing multicollinearity in predictors could lead to erratic changes in coefficient estimates, or could lead to an over estimation of the R^2.
We can see on the Correlation heatmap [Figure 2] that the correlation between our two main features is 0.17. The backward returns are decorrelated. So we will need to decorrelate **the imbalance return** with a horizon of 10 seconds ($r_{imb,t,10}$) and **the imbalance ratio** ($imb_{t,1}$), to be sure that there is no collinearity between these two features.

To do so, I used the Gram-Schmidt Process. [1] [2]

The Gram-Schmidt process is typically presented as a process for orthonormalizing the columns of a matrix. Let $\mathbf{A}$ be an m x n matrix with each row being a sample and each column a feature. Further, define $\mathbf{B}$ as the transformed matrix and $\mathbf{X}_i$ as the i-th column of a matrix $\mathbf{X}$. Using this notation, the orthonormalized matrix $\mathbf{B}$ is computed as :

$$\mathbf{B}_i^* = \mathbf{A}_i - \sum_{j=1}^{i-1} P(\mathbf{B}_j^*, \mathbf{A}_i)$$

$$\mathbf{B}_i = \frac{\mathbf{B}_i^*}{||\mathbf{B}_i^*||}$$

for i = 1,2,. . . ,n,

where

$$P(\mathbf{u}, \mathbf{v}) = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle} \mathbf{u}$$

is the projection of u onto v, and $\langle \mathbf{u}, \mathbf{v} \rangle$ denotes the inner product of these vectors

To apply this method on a dataset, the data matrix $\mathbf{A}$ is standardized. This fulfils the condition required above and causes the covariance of columns in the standardized matrix to be equal to the correlation. Recall that correlation is defined as

$$R(\mathbf{x}, \mathbf{y}) = \frac{C(\mathbf{x}, \mathbf{y})}{\sigma_\mathbf{x} \sigma_\mathbf{y}}$$

Where $\sigma_\mathbf{x}$ is the standard and C(x,y) is the covariance. deviation of x. Thus, the correlation and covariance are equal, since the standardized columns have variance one.

Next the Gram-Schmidt process is applied sequentially on the columns. The first column is left unchanged since it already has variance 1. The second transformed column is computed as the second original column with any portion that is correlated to first column removed. In general, the i-th transformed column is equal to the i-th original column with any portion that is linearly correlated to the j-th transformed column removed for all $j < i$.

	ratio_imb	imb_return_10second_2
ratio_imb	1.000000e+00	-7.038665e-14
imb_return_10second_2	-7.038665e-14	1.000000e+00

Figure 4: Correlation after the Gram-Schmidt Process

After the Gram-Schmidt Process, the correlation between the ratio imbalance and the imbalance return is zero. The features are decorrelated and we can apply a linear regression without the collinearity problem.

There is no need to decorrelate the other features. The backward returns are already decorrelated (by defining the backward return at 10 seconds to be the price difference between 0 and 10 seconds and the price return at 20 seconds to be the price difference between 10 and 20 seconds). This gives decorrelated backward returns because price returns have a small autocorrelation.
The correlation between backward returns and other features is already very low.

3.1.4 R squared computation

Let's recall the definition of the R^2 :
The R^2 provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model. In the context of regression it is a statistical measure of how well the regression line approximates the actual data. It is computed with the following formula:

$$R^2 = 1 - \frac{SSR}{SST}$$

where SSR is the sum squared regression
and SST is the total sum of squares

$$R^2 = 1 - \frac{\sum(y_i - \hat{y_i})^2}{\sum(y_i - \bar{y})^2}$$

where $\hat{y_i}$ is the estimated y_i
and $\bar{y}$ is the mean of y over the whole dataset

The highest possible R^2 is 1, which means that the data is perfectly predicted. If the R^2 gets negative, it means that the baseline would be a better predictor, so here the mean of y over the whole dataset.

That is the definition used by python when we use the r^2 function. The baseline used is the mean of y over the whole dataset. But actually, using the mean over the whole dataset is forward looking, and maybe this is a baseline that is quite hard to beat.

So I defined other baselines :

- A baseline of 0. Actually, this should be quite similar to the python R^2, because the mean of forward returns over the whole dataset should be zero.
- A baseline that is the forward return prediction, but only made with backward returns as features. This means that we train the model with the features being the backward returns, and then the baseline is the prediction made with this model.

Let's now plot the out of sample R^2, for different baselines.

Here, I trained the model and then computed the out of sample R^2 (in the test set) for different baselines. We can see that the R^2 decreases with the forward return horizon. The model predicts better the price returns with small horizons. For 2 seconds forward returns, the R^2 is quite large : 0.07 for python computation. As the horizon grows, the R^2 gets close to zero and becomes negative over 60 seconds, so we will not work with horizons beyond 60 seconds.

Something else we can notice is the small difference between python baseline and 0 baseline. They should be similar, as mean of forward returns over the whole dataset should be equal to zero. But the mean of forward return is

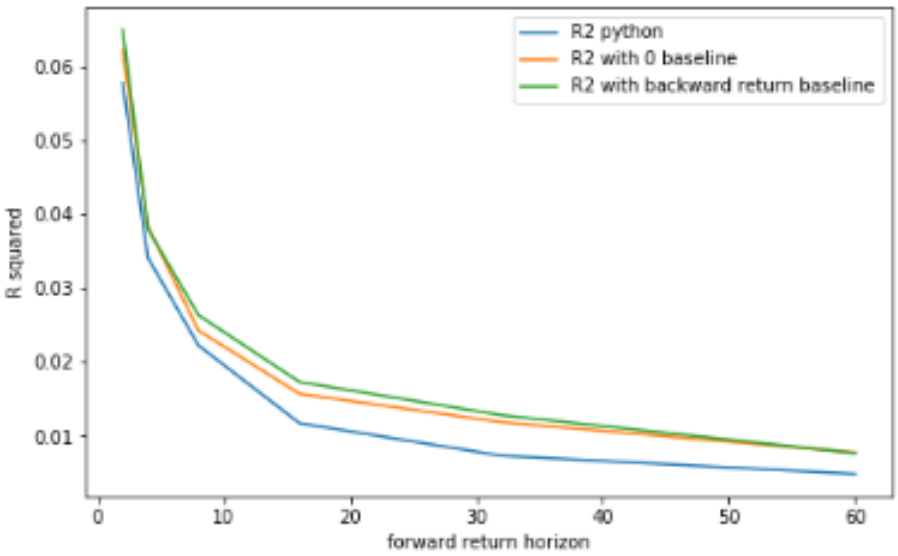


Figure 5: R squared for different baselines and different forward returns horizons

actually a bit larger than zero, which explains the difference. The python R^2, with some forward looking, is harder to beat and that is why python R^2 is always the smallest one.

3.1.5 Coefficients

Let's now have a look to the coefficients of the linear regression. (next page)

These are the plots of the coefficients of two features : the imbalance and the imbalance return. For recalling, the definition of these two features are :

$$imb_{t,1} = \frac{sellv_t - buyv_t}{\overline{vol_{close}}}$$

$$r_{imb,t,n} = imb_{t,1} - imb_{t-n,1}$$

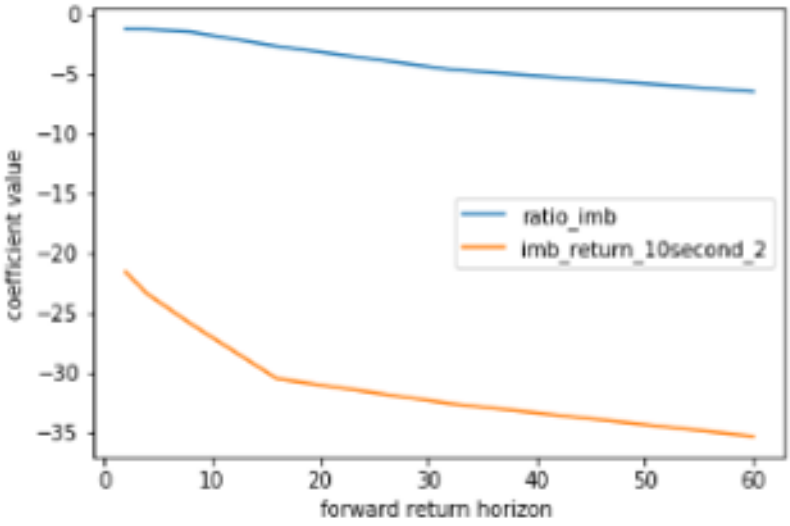


Figure 6: Evolution of the coefficients of the linear regression with the forward return horizon.

The two coefficients are negative. Let's check if this sign makes sense : If the imbalance is positive, it means that the sell volume is larger than the buy volume. This imbalance will lead to a decrease in the midquote, because there is more sell than buy. If the price decreases, then the forward return is negative. The process is the same for the other feature.

Let's plot the evolution of these coefficients with standard deviation bars, to see if these coefficients are stable:

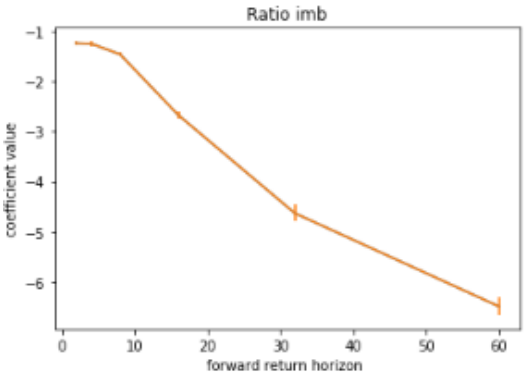


Figure 7: Ratio imbalance coefficient

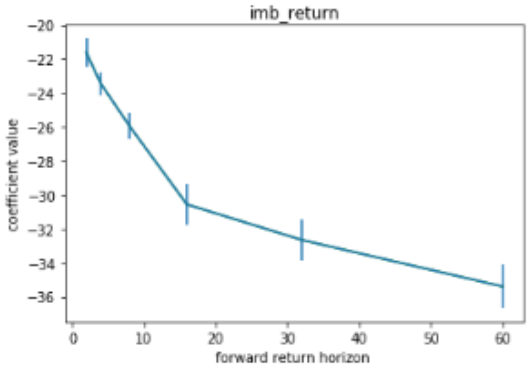


Figure 8: Imbalance return coefficient

To do this, I used a bootstrap strategy. I randomly took 70% of the data and computed the coefficients with this new dataset. I did that a hundred times, and then computed the standard deviation of each coefficient. We can see that both coefficients, and particularly the first one, are particularly stable, because the standard deviation is quite small.

Plot of the evolution of coefficient with the size of the dataset?

3.1.6 Model by minute

Currently, we are training the model for S&P 500 stocks on NASDAQ, between 3:50 and 3:55 pm. (We will see after 3:55 pm later).

I tried to train a model per minute, to see if there was a different behaviour for each minute. I then ploted the out of sample R^2 for each model, using the python R^2.
On the plot (Figure 9), we can see that there is an obvious difference between every minute.

- Minutes 51 and 52 have a quite low R^2, around 0 for every forward return horizon.
- Then minute 53 has a higher R^2
- And it continues to grow for minute 54, that outperforms the R^2 for the whole dataset.

Why does the R^2 grow when we get near 3:55 pm? The NASDAQ cutoff time for MOC/LOC order entry is 15:55. So it seems like the predictability of returns is improved when we get near 3:55pm.

Let's now train the model on the whole training set, and compute the R^2 minute. I trained the model for the whole dataset, between 3:50pm and 3:55pm, and then separated the test set into several test sets: one for each minute, to see for which minute the model predicts better forward returns.

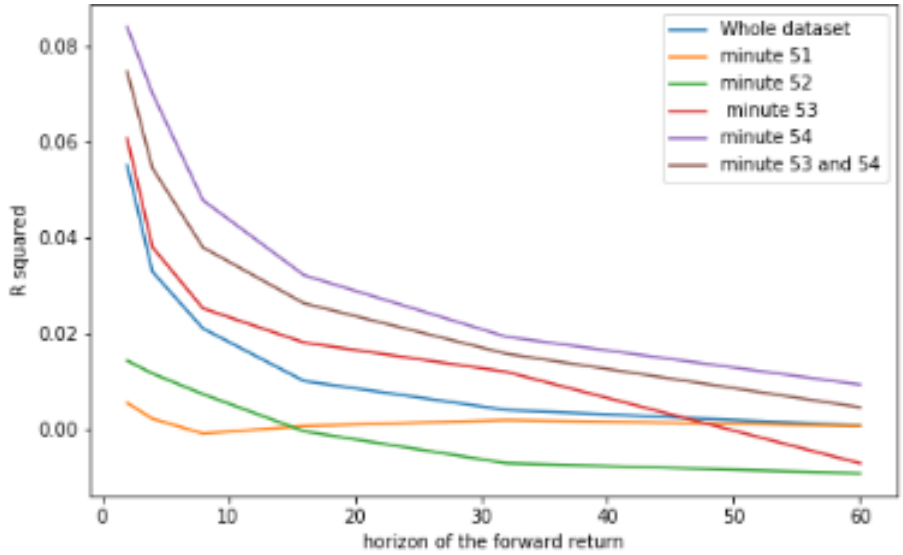


Figure 9: Evolution of the R squared for each minute between 3:50 and 3:55pm

The following plot (Figure 10) shows the evolution of the R^2 for each minute. Just like for previous plot, where I trained by minute, each minute behaves differently. For minute 51 and 52, the R^2 is very low, around 0. It is larger for minute 53, and is at its largest for minute 54. Again, it seems like the predictability of returns is improved when we get near 3:55pm, the NASDAQ cutoff time for MOC/LOC order entry.

This corresponds to a spike of orders coming in just before the cutoff time. Let's define the total volume as the sum of buy volume and sell volume on the imbalance feed . Let's also define the change in this total volume, between t seconds and t+10 seconds. Figure 11 is a plot of the mean of this change in the total volume between 3:50pm and 4pm, for 4 stocks : Apple, Amazon, Google and Amazon.
We can clearly see a spike at 3:55pm for all these stocks. If we take the mean percentage of change for all stocks (Figure 12), this spike goes up to 35% of the total volume just before 3:55pm.

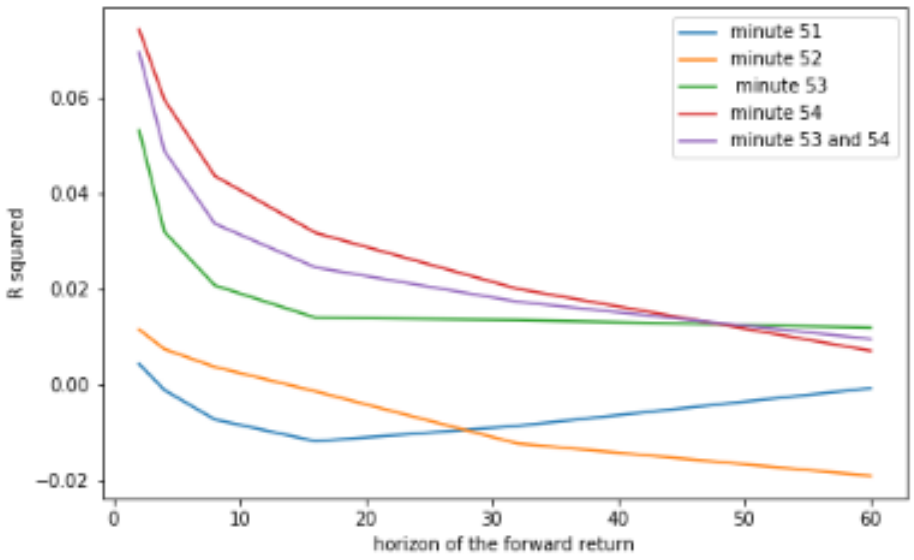


Figure 10: Evolution of the R squared for each minute between 3:50 and 3:55pm

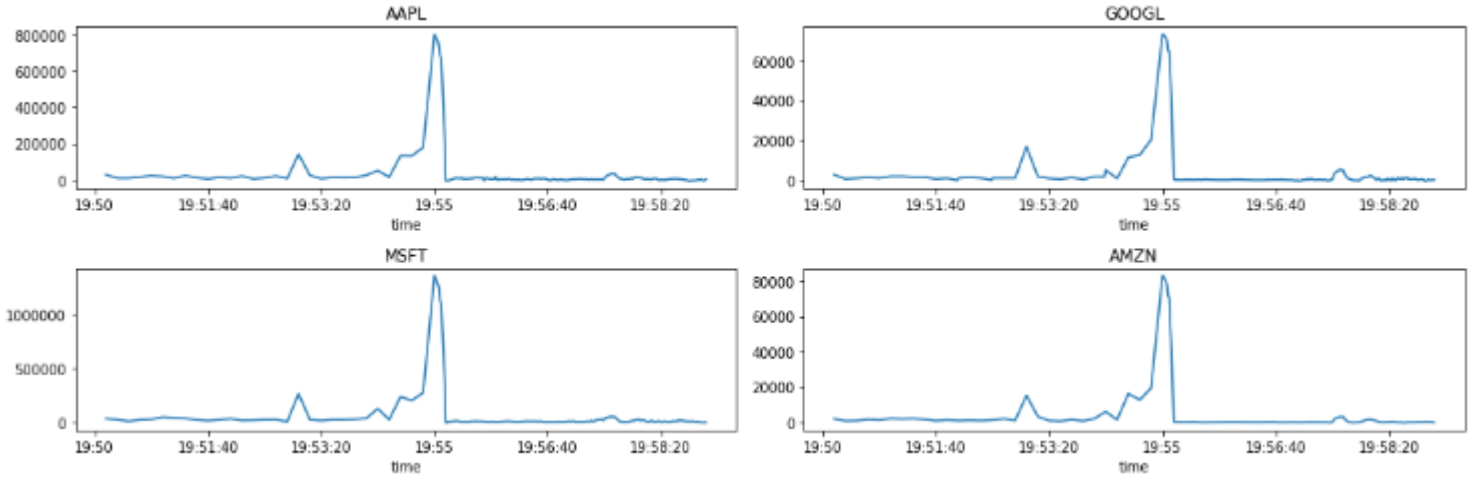


Figure 11: Evolution of changes in the order volume for different stocks

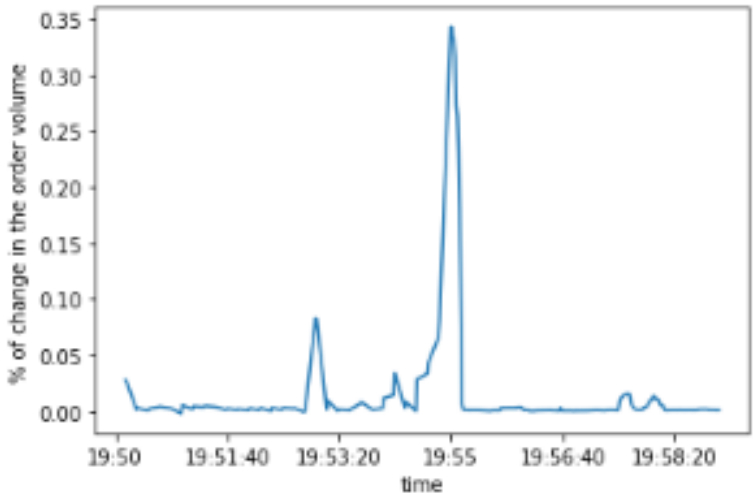


Figure 12: Evolution in the percentage of change in total order volume

3.1.7 Another metric

The R^2 is a metric of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model. I wanted a metric that would give me a measure of how much we can actually earn with this model, information that is not given by the R^2 metric.

I used a metric to compute how much of the spread we earn with the prediction at 60 seconds. To compute this metric, I used the following algorithm:

- Forecast the 60 seconds returns with the model, out of sample (in the test set)
- Regress the realised 1, 2 .....60 seconds forward returns against the forecasted 60 second return.
- For each of this regression, there is a regression coefficient β. Multiply the coefficient of the regression with the standard deviation of forecasted 60 second returns, out of sample.
- The result is the percentage of the spread we earn with the prediction.

It is a view on the path of realisation of the alpha. If, instead of forecasting the 60 second return out of sample, I do it in sample, the coefficient will grow

to exactly 1 as we regress 60 second returns on 60 second predictions. Are we realising 80% of the value of the alpha within 5 seconds? And what about beyond 60 seconds? Does the coefficient go above one, i.e. the trajectory continues?

Let's first plot the coefficients of the regression of the forecasted 60 seconds return in sample against the actual 1,2 .... 60 seconds return.

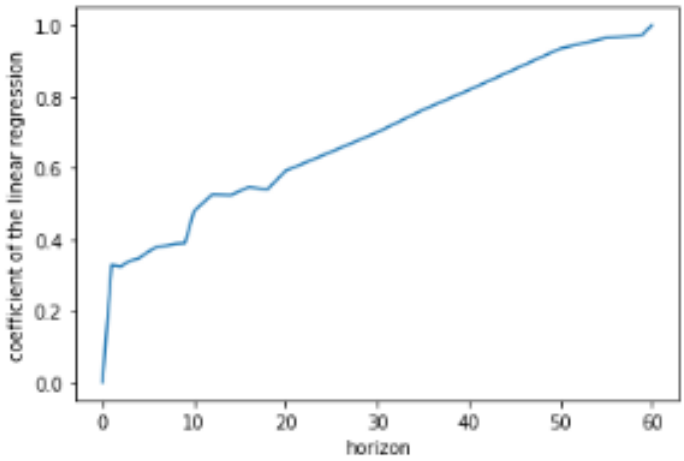


Figure 13: Evolution of the coefficient of the linear regression for each forward return horizon, in sample

There is a sudden increase in the first few seconds, then it grows linearly up to 1 for the regression of forecasted 60 seconds return against the actual 60 seconds return.

The following plot (next page) is the evolution of the coefficient of the regression times the standard deviation of forecasted 60 second returns, out of sample, for each return horizon.

We can see that we are realising the entire value of the alpha within 20 seconds. After 20 seconds, the curve flattens. For this model, the predictive power gets a lot weaker after the first few seconds. The percentage of the spread earned for the 2-second return horizon is already 28 %. It grows to 34% of the spread within 20 seconds. So we can earn 34 % of the spread with this model, 1/3 of the spread.

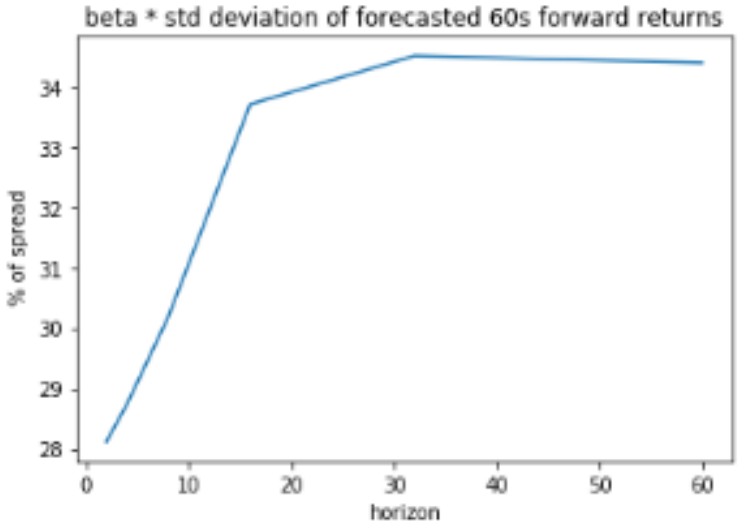


Figure 14: Evolution of the % of the spread earned with the prediction, out of sample

3.1.8 After 3:55 pm

Now that we have studied the model before 3:55pm, let's study the model after the 55th minute.
As a reminder : The NSDQ cutoff time for MOC/LOC order entry is 15:55. After 15:55, Imbalance-Only (IO) orders can be submitted before the close and late LOC orders can be submitted until 15:58, but IO and late LOC Orders cannot be updated or cancelled. Imbalance data is published every 10 seconds before 15:55 and every second after 15:55. [4]

We can add a new feature : the $diff_{price,t}$, defined in section 2.3 , because the indicative auction price is only published after 3:55pm.

Let's first have a look to the correlation table between the features and different forward returns. (next page)
We can see that all the correlations are very small. No feature has a high correlation with forward returns.

	b_return_4second	b_return_8second	b_return_16second	b_return_32second	ratio_imb	imb_return_10second_2	diff_price
b_return_4second	1.000000	-0.001306	-0.015736	-0.026656	0.008576	0.050619	-0.002169
b_return_8second	-0.001306	1.000000	-0.008806	-0.025176	0.006090	0.008436	-0.002944
b_return_16second	-0.015736	-0.008806	1.000000	-0.037150	0.002962	0.013166	-0.010605
b_return_32second	-0.026656	-0.025176	-0.037150	1.000000	-0.003485	0.009061	-0.019132
ratio_imb	0.008576	0.006090	0.002962	-0.003485	1.000000	0.004112	0.490356
imb_return_10second_2	0.050619	0.008436	0.013166	0.009061	0.004112	1.000000	-0.018133
diff_price	-0.002169	-0.002944	-0.010605	-0.019132	0.490356	-0.018133	1.000000
side	-0.017575	-0.012933	-0.010427	-0.005823	-0.376394	0.008011	-0.399651
f_return_2second	-0.004264	-0.004785	-0.015458	-0.022309	-0.013514	-0.021615	-0.009673
f_return_4second	-0.000785	-0.010032	-0.022816	-0.029663	-0.015910	-0.021913	-0.009799
f_return_8second	-0.007403	-0.016701	-0.030178	-0.040596	-0.019335	-0.020421	-0.009840
f_return_16second	-0.020427	-0.025189	-0.040516	-0.045120	-0.022640	-0.018494	-0.010940
f_return_32second	-0.031713	-0.036853	-0.046219	-0.017866	-0.033791	-0.016160	-0.015675
f_return_60second	-0.018965	-0.019499	-0.022411	-0.015163	-0.054870	-0.013652	-0.030225

Figure 15: Correlation table

```
[['b_return_4second', 'b_return_8second', 'b_return_16second', 'ratio_imb', 'imb_return_10second_2', 'diff_price'], ['f_return_
2second']]
0.0008101251281658062
```

Figure 16: Feature selection

Let's run a feature selection for the forward return of 2 seconds. We can see that with the feature selection, which gives the highest R^2 between every subset of features. As I said before, the R^2 remains very low. There is no subset of feature that gives a R^2 significantly different from zero.

Why? As I said before, The NSDQ cutoff time for MOC/LOC order entry is 15:55. After 15:55, Imbalance-Only (IO) orders can be submitted before the close. Which means that after 3:55pm, there is no more MOC/LOC orders, there are only IO orders, that are limit orders that provide liquidity during the opening cross and closing cross on the Nasdaq stock exchange. After 3:55pm, there are only orders that are given to offset an imbalance in the opening or closing cross. This might be an explanation of why we can't make price returns prediction with imbalance data.

3.2 NYSE

3.2.1 Data

Let's first focus on the time between 3:50 and 3:55pm. The dataset includes all trading days between 01/04/2020 and 01/08/2020, and there is a datapoint every second between 3:50 and 3:55pm, which is the frequency at which imbalance data is published.
At the close, a stock reverts to the exchange where it is listed. Here, we are focusing on S&P 500 stocks listed on NYSE venue. For the time period studied here, there were 360 such stocks.
The training set is 2 months long, and then there is a validation test (used to find the best features by features selection), followed by a 1-month-long test set.

Figure 17: Dataset separation

3.2.2 Features selection

First, the correlation map only shows very low correlations, not significantly different from zero. The results of the gridsearch feature selection are the same than for S&P stocks :
For every forward return horizon, two features are always selected : **the imbalance return** with a horizon of 10 seconds ($r_{imb,t,10}$) and **the imbalance ratio** ($imb_{t,1}$). Then for each forward return horizon, different backward returns are also selected, but they improve very slightly the R^2.
But the R^2 of the regression with the features selected is smaller than for NASDAQ stocks.

3.2.3 R squared computation

If we do the linear regression with the whole dataset, it gives a very low R^2, around 0.01 for 2-second forward returns.
Let's recall that D-quotes can be sent up until 3:59:50 PM, ten seconds before the close, regardless of the direction of the trade or whether an imbalance exists.
As for the cut-off time for LOC/MOC orders at 3:55pm where the predictability of returns is improved when we get near 3:55pm, we could try and see if the R^2 is better when we get near the D-quotes cut-off time, 3:59:50 pm.
Let's take the dataset of the 59th minute (seconds 1 to 40). The following plot is the evolution of R^2 for this dataset, for short horizon forward returns (2 to 16 seconds).

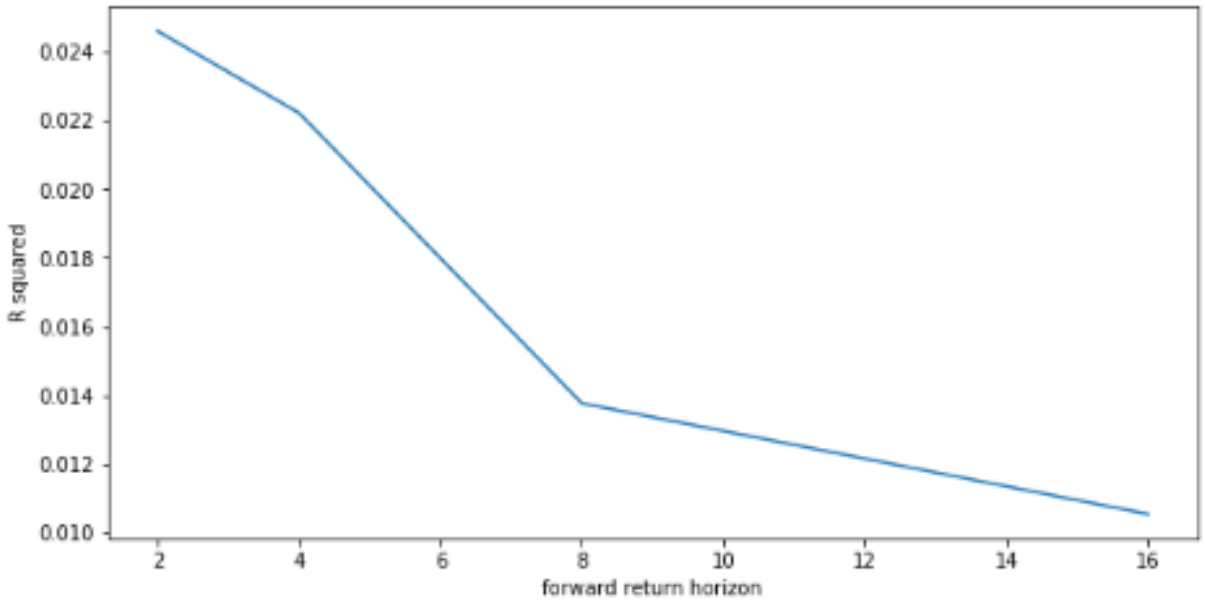


Figure 18: Evolution of R^2 with the horizon of forward return for minute 59

For very short horizons, the R^2 for minute 59, just before the NYSE D-quotes cut-off time, is noticeable. The predictability of the returns of the NYSE stocks seems to increase a bit when we get near the cut-off time for D-quotes. But it is the only time we can predict forward returns for S&P stocks on NYSE venue, between 3:50pm and 4pm.

4 RUSSELL 2000 Stocks

4.1 NASDAQ

4.1.1 Data

The Russell 2000 Index is a small-cap stock market index of the smallest 2000 stocks in the Russell 3000 Index. It has been embraced as the small cap index of choice for measuring the small cap market segment. Stocks on the Russell 2000 are also less liquid than those one on S&P 500.
Let's first focus on the time between 3:50 and 3:55pm. The dataset includes all trading days between 01/06/2020 and 01/07/2020, and there is a datapoint every 10 second between 3:50 and 3:55pm, which is the frequency at which imbalance data is published.
At the close, a stock reverts to the exchange where it is listed. Here, we are focusing on S&P 500 stocks listed on NASDAQ venue. For the time period studied here, there were 1209 such stocks.
The training set is 2 weeks long, and then there is a validation test (used to find the best features by features selection), followed by a 1-week-long test set. (shorter sets than before because there are a lot more stocks)

```
[['b_return_4second', 'b_return_8second', 'b_return_16second', 'ratio_imb', 'imb_return_10second_2'], ['f_return_2second']]
0.0017487033577783 25
[['b_return_4second', 'b_return_8second', 'ratio_imb', 'imb_return_10second_2'], ['f_return_2second']]
0.0017426885868890851
[['b_return_4second', 'b_return_16second', 'ratio_imb', 'imb_return_10second_2'], ['f_return_2second']]
0.0017383314107267678
```

Figure 19: Feature Selection

4.1.2 Features Selection

I ran my grisearch algorithm on the whole dataset. With the whole dataset, the best R^2 we can get is very low. Here are the three best subsets of features that could get us the best R^2 for 2 seconds returns:

```
[['b_return_4second', 'b_return_8second', 'b_return_16second', 'ratio_imb', 'imb_return_10second_2'], ['f_return_2second']]
0.001748703357778325
[['b_return_4second', 'b_return_8second', 'ratio_imb', 'imb_return_10second_2'], ['f_return_2second']]
0.0017426885868890851
[['b_return_4second', 'b_return_16second', 'ratio_imb', 'imb_return_10second_2'], ['f_return_2second']]
0.0017383314107267678
```

Figure 20: Feature Selection

Those are obviously very low R^2, not significantly different than 0 . Again, the correlation table also shows very low correlations. It looks like the features can't predict well the forward returns on the whole dataset.
Remembering that for S&P stocks on NASDAQ, the prediction is way better for minutes 53 and 54, I tried to see if it was the same for RUSSELL 2000 stocks.

Let's see what gives the features selection for 2 seconds froward returns, on minutes 53 and 54 :

```
[['b_return_4second', 'b_return_8second', 'ratio_imb', 'imb_return_10second_2'], ['f_return_2second']]
0.008824472588036092
[['b_return_4second', 'ratio_imb', 'imb_return_10second_2'], ['f_return_2second']]
0.008671463223770881
[['b_return_8second', 'ratio_imb', 'imb_return_10second_2'], ['f_return_2second']]
0.008503058217424253
```

Figure 21: Feature Selection for minute 53 and 54

And on minute 54 :

```
[['b_return_8second', 'ratio_imb', 'imb_return_10second_2'], ['f_return_2second']]
0.011088436497916243
[['ratio_imb', 'imb_return_10second_2'], ['f_return_2second']]
0.011053214006046197
[['b_return_4second', 'b_return_8second', 'ratio_imb', 'imb_return_10second_2'], ['f_return_2second']]
0.010782907741487047
```

Figure 22: Feature Selection for minute 54

Here again, the R^2 is way bigger for minutes 53 and 54, and even bigger for only minute 54. It seems like the predictability of returns is improved when we get near 3:55pm, the NASDAQ cutoff time for MOC/LOC order entry. We will see that a little bit more in details later .

The correlation table (for minutes 53 and 54) shows significant correlations between the imbalance ratio, the imbalance return and the different targets.

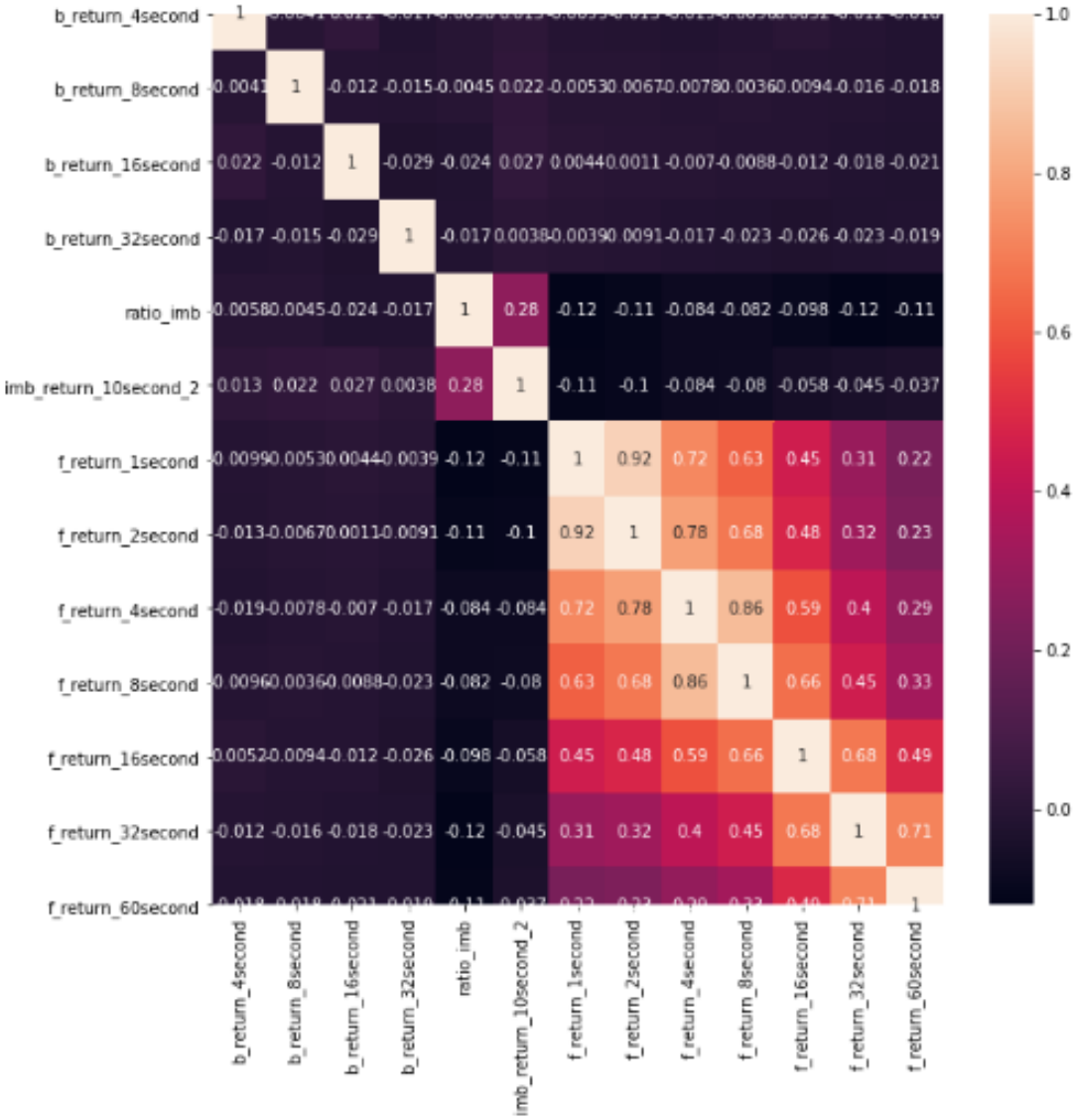


Figure 23: Correlation map

So this confirms the features selected by the gridsearch algorithm :

For every forward return horizon, two features are always selected : **the imbalance return** with a horizon of 10 seconds ($r_{imb,t,10}$) and **the imbalance**

ratio ($imb_{t,1}$). Then for each forward return horizon, different backward returns are also selected, but they improve very slightly the R^2. The two main features are the same for the S&P stocks.

For minutes 53 and 54, the features selected for each forward return horizon (We are not going beyond 16 second forward horizon, because the R^2 is negative) :

Horizon	Features selected
2 seconds	'b return 4second', 'b return 8second', 'ratio imb', 'imb return 10second'
4 seconds	'b return 4second', 'b return 8second', 'ratio imb', 'imb return 10second'
8 seconds	'b return 8second', 'ratio imb', 'imb return 10second'
16 seconds	'ratio imb', 'imb return 10second'

4.1.3 R squared computation

I decorrelated the features with the same methods as for S&P stocks.

This is the plot (next page) of the R^2 for minutes 53 and 54, over different time horizons, for python baseline (mean of forward returns over the whole dataset).
We can see that the R^2 is quite low, around 1 %, even for short-term horizon, and becomes negative beyond 30 seconds horizon. The R^2 is way smaller than for S&P stocks.

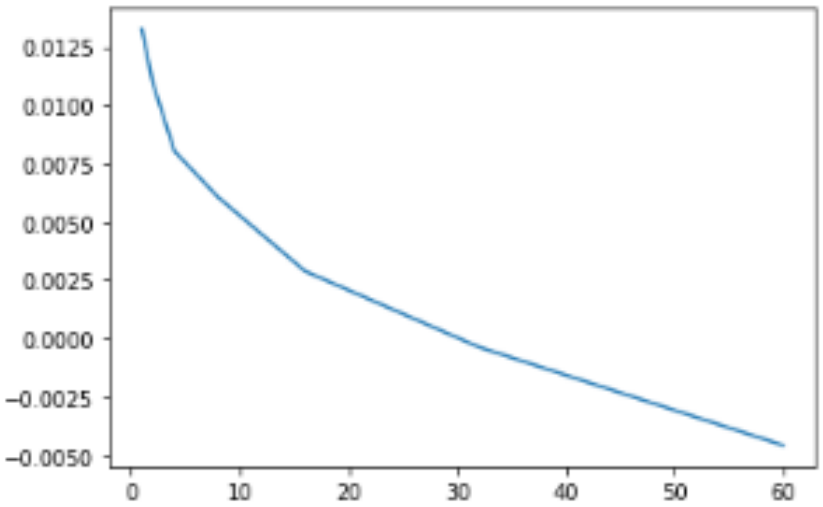


Figure 24: Evolution of the R squared of the linear regression for each forward return horizon

4.1.4 Another metric

I used the same metric as for S&P stocks, a metric to compute how much of the spread we earn with the prediction at 60 seconds. Let's recall how to compute this metric :

- Forecast the 60 seconds returns with the model, out of sample (in the test set)
- Regress the realised 1, 2 .....60 seconds forward returns against the forecasted 60 second return.
- For each of this regression, there is a regression coefficient β. Multiply the coefficient of the regression with the standard deviation of forecasted 60 second returns, out of sample.
- The result is the percentage of the spread we earn with the prediction.

It is a view on the path of realisation of the alpha.
Let's first plot the coefficients of the regression of the forecasted 60 seconds return in sample against the actual 1,2 .... 60 seconds return.

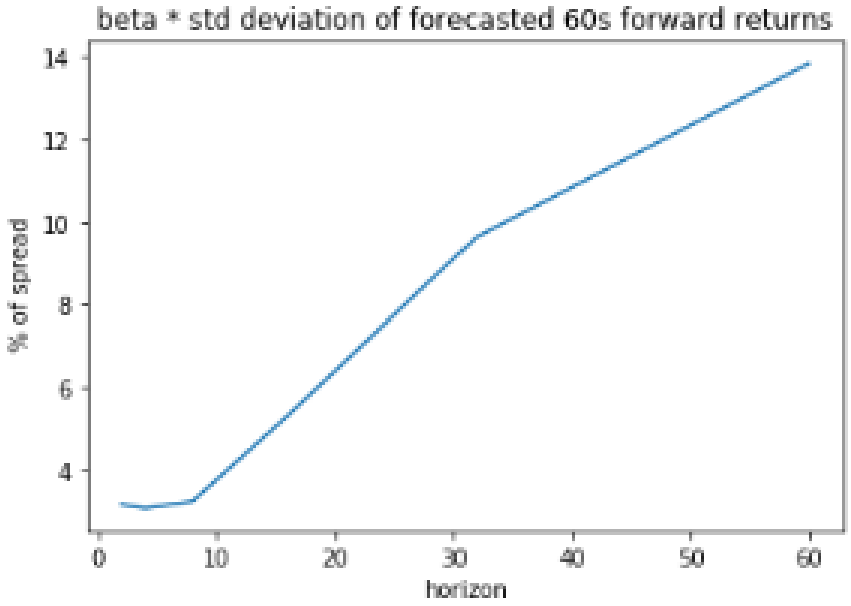


Figure 25: Evolution of the % of the spread earned with the prediction, out of sample

The percentage of the spread earned for the 2-second return horizon is about 3 %, which is quite low. It grows to 14%. But it looks like it continues to grow beyond 60 seconds, the curve doesn't seem to flatten.
So I tried to continue beyond 60 seconds to see if the signal has explanatory power beyond that time horizon.

I plotted the coefficients of the regression of the forecasted 60 seconds return in sample against the actual 1,2 .... 250 seconds return. (next page)
We can observe that the signal seems to have explanatory power beyond 60 seconds time horizon. It continues to grow and finally starts to flatten beyond 200 seconds time horizon.
It grows to more than 25% of the spread within 20 seconds. So we can earn 25 % of the spread with this model, 1/4 of the spread.
The main difference with S&P stocks is the fact that the curve doesn't flatten after 20 seconds. We are not realising the entire value of the alpha within 20 seconds. On contrary, it continue to grow for long time horizons.
This may be due to the fact that Russell 2000 stocks are generally less liquid that S&P stocks.

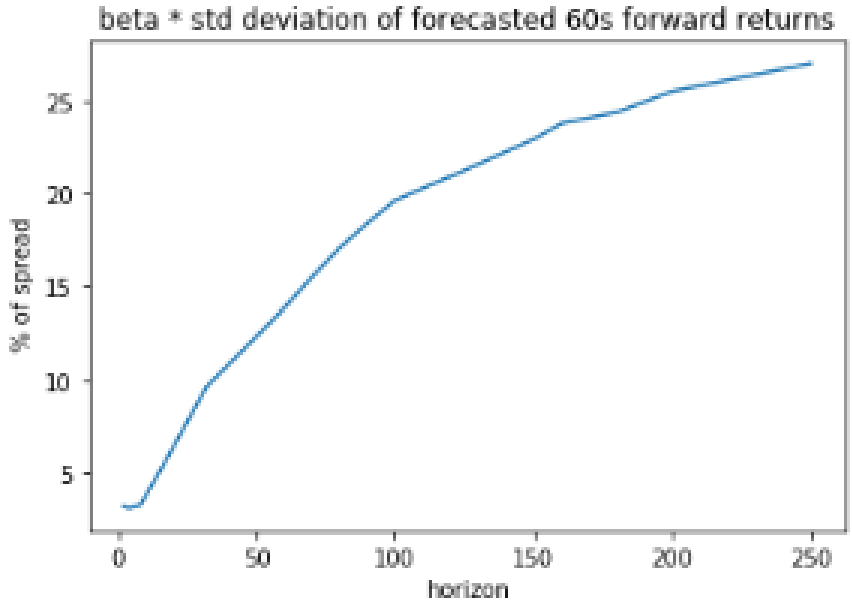


Figure 26: Evolution of the % of the spread earned with the prediction

4.1.5 After 3:55 pm

Now that we have studied the model before 3:55pm, let's study the model after the 55th minute.
As a reminder : The NSDQ cutoff time for MOC/LOC order entry is 15:55. After 15:55, Imbalance-Only (IO) orders can be submitted before the close and late LOC orders can be submitted until 15:58, but IO and late LOC Orders cannot be updated or cancelled. Imbalance data is published every 10 seconds before 15:55 and every second after 15:55. [4]

Like S&P stocks, when I ran features selection, the regression with the features selected had a very low R^2, not significantly different from zero. But something got my attention : a relatively high correlation between the 60 seconds forward return and the diff price feature : a feature that reflects the difference between the indicative auction price and the actual midquote. As a reminder :

$$diff_{price,t} = \frac{log(\frac{p_t}{p_{auction,t}})}{\tilde{s}}$$

where p_t is the midquote at time t

and $p_{auction}$ is the indicative auction price at time t
and $\tilde{s}$ is the the median spread over the last 20 days

But the R^2 of the regression with diff price as a feature is close to zero. So I tried to regress forward returns against diff price for each minute between 3:55 pm and 4pm. And I found a R^2 of more than 0.05 for minute 57. What could explain this behaviour? The only thing that is happening at this time is that late LOC orders can be submitted until 15:58.

Let's split the dataset of the data between 3:57pm and 3:58 pm into 6 other datasets, each one including 10 seconds of the original dataset, and compute the R^2 for each one.

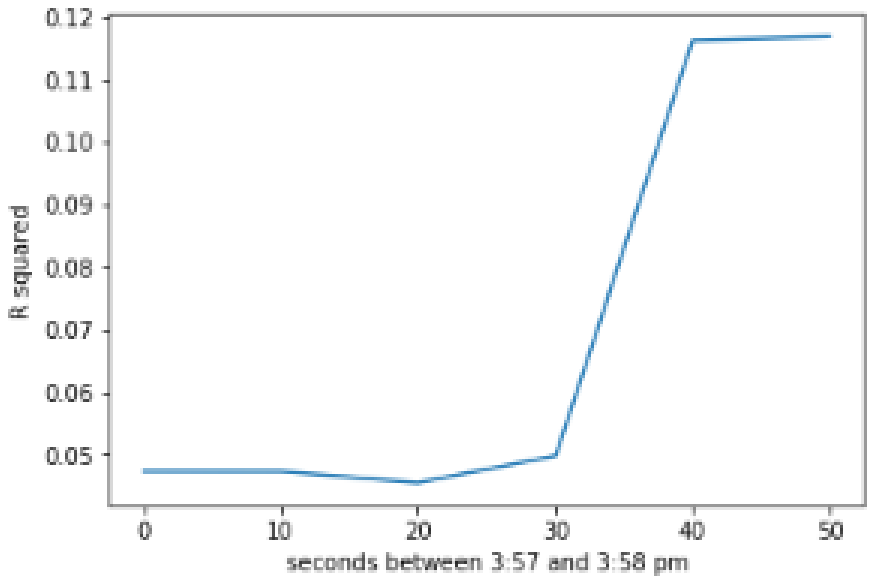


Figure 27: Evolution of the R^2 between minutes 57 and 58

There is a sudden increase in the R^2 after second 40. As wet get closer to 3:58 pm, the R^2 gets higher. So the explanation of this behaviour by the fact that late LOC orders can be submitted until 15:58. makes sense. Beyond 3:58, the R^2 of this regression is close to 0 .

Let's separate the dataset into 1-second bins instead of 10 seconds bins, to have a better precision. The plot is the following:

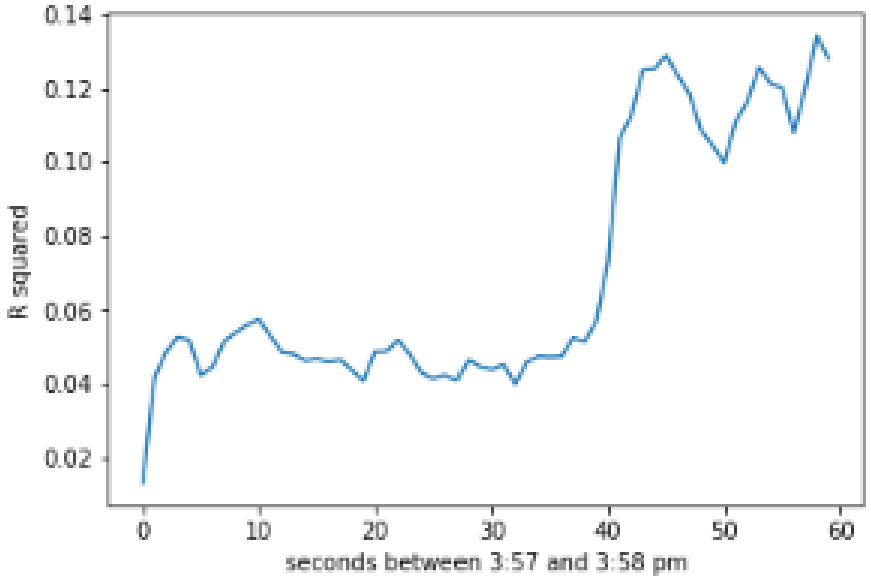


Figure 28: Evolution of the R^2 between minutes 57 and 58

Here again, we can see that the sudden increase in the R^2 seems to be exactly on the 40th second. As wet get closer to 3:58 pm, the R^2 gets higher

Conclusion

In this thesis, we have introduced an approach to US stocks price forward-return forecasting during the 10 minutes before the closing auction, using as explanatory variables the data published by venues during these last ten minutes, displaying the volume imbalance between sell and buy orders.

- First, we have built a model for large-cap stocks, using the S&P 500 stocks. We did a different model for each venue, because NASDAQ and NYSE closing auctions have different behaviours. For NASDAQ venue, the R^2 of the linear regression for tha dataset before 3:55pm is quite large and we can predict up to 30 % of the spread. We also noticed that the R^2 increases for minutes 53 and 54, as we get closer to MOC and LOC cut-off time in NASDAQ venue. After this cut-off time, we don't have any conclusive results, forward returns seem to be a lot more difficult to predict.

- Then we did the same for NYSE venue. The results were worse than for NASDAQ, but we still had a noticeable R^2 for short horizon forward returns just before the NYSE D-quotes cutoff time, 3:59:50 pm.

- Then we worked on RUSSELL 2000 stocks, that is an index of mostly small-cap symbols. First for NASDAQ venue, where we have a R^2 significantly different from 0 for the dataset between 3:50pm and 3:55pm. After 3:55pm, there is a noticeable increase in the R^2 of the linear regression when we get near 3:58pm, cut-off time for late LOC orders.

Some of these results are really encouraging , and there are still things to work on and improve. Among these improvements to be made, we can cite:

- RUSSELL stocks on NYSE venue. I have only started to work on this data, with very little results. But I didn't have time to work in depth on this dataset. I really would like to work more on this dataset to see if we can have some interesting results with this dataset.

- I also want to work more on the results I go at the 57th minute, and try to see if I have the same results for another dataset. That would confirm the influence of the closure of the late LOC orders, that can be submitted until 15:58, on stock return predictability.

References

[1] **Dimension Reduction Based on Orthogonality —a Decorrelation Method in ICA**
Kun Zhang, Lai-Wan Chan Department of Computer Science and Engineering
The Chinese University of Hongkong, Hongkong

[2] **Decorrelating features**
https://nicholastsmith.wordpress.com/2018/10/03/decorrelating-features-using-the-gram-schmidt-process/

[3] **Predictability of Stock Returns: Robustness and Economic Significance**
M. Hashem Pesaran and Allan Timmermann
The Journal of Finance
Vol. 50, No. 4 (Sep., 1995), pp. 1201-1228

[4] **Closing volume discovery**
https://www.fixglobal.com/home/closing-volume-discovery/

[5] **Closing auction**
https://www.blackrock.com/corporate/literature/whitepaper/viewpoint-a-global-perspective-on-market-on-close-activity-july-2020.pdf

[6] **Stock return predictability: Evidence from a structural model**
Pholile Dladla Christopher Malikane
Macro-Financial Analysis Group, School of Economic and Business Sciences, University of the Witwatersrand, South Africa
International Review of Economics & Finance Volume 59, January 2019, Pages 412-424

[7] **OLS regression**
https://towardsdatascience.com/understanding-the-ols-method-for-simple-linear-regression-e0a4e8f692cc

[8] **Forecasting: Principles and Practice**
G. Athanasopoulos, R. Hyndman (2013)

[9] **Applied Linear Regression Models**
M. H. Kutner, C. J. Nachtsheim, J. Neter (2004)

[10] **intelligently allocate benchmark closing price**
https://www.bloomberg.com/professional/blog/intelligently-allocate-benchmark-closing-price/

[11] **The role of closing auctions in wellfunctioning markets**
Norges Bank Investment Management

[12] **D quotes**
https://www.bacidore.com/post/the-nyse-d-quote-the-disney-fastpass-of-trading