Closer to Size: A Preliminary Investigation into the Predictability of Closing Auction Sizes in US Equities

Allison Bishop Adele Shahi Yuqian Zhang

Abstract

We describe Proof Trading's initial research into the predictability of closing auction sizes. This is a first step in our process of designing an algo for trading into the close.

1 Introduction

The US equities market can sometimes feel like a procrastinator's paradise. It wakes up sleepily around 9:30 am, shuffles some papers around for several hours, takes a bit of an unofficial lunch break, and then snaps to attention for an action-packed half hour or so leading up to the closing bell at 4 pm. The day typically ends with a heavy exclamation mark - a closing auction that represents a considerable fraction of the total activity over the day. The closing auction is a structurally unique opportunity for buyers and sellers to convene in time and space, like high school cliques being forced together for a school assembly.

The closing auction can provide a fair amount of cover for large trades that might otherwise move the market. But misjudging the amount of cover on any particular day can lead a trader to expose too much interest and move the market against themselves, or to expose too little and miss an opportunity to complete more of their order while staying under the radar. In either direction, mis-estimation can be costly.

There are many reasons to suspect, though, that decent predictions of closing auction size are possible. First, historical data is widely available across symbols and across years, making it easy to test out various prediction strategies and compare their outputs to the true outcomes. Second, the same reasons that may lead someone to trade in the close may lead them to trade also throughout the day and/or in the opening auction, meaning that measurements of trading activity earlier in the day may be correlated with closing auction volumes.

In this paper, we present our initial work on developing and evaluating models that attempt to predict closing auction sizes as a function of historical data from previous days as well as measurements that can be made earlier in the same trading day. Since we know the true close sizes in a historical data, this can be approached as a classic supervised learning problem. We will:

- Explore features that may be helpful in prediction
- Explore metrics for grading the quality of our predictions
- Explore function forms for expressing our prediction as a function of our (hopefully) predictive variables
- Explore algorithms for choosing specific functions that score well on our chosen metrics

After all putting all of this together, we will have arrived at an initial model. There is certainly room for future improvement though, as our choices at each of these steps can affect our outcomes and should be continually revisited.

While one can follow this template for all supervised machine learning problems, which steps are the most difficult or the most crucial often varies as a function of the underlying data and the underlying context. In many financial data problems, finding features that are robustly meaningful is often a challenge due to high levels of noise. As we will detail later, we find that not to be the case here. Somewhat to our surprise, we quickly found a few different basic features that were highly correlated to closing auction sizes. But the choices of metrics, function forms, and algorithms were a bit more challenging than usual, as we saw good reasons to deviate from typical defaults. Ultimately though, we arrive at what we think is a relatively simple and explainable model that robustly outperforms a default

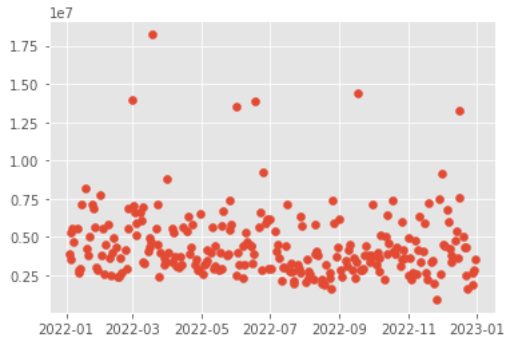
baseline of using historical averages over recent days (e.g. the average of close sizes in the same symbol over the previous 20 days).

We are certainly not the first people to investigate this problem. Far from it! Closing size prediction is the sort of the thing that nearly every trader, every broker, every proprietary trading firm, and every hobbyist has their own take on. Most of those takes are not disclosed as part of the public scientific record. As with many of our quantitative research efforts at Proof, we chose to start here with a clean slate, investigating the problem from first principles from a machine learning perspective, rather than scouring the financial literature. Since we cannot know how our findings here compare to undisclosed proprietary solutions, we do not make any claims of novelty. But we do think what we lay out here is a framework that others may find understandable and useful, as it is derived through a systematic process and can be adapted at various points to fit more specific goals in more specific contexts.

2 Data Selection and Feature Exploration

In any data science problem, the first thing we have to decide is: what is our data set? Since we expect our data to be noisy, we will probably want to pull a lot of historical data for building our models. We will also want to cover important seasonal events, such as monthly options expirations, quarterly earnings releases, etc. For this reason, we choose to pull historical data from all of 2022 for 1000 symbols as our training set. The 1000 symbols were chosen according to top notional value traded. Data for the same symbols from 2023 will be used to test our final model candidate, but not used during training. This helps us avoid over-fitting as well as confirming that any patterns we find stay stable enough over time to remain predictive on later data.

After pulling historical training data, we can take a look to get a basic sense of how close sizes behave. Here is a scatter plot of the closing auction sizes for BAC over the year 2022:



There's not too much we can see by eye here, other than the fact that there is a fair amount of noise and some noticeable outliers floating above the rest of the data. A natural question is: do these outliers occur on special known event dates, like days when earnings are released? We'll enrich our data with columns that mark special events in order to find out.

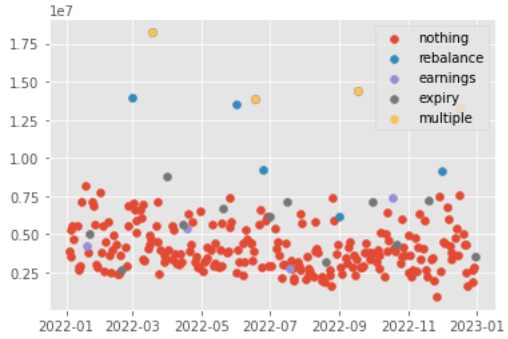
2.1 Special events

There are several different kinds of events that can be expected to affect closing auction sizes. Options expiry days, for example, are pre-scheduled events that may lead to increased trading. There are monthly expirations on the third Friday of every month, as well as quarterly expirations. Some of these expiration days are called "triple witch" or "quadruple witch" events because multiple things are happening at once: e.g. stock options, index futures, and index options all expiring on the same date. Since option expiry events are scheduled in advance, we can mark them in our data set and potentially use this information in making our predictions. For now, we will do this in a fairly simple way. We'll create a binary feature called "IsExpiry" that is equal 1 when it is an option expiry day and equal to 0 when it is not. For now, all of the different kinds of expirations (monthly, quarterly) will be grouped together as "1" values.

Earnings releases are another category of events that we would expect to heavily affect trading volumes. For each stock, quarterly earnings releases are scheduled in advance, so we might expect levels of activity to be impacted both before and after such announcements. For now, we will consider a basic binary feature called "AfterEarnings" that is equal to 1 on the first day of trading after an earnings announcement, and equal to 0 otherwise.

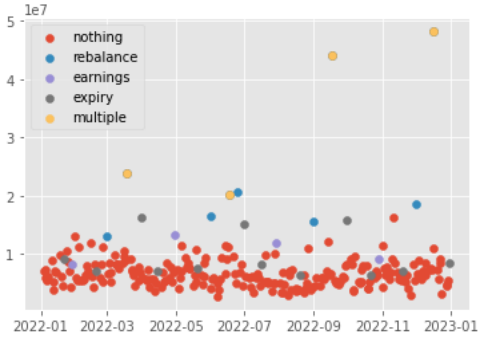
A third important category of events are index rebalances. The Russell Reconstitution is a yearly event that finishes on a particular day, while the S&P and MSCI indexes rebalance four times a year. For now, we create a simple feature called "IsRebalance" that is equal to 1 on rebalance days and equal to 0 otherwise.

Here is a color-coded scatter plot for the same BAC 2022 data with these kind of events identified. The "multiple" category indicates days that were special in more than one way.



For BAC at least, it seems that monthly options expirations and earnings releases in 2022 were not a big deal in terms of affecting closing auction sizes, but rebalance and multiple event days were. In 2022, the multiple event days were the four days when S&P rebalance coincided with a witch event (so we might suspect here that the "IsRebalance" variable is the most relevant one for BAC).

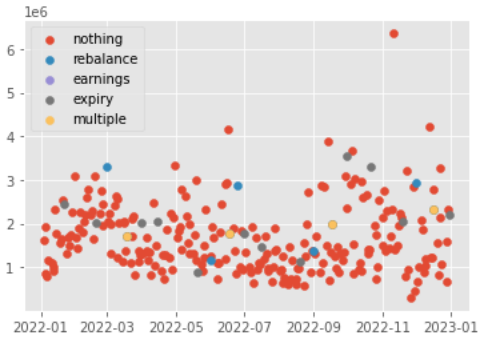
We can see somewhat different effects here for different symbols. Here is a color-coded plot for AAPL closing sizes for 2022:



Here we see that quarterly expirations and earnings releases may be having a more meaningful impact on AAPL closing sizes in this time period. It is important to keep in mind then, that our indicator variables for special events may have very different relevance for different symbols. This should inform how we ultimately incorporate them into our models.

2.2 Symbol types

We might also expect trading dynamics around the close to be different for ETFs vs. common stocks. Here is what closing sizes looked like for SPY in 2022, for example:



We notice that the scale here for the y-axis is smaller than BAC and AAPL, despite the fact that volume in SPY over the day is typically much higher than volume in BAC and AAPL. We might expect a general muting of close sizes in ETFs vs. common stocks, as well as a decreased or differing impact of special events.

2.3 Historical and intra-day features

A common basic prediction strategy is to use a rolling average of close sizes for recent days in the same symbol. For example, we might predict that today's closing auction in BAC will be the average size of the last 20 or 30 days of closing sizes for BAC. It is worth noting that this baseline already has several desirable features. For one, it yields predictions that are customized by symbol and dynamic over time, meaning that the predictions adjust higher when closing sizes in a symbol trend higher and lower when they trend lower (though with a little lag due to the length of trailing window). However, averages are also volatile and can be unduly swayed by outliers. One approach to mitigating this would be to remove special event days from the average calculation and/or exclude numbers that are out of line with their temporal peers. A simpler approach is to use medians instead of averages, since medians are inherently more robust. We will try both historical averages and medians as baseline predictors and features in our models later once we have decided on a metric for grading their respective performance.

Beyond this basic trailing window approach, our selection of features is very open-ended. There are an endless number of things we can measure about historical and intra-data that we might guess are somehow related to closing auction sizes. A kitchen-sink approach would be to collect as many of them as possible and then throw them all into the mix later when we consider functional forms and model-fitting. However, such an approach would have us starting with a fairly high level of complexity even as we try to fit basic functional forms. Having too many candidate features to start with isn't a big problem when you have a lot of clean and non-noisy data to help you sort out later which features are really needed, but when data is sparse and/or noisy, you may be setting yourself up for over-fitting and confusion. Since financial data tends to be extremely noisy, we will take a more conservative approach and pre-vet potential features by looking at their correlation (or lack thereof) with the phenomenon we are trying to predict. In our case, this means checking if a feature is meaningfully correlated with close sizes in a way that is likely to be valuable in the context of a prediction model. This is naturally a heuristic, and could mean we end up discarding features that aren't directly correlated with our prediction target, but would combine with other features in subtle ways to improve predictions. Nonetheless, we think that limiting ourselves to a small feature set that is clearly correlated with our prediction target is a good way to set us up for success in building an initial model that we can then improve and enhance iteratively.

2.3.1 An opening size feature

We might suspect that there is some correlation between opening auction sizes and closing auction sizes. Days where opening auctions are particularly large or small might also tend to be days where closing auctions are particularly large or small. This kind of correlation is typically captured by a correlation coefficient. The Pearson correlation coefficient, for example, is defined as follows for two random variables X and Y:

$$Pearson(X, Y) := \mathbb{E}\left[\frac{(X - \mu_X)(Y - \mu_Y)}{\sigma_X \sigma_Y}\right].$$

There are several things going on in this expression that we should stop and absorb. First, μ_X and μ_Y denote the average values of X and Y respectively, so subtracting these effectively re-centers X and Y around 0. Next, σ_X and σ_Y denote the standard deviations of X and Y respectively, so dividing by these effectively scales each of X and Y to have standard deviation 1. It may be revealing to write the whole expression a bit differently. We can define:

$$\tilde{X} := \frac{X - \mu_X}{\sigma_X},$$

$$\tilde{Y} := \frac{Y - \mu_Y}{\sigma_Y}.$$

Then we have:

$$Pearson(X, Y) = Pearson(\tilde{X}, \tilde{Y}) = \mathbb{E}[\tilde{X}\tilde{Y}].$$

The main point here is that a product of two zero-centered variables captures what we intuitively mean by correlation: if it tends to be positive, than both variables tend to agree in sign, and if it tends to be

negative, then both variables tend to disagree in sign. If the variables are independent, we expect them to agree and disagree in sign in roughly equal measure, and the correlation coefficient should come out to be close to 0. The correction for the standard deviations (the dividing by $\sigma_X \sigma_Y$), doesn't change the sign of the correlation coefficient, but is helpful for interpreting its magnitude. With this correction, the Pearson coefficient is always in the range from -1 to 1, with values near -1 representing strong negative (a.k.a. inverse) correlation, values near 1 representing strong positive correlation, and values near 0 representing a lack of correlation.

Of course, we don't actually know what our underlying random variables are, we only have samples of them (e.g. samples of opening auction and closing auction sizes). So what we're actually doing when we compute a Pearson correlation coefficient is averaging over the samples that we have. More precisely, if we have pairs of (evenly weighted) samples for X and Y, say $(x_1, y_1), \ldots, (x_n, y_n)$, we compute:

$$\mu_X := \frac{1}{n} * \sum_{i=1}^{n} x_i, \quad \sigma_X := \sqrt{\left(\frac{1}{n}\sum_{i=1}^{n} x_i^2\right) - \mu_X^2}$$

$$\mu_Y := \frac{1}{n} * \sum_{i=1}^{n} y_i, \quad \sigma_Y := \sqrt{\left(\frac{1}{n}\sum_{i=1}^{n} y_i^2\right) - \mu_Y^2}$$

$$Pearson(X, Y) = \sum_{i=1}^{n} \frac{(x_i - \mu_X)(y_i - \mu_Y)}{\sigma_X \sigma_Y}$$

Writing it out this way, it is clear that the Pearson correlation is an average over empirical data and so it is vulnerable to outliers. We might think that the standard deviation correction protects us from this, but it doesn't really. Just because we've scaled our data does not make it Gaussian or any other well-behaved shape, and our calculation here can be heavily impacted by small sets of extremal values. The Pearson coefficient is a good test on its own when variables are relatively well-behaved and have a somewhat linear relationship between them, but on noisy data it can be unreliable.

A more robust version is the Spearman correlation coefficient, which looks at the ranks of values for X and Y, rather than the values themselves. To compute a Spearman coefficient, we sort the values $\{x_i\}$ from least to greatest, and replace each value with its rank in this sorting. For example, values of $\{1001, 342, 56077, 2455\}$ would be replaced by $\{2, 1, 4, 3\}$. We do the same for the values $\{y_i\}$. We then compute a Pearson coefficient on the respective ranks of X and Y values. The fact that we are only looking at ranks instead of raw values mutes the effects of outliers and allows us to better capture relationships between X and Y that may be highly nonlinear. Because of these advantages, we prefer to use Spearman coefficients to test for correlations in our data.

If we want to test for correlation between the opening and closing auction sizes in a single symbol, directly computing a Spearman coefficient with X representing close sizes and Y representing open sizes could make sense, but there are a few pitfalls we can anticipate. One is that volume might holistically trend up or down over time, affecting both open and close sizes. This would represent a correlation, but not a particularly helpful one. This kind of trend is already captured in using historical averages over the last 20 days, for example, and so adding open sizes as a feature would not add much value if this is the only kind of correlation present. Another issue is that for a single symbol over the span of a year, we only have about 250 data points. This is not a lot of data, so we might get some strange results by chance. It's also not really clear what we should do to interpret a set of 1000 correlation coefficients computed symbol by symbol. Surely some will be negative, some will be positive, some will be large, some will be small, etc. It's not at all clear what conclusions we can draw overall this way.

To address the potential influence of volume trends over time, we can use the ratios of open and close sizes over their historical averages or medians rather than the raw sizes themselves. [We will use medians for their greater robustness, but averages would probably be fine too.] Letting $OpenSize$ denote the size of the opening auction and $OpenHist$ denote a historical median of opening auction sizes over the prior 20 days, we define the feature:

$$OpenF := \frac{OpenSize}{OpenHist}.$$

Analogously, we define:

$$CloseF := \frac{CloseSize}{CloseHist}.$$

Dividing by a trailing historical benchmark should help mute volume trends over time and isolate the kind of day-to-day correlation between open and close sizes that we are more interested in. It also makes

it reasonable to aggregate data across symbols, now that things are a bit more apples-to-apples. We can now get a rough sense of whether open auction sizes might have any predictive power with respect to close sizes by computing a single Spearman coefficient over all of our data, treating $OpenF$ as variable X and $CloseF$ as variable Y. However, we might still worry that any correlation we find could simply be due to special event days causing *both* $OpenF$ and $CloseF$ values to be larger than usual. So we'll exclude special days (as indicated by our AfterEarnings, IsExpiry, and IsRebalance variables) from our computation. The resulting Spearman coefficient is 0.17 if we weight data points equally or 0.18 if we weight them proportionally to the notional value traded for each symbol/day.

This looks promising! There seems to be a positive correlation! But there is still a remaining issue. We can see the number is positive, but is it *meaningfully* positive? Could it be explained by chance rather than an underlying relationship between the variables? To get a sense of this, we will use Monte Carlo simulations. In each simulation, we will randomly and independently permute the ranks of all of $OpenF$ and $CloseF$ values across our data set and will then compute a Spearman coefficient again. We'll then see what fraction of our simulations yield a correlation coefficient with an absolute value at least as large as we originally observed. Doing this for 100 simulations, the fraction is 0 (regardless of whether we use uniform or notional value weights). This is strong evidence that $OpenF$ can be a helpful feature in closing size prediction. Unsurprisingly, the correlation coefficient increases even further (to around 0.24) if we remove ETFs from our data set (since they may have systematically differing auction behaviors).

2.3.2 A late-in-the-day volume feature

We might also suspect that volumes throughout the day are correlated with closing auction sizes. In particular, we'll consider volume traded between 9:30 and 3:45 pm. We'll call this $LateSize$, as it represents the volume traded up to this late point in the day. Similar to our open size feature, we then define:

$$LateF := \frac{LateSize}{LateHist},$$

where $LateHist$ represents a historical median of the $LateSize$ quantity over the prior 20 days.

We will screen this $LateF$ for correlation with closing sizes in the same way that we screened the $OpenF$ feature. We'll compute a Spearman coefficient for $LateF$ and $CloseF$ over the non-special days in our data, and use Monte Carlo simulations to get a sense of whether its magnitude is meaningful. The resulting correlation coefficient is about 0.30, a greater magnitude than observed across any of 100 Monte Carlo simulations. Like in the case of $OpenF$, this coefficient is relatively insensitive to weighting data points equally or by notional value, and it increases when we exlude ETFs from our data set. (In this case, it increases to about 0.36.)

2.3.3 Imbalance feed features?

We would naturally suspect that features of imbalance feeds could help in predicting closing auction sizes. However, these feeds are only available very close to the time of the auction, and so using such features may be prohibitive, depending on the intended application. For MOC and LOC order types (a.k.a. "market on close" and "limit on close"), orders have to be submitted before the imbalance feeds begin releasing any information. Such feeds are also prohibitively expensive to purchase real-time access to, as they are not part of the SIP feeds at this time and are only available through proprietary exchange data products. Proof Trading does not purchase imbalance feeds currently, and hence we will keep them out of scope for this initial investigation. We intend to purchase some historical data of imbalance feeds in the future in order to assess how much additional predictive value they might represent for our developing use cases. To perform such an assessment, we will need to have a strong baseline for how well closing sizes can be predicted from our existing data, so the work here represents an important precursor, even for use cases where imbalance feeds would be actionable.

3 Choosing a prediction target and a metric of success

In many applications of supervised learning, this part of the process is fairly obvious. And at a first glance, it may seem so here. We know what we want to predict: the size of the closing auction. Since this is a numerical variable, there is a standard choice for a metric of success: mean-squared error.

If we have a sequence of close sizes $x_1, \ldots, x_n$ and a corresponding sequence of predictions $y_1, \ldots, y_n$, we can compute the mean-squared error as:

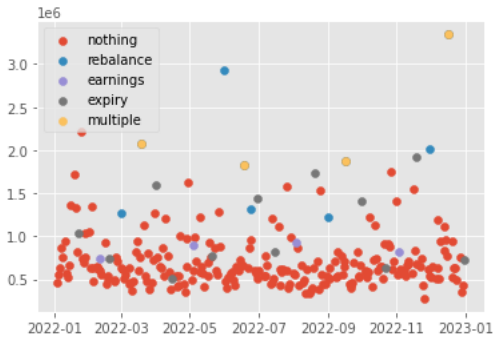
$$MSE := \frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n}.$$

This is a convenient metric for many reasons. The squaring is a clean way to cost errors of both under and over-shooting, and it produces a nice continuous, differentiable, and convex function that is relatively easy to minimize when it comes to fitting functional forms. The summation composes well over different kind of data points (in our case, different symbols for example), and the averaging can be easily tweaked to reflect weights if we care more about some data points than others:

$$MSE \text{ with weights } := \sum_{i=1}^{n}(x_i - y_i)^2 w_i,$$

where w_i values here represent the desired weights.

Despite all of these advantages, MSE is probably a poor choice of metric for close size prediction. The reason lies in the nature of the most extremal points. As we have seen in the plots above, close sizes tend to bounce around somewhat wildly, and the outliers can be extreme. We might hope to sift out some of these outliers due to our flagging of special event days, but it would be overly optimistic to assume that all outliers can be easily explained or anticipated. Here for example is a plot of the closing sizes for CVS over 2022:



If we use a metric like MSE to fit models on data like this, it will force the models to really try to predict just *how high* each abnormally high value is going to be. Failing to do so will incur very large penalties of $(x_i - y_i)^2$ for times when x_i is huge and y_i is middling. These big failures are amplified by squaring, and so will likely become the primary contributors to the overall mean squared error. But it is probably an incredibly difficult task to predict exactly how high a high volume close is going to be! So we would be setting our models up for failure - telling them that what's most important is actually the thing they are least likely to be able to do. As a result, we'd probably get a volatile mess of over-fitting, rather than making steady and robust improvements in prediction holistically.

We could try to mitigate this by using mean absolute error, which simply takes an absolute value instead of squaring. This probably doesn't go far enough though. A better choice for our scenario is something like mean absolute percentage error (MAPE). For a sequence of close sizes $x_1, \ldots, x_n$ and a corresponding sequence of predictions $y_1, \ldots, y_n$, the mean absolute percentage error is defined as:

$$MAPE := \frac{1}{n}\sum_{i=1}^{n}\frac{|x_i - y_i|}{x_i}.$$

With this metric, we'd be looking at error as a percentage of the true value x_i, a change which powerfully controls the influence of data points where x_i is large, since we are now dividing a potentially large error by a large denominator.

But a related problem still remains - what about outliers where x_i is unusually small? In fact, our data set contains 36 points where the close size was exactly 1 (mostly for BRKA, one for IMPP), and 455 points where the close size is less than 1000 (these occurred for 8 symbols). While we could investigate the nature of such low outliers and try to filter them out in some principled way, it's cleaner if we can simply choose a metric that doesn't place undue attention on trying to predict them. MAPE will blow up when x_i is suddenly small - a prediction of $y_i = 100$ for example when $x_i = 1$ will contribute a whopping

$\frac{|1-100|}{1} = 99$ to our error sum. This probably doesn't reflect our actual goals. After all, if the closing auction ends up being abnormally small, it's probably because we didn't manage to trade much in it. In this case, we may not care too much about our prediction being larger, because we didn't end up trading too much anyway.

We don't know of a standard metric that seamlessly corrects for this, so we propose one of our own that we will call MAREγ. It stands for mean absolute ratio error - with a parameter γ. We pronounce it as "Mary" because, well, statistics could use a few more things named after women. It is defined by the formula:

$$MARE\gamma := \frac{1}{n}\sum_{i=1}^{n}\frac{|x_i - y_i|}{x_i + \gamma y_i}.$$

Let's unpack the motivation and properties of this metric. First, what kind of real error does it capture? Suppose, for example, that once we have a prediction y_i for what we think the closing auction size will be without our participation, we decide to put in an auction order for γy_i shares. In other words, we are expecting to represent a $\frac{\gamma}{1+\gamma}$ fraction of the total auction size if our order gets fully filled. Thus, $\frac{|x_i-y_i|}{x_i+\gamma y_i}$ represents the absolute percentage error *under the assumption that we successfully add γy_i shares to the closing auction size.* In other words, we assume the true answer becomes $x_i + \gamma y_i$, and our prediction becomes $y_i + \gamma y_i$. [Note that the difference between these two things stays equal to $x_i - y_i$, as the γy_i terms added to both cancel out.] For this reason, we feel that the MAREγ metric has a defensible interpretation as being MAPE in a world where we assume trading success, which is probably the world we most care about anyway.

This extra γy_i term acts as a floor that prevents the denominator from being suddenly low, allowing our models some grace when they fail to anticipate low outliers. Similar to move away from mean-squared error, this kind of correction is likely to be necessary if low outliers are not particularly predictable. However, introducing this parameter γ presents a new problem - how should we choose γ? If γ is too small, the corrective goal here will not be accomplished. If γ is too large, our assumption that our trading activity will fully accomplished becomes less and less realistic. Theoretically, we could adjust γ surgically for every data point here, reflecting some notion of how much we may typically want to trade in the close for various symbols on various days. But for now, we will do something relatively simple and set $\gamma = 0.1$ universally. We will also later test a few nearby values of γ to check that our results are not too dependent on the exact value of γ.

4 Evaluating baseline models

With all of this in place, we can finally compute our chosen MAREγ metric on some baseline models to see what we should be aiming to beat. Like MSE or MAPE, the MAREγ metric can be augmented with arbitrary weights w_i if we want, but for now we'll just weight all symbol/day pairs evenly.

As a first test, let's compare using a 20-day rolling average of close sizes as a prediction to using a 20-day rolling median. Computing over our entire training data set of 2022, we get a MAREγ value of 0.672 for the rolling average, and a MAREγ value of 0.414 for the median. This is an immediate win for the median! [If it helps you to remember, error scores are like golf. Lower is always better.]

We might wonder how much of this is due to the fact that the special event days are still being included inside the rolling average calculations. To get a sense of this, we'll remove all of the special event days from our data set, recompute rolling averages and medians on the remaining data only, and compare everything on the now-pruned data set. The results are quite striking:

prediction method	**MAREγ score**
rolling average	0.689
rolling average of non-special days	0.569
rolling median	0.408
rolling median of non-special days	0.385

Table 1: Baseline prediction methods - performance on non-special days

There is some gain to just removing special days from the rolling computations, but the general robustness of medians here vs. averages is an even bigger effect. This is a strong signal that we should be using median-based prediction as our baseline to beat, as this is demonstrably better than average-based prediction.

5 Functional forms and (re)-choosing a prediction target

Now it's time to think about how we might incorporate our chosen features of rolling historical medians, $OpenF$, $LateF$, $IsExpiry$, $AfterEarnings$, and $IsRebalance$ into functions that produce predictions of closing sizes. We'll let $CloseH$ denote the rolling historical median of closing sizes, computed over a trailing window of 20 days. Our baseline to beat can now be expressed as a simple linear function $Y = CloseH$, where Y represents our prediction for the close size. [We could alternatively use medians computed with special days removed, but it's not clear that the small improvement we observed above on non-special days is worth the additional complication in implementation. For now we're trying to keep it simple and find big relative improvements, rather than small refinements.]

Even before we bring in the additional features, we should ask: Is $Y = CloseH$ the best prediction we can make for closing sizes using only the feature $CloseH$? What if we consider predictions of the form $Y = \alpha * CloseH$, where α is a constant? Are we sure that 1 is the best constant under our metric? It turns out that it's not!

Let's solve for the value of α that minimizes our MAREγ score on our training data for predictions of the form $Y = \alpha * CloseH$. Technically we will minimize n times the MAREγ score, essentially removing the averaging. This is equivalent to minimize when the number of data points n is fixed, and it can de-clutter our expressions a bit since we don't have to carry out factors of $\frac{1}{n}$. If we let x_i denote the true close sizes over our data set, we can write the total error sum as a function of α:

$$n * MARE\gamma(\alpha) := \sum_{i=1}^{n} \frac{|x_i - y_i|}{x_i + \gamma y_i} = \sum_{i=1}^{n} \frac{|x_i - \alpha * CloseH_i|}{x_i + \gamma * \alpha * CloseH_i}$$

Staring at this, we immediately feel the downside of moving away from mean-squared error as our metric: minimizing a nice quadratic function of α on our data set would be a breeze, whereas this looks like a mess! There are a few different approaches we can take to deal with this. The traditional mathematical approach would be to find all of the values where this function of α is either not differentiable or where the derivative is equal to 0. These would be the candidate values of α and then we would check them to find the minimum. A more heuristic and brute-force computational approach is to simply try all values of α at some fixed level of precision and in some fixed range. If we were planning to do this sort of minimization frequently on very large data sets, it could be worthwhile to develop the more principled mathematical approach, but since we are only doing this to get a sense of a good choice of α, and don't expect this to be our final model anyway, we'll take the heuristic computational path for now, trying all potential values of α between 0.50 and 1.50, rounded to two decimal places)

Computing across our whole data set but excluding special event days, we get a minimizing value of $\alpha = 0.82$. Using a prediction of $Y = 0.82 * CloseH$ on these days yields a MAREγ score of 0.37, compared to 0.40 for using $Y = CloseH$.

We might suspect that the MAREγ-minimizing value of α should be different for special event days. We can check this by computing minimizing values separately on the sub-data sets where $AfterEarnings = 1$, where $IsRebalance = 1$, and where $IsExpiry = 1$. For options expiry days, we find that $\alpha = 1$ is the best choice, which brings us back to $Y = CloseH$ as a baseline prediction for those days. But for earnings and rebalance days, we find higher values of α perform better. For after earnings days, we get a minimizing value of $\alpha = 1.18$. For rebalance days, we have to raise the ceiling of our search a bit and end up with a winning value of $\alpha = 1.95$.

Here is a summary of our results for using a multiple of $CloseH$ to predict close size, using the MAREγ scoring metric and dividing our data by special events:

data set	**minimizing** α	**score for** $Y = \alpha * CloseH$	**score for** $Y = CloseH$
non-special days	0.82	0.37	0.40
after earnings days	1.18	0.26	0.29
expiry days	1	0.48	0.48
rebalance days	1.95	0.61	0.68

Table 2: Using a function of the form $\alpha * CloseH$ for prediction

These results represent a new baseline for more complicated models to beat.

5.1 Linear functions using OpenF and/or LateF

A natural thing to try next is linear functions over more of our variables. Focusing on non-special days for now, we might want to get $OpenF$ and $LateF$ into the mix to see how much we can leverage their correlation with $CloseF$ to improve our predictions. Since $OpenF$ and $LateF$ are ratios, it wouldn't really make sense to use a functional form like $Y = \alpha * CloseH + \beta * OpenF + \delta * LateF$. Instead, we might imagine:

$$Y = \alpha * CloseH + \beta * OpenF * CloseH + \delta * LateF * CloseH.$$

If we use out-of-the-box linear regression in the scikitlearn package in python to fit a linear form like this to our non-special training data, we get $\alpha = 0.87$, $\beta = 0.15$, and $\delta = 0$ (each rounded to two decimal places). If we use these values to make predictions for the close size (still on the training data set), we get an overall MAREγ score of ... drum roll please ... 0.41. [Sad trombone sound.]

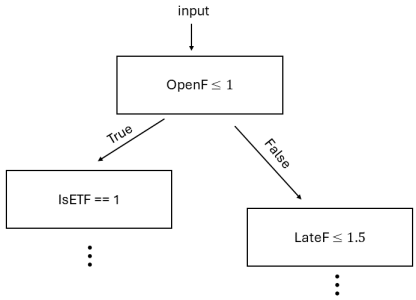
This is a *worse* score than we were getting for just using $\alpha CloseH$! Since linear regression solves for an optimal solution on the training data, and setting $\beta = \delta = 0$ is an option it should be considering, this may seem nonsensical at first. In fact, it would be nonsensical, except for the key fact that the out-of-the-box package is optimizing *with respect to mean-squared error*, and we are measuring predictive success in a different metric.

Before we spend effort to either find or build a more sophisticated implementation of linear regression that can minimize with respect to our custom MAREγ metric, we should try to get a quick sense of whether linear models are likely to perform well on our data. One brute-force way to do this is a to do a grid search over some range of coefficient values and pick out the combination that minimizes our error metric (among the values we tried). For a quick and dirty version, we'll do this just for two coefficients at a time. We'll start with a grid search over α and β values to find the best fitting function of the form $Y = \alpha * CloseH + \beta * OpenF * CloseH$. If we search a range of 0 to 1 for each coefficient, rounding to one decimal point, we come up with values of $\alpha = 0.7$ and $\beta = 0.1$ The overall MAREγ score still rounds to 0.37, so is not noticeably better than what we had without using the $OpenF$ feature. Similarly, we can try $Y = \alpha * CloseH + \delta * LateF * CloseH$, and the best combination we find is $\alpha = 0.6$ and $\delta = 0.2$. This yields a score of about 0.36, which is an underwhelming improvement.

5.2 Decision tree models and a shift of perspective

When linear models don't show meaningful improvements when we add features, there are several possible explanations. One is that the features simply aren't meaningful, or are redundant to features that are already present. Another is that the underlying relationships aren't linear, and our choice of functional form is holding us back. Since we are still considering such a small feature set here, and the Spearmen correlation coefficients for both $OpenF$ and $LateF$ were clearly meaningful, there are good reasons to suspect that the presumption of linearity is a large part of the problem.

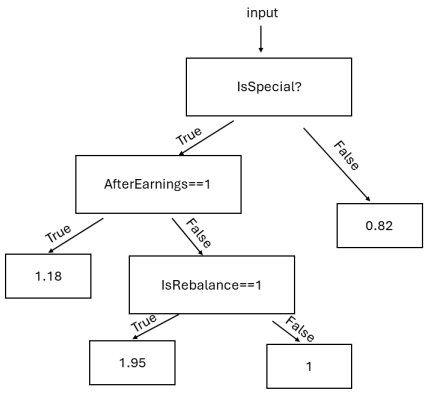
Decision trees are a class of non-linear functions that may work well when linear models fail. They allow us to partition our data into pieces through a sequence of decisions, where each decision is determined by checking on our variables against a threshold. For example, we might have a tree whose top node splits our data in two by whether $OpenF$ is ≤ 1 or > 1. From there, we might split each subset of our data further according to similar criterion:



Ultimately, each leaf of a decision tree is labeled by a single prediction. The tree represents a function from our input feature values to our resulting predictions. The function is evaluated by taking a single data point and following the rules of the tree nodes down from the root until arriving at a leaf and assigning its value as the prediction. In this way, decision trees represent piecewise constant functions on our data set, where the boundaries of the pieces are defined by a combination of thresholds on our input features.

At a first glance, these may seem like a terrible fit for our problem of closing size prediction. Close sizes are nowhere near constant! Since the number of leaves is a decision tree is a bound on the number of distinct values it will predict, we'd expect to need ridiculously big trees in order to produce good closing size predictions, and such large trees are definitely going to be over-fit.

But actually - there is a key shift in our perspective that is useful here. What if instead of predicting close sizes directly, we use decision trees to predict $CloseF := CloseSize/CloseH$? If we view $CloseF$ as the prediction target instead of the close size, then our current progress as summarized in Table 5 can be viewed as a simple decision tree with four leaves:



We can even re-imagine our MAREγ metric for close size predictions as a metric on $CloseF$ predictions! To see this, let's go back to the formula for computing MAREγ:

$$MARE\gamma := \frac{1}{n}\sum_{i=1}^{n}\frac{|x_i - y_i|}{x_i + \gamma y_i}.$$

In this formula, x_i represents the true close size, and y_i represents our prediction of close size. Let's introduce a new variable z_i to represent the value of $CloseH$ for data point i. Our metric doesn't change if we divide by z_i in both the numerator and denominator of the i^{th} term:

$$MARE\gamma := \frac{1}{n}\sum_{i=1}^{n}\frac{|\frac{x_i}{z_i} - \frac{y_i}{z_i}|}{\frac{x_i}{z_i} + \gamma\frac{y_i}{z_i}}$$

[Technical note: we are assuming here that z_i is never 0 so that it makes sense to divide by it. We suspect dropping any points where $z_i = 0$ would not be a big deal as this shouldn't happen often.]

If we define $\tilde{x}_i := \frac{x_i}{z_i}$ and $\tilde{y}_i := \frac{y_i}{z_i}$, we can rewrite this as:

$$MARE\gamma := \frac{1}{n}\sum_{i=1}^{n}\frac{|\tilde{x}_i - \tilde{y}_i|}{\tilde{x}_i + \gamma\tilde{y}_i}.$$

Thus, viewing $\tilde{x}_i$ as the true value of the prediction target $CloseF$ and $\tilde{y}_i$ as our prediction for $CloseF$, we get exactly the same computation under the MAREγ metric as we had before.

There is a remaining problem though. Common off-the-shelf decision tree fitting packages, like the one in scikit learn, will try to find trees that perform well on the default error metric of mean-squared error. As we saw with linear models above, it's going to be important to try to fit models based on our desired error metric instead. Unfortunately, the scikit learn decision tree package doesn't let us specify our own arbitrary metric, and it's not hard to understand why, as supporting arbitrary metrics would likely make the package prohibitively inefficient.

In this case, however, we can use a somewhat slick trick to avoid having to build own our package around our novel metric choice. The scikit learn package does support another option for the metric

used to construct a good tree: mean absolute error (MAE). Trying to minimize MAE for predictions of $CloseF$ would mean finding trees that have smaller values for:

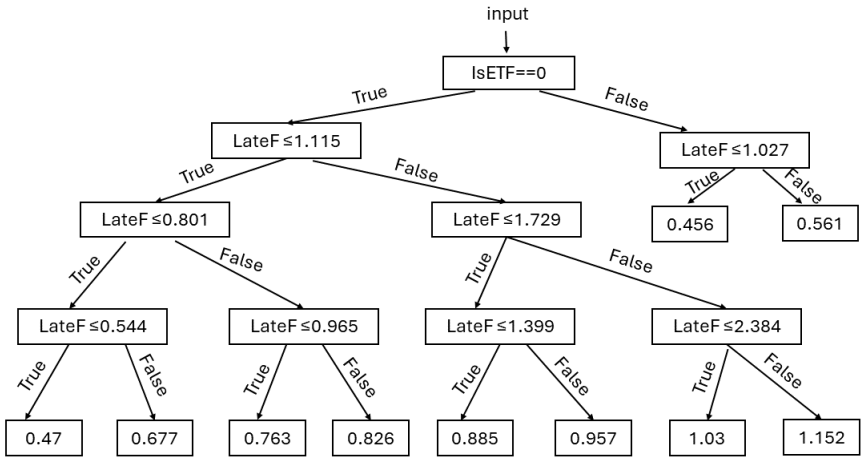
$$MAE := \frac{1}{n}\sum_{i=1}^{n}|\tilde{x}_i - \tilde{y}_i|.$$

This has the same numerator as our desired MAREγ metric, but it is missing the denominator. However, the scikit learn package has one more built-in degree of freedom. We can specify weights w_i for the data points and instead ask it to try to minimize:

$$\frac{1}{n}\sum_{i=1}^{n} w_i|\tilde{x}_i - \tilde{y}_i|.$$

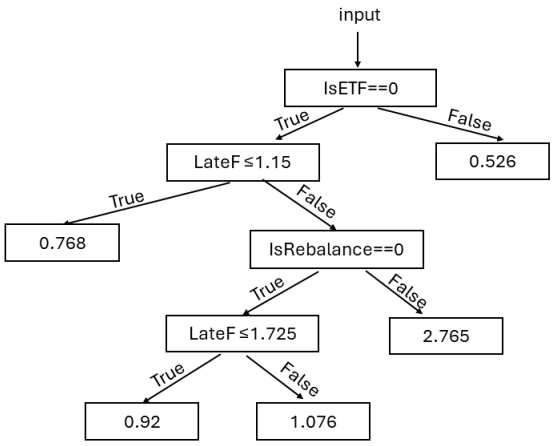
If we could set $w_i = \frac{1}{\tilde{x}_i+\gamma\tilde{y}_i}$, this would be exactly what we want! But we can't do that, as the predictions $\tilde{y}_i$ are not fixed ahead of time, but rather what we are trying to determine through the tree-fitting process. We can reasonably approximate what we want, however, by setting $w_i = \frac{1}{\tilde{x}_i+\gamma}$, which is what the denominator would be in MAREγ for the default case of predicting $CloseSize = CloseH$, or equivalently, $CloseF = 1$.

Using the pre-built functionality of weighted MAE minimization with these weights, we are able to obtain rather simple trees that perform well under MAREγ. Looking at only non-special days for example, here is a 10 leaf tree that just uses the features $IsETF$ and $LateF$ to obtain an overall MAREγ score of 0.34 on the training data, which is a promising improvement over the 0.37 we obtained in Table 5:



We asked the scikit learn package to fit this tree using three variables: $OpenF$ which would appear in the tree as $x[0]$, $LateF$ which appears as $x[1]$, and $IsETF$ which appears as $x[2]$. It is encouraging to see the tree make some intuitive choices, like splitting based on $IsETF$ right at the top, allowing it to create two submodels: one for ETFs and one for non-ETFs. It is also interesting that the $OpenF$ feature wasn't used - this seems to be a general tendency to use $LateF$ more than $OpenF$ in the constructed trees of various sizes, so we suspect that $LateF$ is the more useful feature.

We can also throw in all the data along with the additional variables $IsExpiry$, $IsRebalance$, and $AfterEarnings$ to let the tree fitting decide for itself when and how to adjust for special events. Here for example is a 5 node tree fitted over the entire training set:



The variable $x[1]$ here is $LateF$, the variable $x[2]$ is $IsETF$, and the variable $x[3]$ is $IsRebalance$. The overall MAREγ score on the full data set for this tree is 0.35. This compares favorably to an overall score of 0.38 if we aggregate the results in Table 5 (weighting each individual data point evenly).

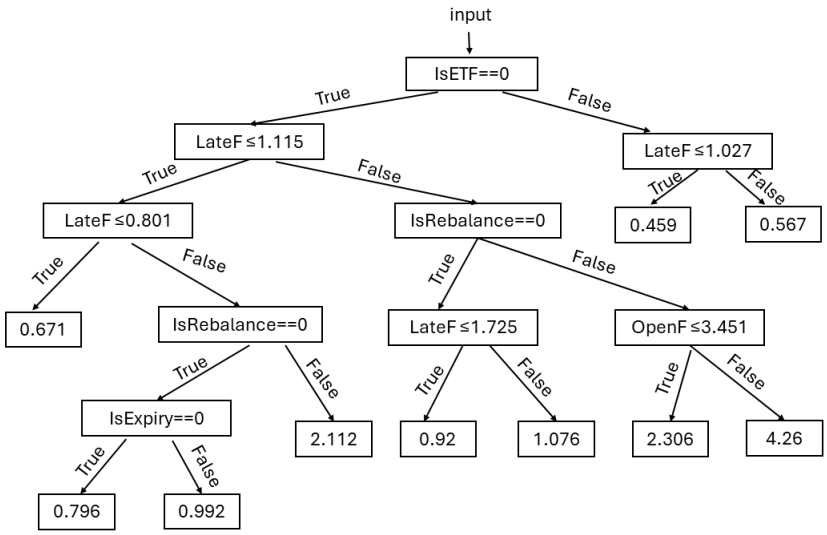
Naturally, adding more nodes to our trees can give us better performance on our training set, but there are a few downsides. For one, the models become less explainable from a human perspective. They also may start to suffer from over-fitting, meaning that they begin making decisions that depend too heavily on the granular details of the training data in ways that won't generalize well to making good predictions on fresh data. For this reason, we probably want to stop adding nodes to our model as soon as the performance benefits begin to dwindle. Let's look at MAREγ scores on the full training data set for different tree sizes in order to help us pick a size. [Here's we'll report scores to greater precision to help us see the dwindling performance effect.]

Number of leaves	**Variables used**	**MAREγ score**
5	$LateF$, $IsETF$, $IsRebalance$	0.3503
10	$OpenF$, $LateF$, $IsETF$, $IsRebalance$, $IsExpiry$	0.3441
20	$OpenF$, $LateF$, $IsETF$, $IsRebalance$, $IsExpiry$	0.3411
30	$OpenF$, $LateF$, $IsETF$, $IsRebalance$, $IsExpiry$	0.3401

Table 3: Comparing different tree sizes on the training set

It is perhaps interesting here that all trees were fit with all of the variables $OpenF$, $LateF$, $IsETF$, $IsRebalance$, $IsExpiry$, and $AfterEarnings$ as options, and $AfterEarnings$ was never used. It is also interesting that the performance improvements accrue and then dwindle fairly quickly, meaning that small trees seem to be sufficient to capture what we can with this approach.

Let's take a look at the fitted tree with 10 leaves:



To help us understand the function that this tree represents, we can order its leafs from the smallest output to the largest and list the ranges of variables that each leaf represents:

Output	**IsETF**	**OpenF**	**LateF**	**IsRebalance**	**IsExpiry**
0.459	True	Any	≤ 1.027	Any	Any
0.567	True	Any	≤ 1.027	Any	Any
0.671	False	Any	≤ 0.801	Any	Any
0.796	False	Any	$(0.801, 1.15]$	False	False
0.92	False	Any	$(1.15, 1.725]$	False	Any
0.992	False	Any	$(0.801, 1.15]$	False	True
1.076	False	Any	> 1.725	False	Any
2.112	False	Any	$(0.801, 1.15]$	True	Any
2.306	False	≤ 3.451	> 1.15	True	Any
4.26	False	> 3.451	> 1.15	True	Any

Table 4: Output table for our 10-leaf tree

There are many things that are intuitive about this model, and a few things that inspire questions for further investigation. On the intuitive side, we see that outputs tend to be lower for ETFs than non-ETFs, lower for lower $LateF$ values and higher for higher $LateF$ values, higher for special days than non-special days, etc. We might ask though: WTF is up for ETFs? Why is it the case that we end up multiplying their historical medians by such low values to obtain our predictions? What does this say about the distributions of their closing auction sizes?

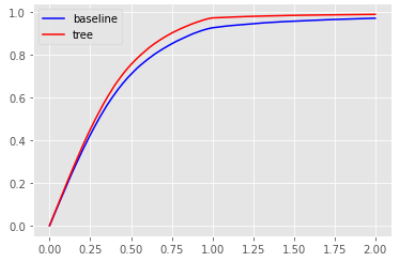
Testing on fresh 2023 data Let's take our 10-tree model and see how it performs at making predictions for 2023 data (collected for the same 1000 symbols). As a baseline, we'll first compute the MAREγ score for the baseline strategy of predicting the historical median every time. This score is 0.402 on our test data, which is extremely similar to the analogous score of 0.410 on our training data. When we use our 10-leaf tree to make predictions on the test data, the score drops to 0.337. This is comforting for two reasons. First, it is very similar to the performance benefit on the training data, suggesting that we are not suffering from overfitting. Second, it is good evidence that the kind of model we are fitting has relevance over fairly long stretches of time.

Let's also take a look at the error distribution for our predictions on test data. For this, we'll avoid having to set the γ parameter for the MAREγ metric and will instead just compute the absolute percentage error for each prediction. In other words, for each day and symbol we'll compute:

$$APE := \frac{|x - y|}{x},$$

where x is the true close size and y is our prediction for it (using the 10-leaf tree). We can then ask questions like: what fraction of the time do we achieve APE values below a certain bound? To visualize the answers to such questions, we can graph the cumulative density function for our APE values. More precisely, for an error bound b, we'll define the function $cdf(b)$ to be the fraction of predictions whose APEs are $\leq b$, and then we'll plot $cdf(b)$ as a function of b.

We'll do this for both our 10-leaf tree predictions and the baseline of using historical medians as predictions:



The fact that the red curve here stays to the left of the blue one is encouraging: it means that we are generally able to keep a higher fraction of predictions within a given error bound using our decision tree method, as compared to the baseline.

6 Ideas for further research

There are several things we plan to investigate next in order to further improve our prediction capabilities.

Anomaly detection and bootstrapping We might wonder if there are detectable patterns in cases where our predictions are most wrong. For example, days where there is significant news related to a symbol might result in trading activity that we can spot as anomalous earlier in the day. Even if it is difficult to obtain *good* predictions for anomalous events, it would be helpful to know that we should have less confidence in predictions in such circumstances.

To reduce more mundane errors, we can also try boosting. This is a common practice in machine learning that involves fitting a new model specifically to predict the errors in an existing model, and then forming a combination of the two models to perform better than the existing model alone.

Symbol clustering, panel data analysis, and other functional forms Right now we are fitting one static model shape (the decision tree) across symbols, using symbol and day specific features as inputs. However, there are a number of more interesting ways to fit shapes that can wholly or partially depend on the symbol and/or time. We could train a fresh model per symbol, for example, though we suspect that this might be slicing the data too thin and result in some model instability and over-fitting. We could alternatively cluster symbols according to some features of trading activity and then train a model shape for each cluster, allowing us some variety of shapes while still keeping a sizeable amount of data for training for each cluster. We can also explore the performance of different re-training frequencies to learn something about how quickly our model shapes should be adapting over time.

Techniques of panel data analysis offer potentially different ways to express the joint influence of symbols and time in our modeling process. We can try fixed-effects models and generalized linear models (GLMs), for example, which can combine symbol-specific effects and time-varying factors.

We can also extend the functional forms our model-fitting process considers in rather natural ways. Random forests are an obvious next thing to try, as they are fit by combining a number of decision trees. Feed-forward neural nets are also a natural extension to try, as they are similar to trees except with an extra degree of freedom that a child node can have multiple parent nodes (unlike a tree, where a child has a single parent). We should suspect though, that these extra degrees of freedom will introduce problems of over-fitting rather quickly, so we will need to be disciplined and cautious in our testing processes to validate such models and their robustness. This is why we do not (as a general practice) jump to using fancier models from the outset. It is important to first reach an improved robust baseline like we have developed here to judge such models against, so we can more clearly see if the added complexity is really worth it.

Feature refinement and enhancement The feature set we have explored so far is fairly intuitive and basic, and we suspect there is a lot of room for refinement and enhancement here. Refinement could include tweaking the definitions of the existing variables, especially $LateF$. We might wonder, for example, how important it is what precise time of day is used for measuring $LateF$, and if it is best measured across the market as we are doing now, or only on the primary exchange, or excluding certain kinds of volume, etc. We might further refine features like $IsETF$ to consider different kinds or clusters of ETFs.

There are also new features we might add to the mix. One could be a symbol-specific feature that encodes which indices a symbol belongs to - something that might be relevant in combination with a feature like $IsRebalance$. Other possible features might include more nuanced properties of trading activity, like how closely trades are clustered together, median trade sizes, etc.