The Evolution of the Crumbling Quote Signal

Allison Bishop

Abstract

In this paper, we illuminate the origins of the crumbling quote signal employed by IEX, provide detailed analyses of its performance, and discuss its continuing evolution.

1 Introduction

In the decentralized US equities market, a change in the National Best Bid and Offer (NBBO) is not an instantaneous event. Quotes in a stock move asynchronously across venues, and these movements are observed at different times by different participants. This affects the experience of pegged orders, which are intended to follow the NBBO as it moves. Midpoint pegged orders, for example, are intended to trade midway between the current national best bid and national best offer, and these orders can be an effective way for institutional investors to obtain price improvement and experience high execution quality. However, the intent of pegged orders may be systematically violated in moments where the NBBO is in transition, as the venue maintaining the relationship between the price of the pegged order on its book and the NBBO may not be the first to observe a change in the NBBO, and hence the order may be temporarily exposed to unfavorable pricing.

Though they may be extremely short-lived, windows of time in which some market participants know information that others have yet to receive are ripe for exploitation. One instantiation of this is *stale quote arbitrage*. This phenomenon arises out of market fragmentation: since one trading venue cannot instantaneously know what is happening on another trading venue, it is possible for a trader to temporarily have an informational edge over a trading venue and manipulate that edge to the detriment of other traders. This is perhaps best explained through an example: suppose that trader A has rested a midpoint-pegged order to buy a particular stock on venue V. The purpose of such an order is to float with the midpoint of the NBBO across all of the exchanges, but even this basic statement is a bit misleading. There is no such thing as "the NBBO" in a technical sense. The order is actually pegged to the midpoint of the *the view of the NBBO as observed by trading venue V*. Suppose at a given point in time, the market has been stable for awhile (meaning the best bid and offer prices have not

changed for several milliseconds), and trading venue V has an accurate, up-to-date view of the NBBO in that symbol as $10.00 by $10.02. So the resting buy order is pegged to the midpoint price of $10.01. Now suppose that the best bid of $10.00 is not available on trading venue V, but instead is only available on trading venue W. Now, a seller comes along and trades with all of the buy interest at $10.00, changing the best available bid to $9.99. The midpoint of the NBBO is now $10.005, but this information does not arrive at trading venue V instantaneously. There is a small window of time in which venue V still believes the midpoint is $10.01, so if a matching sell order arrives at venue V during this window, it can trade with the resting buy order at $10.01. This is bad news for the initiator of the resting order, because the NBBO has already changed in their favor, and this execution at $10.01 goes against the spirit of what a midpoint pegged order is intended to accomplish.

We might ask: who is this trader swooping in to take advantage of the resting order on venue V at this stale midpoint price? It could be a large seller who has himself taken out all of the buy interest at $10.00 on venue W, in which case it is no surprise that this seller can anticipate the change in the NBBO. There is not really anything to do about this case, this is just something that happens as part of normal, healthy market behavior. But there is another possibility: perhaps an opportunistic trader observed the NBBO change before venue V and managed to submit a sell order to venue V that arrived ahead of the new price information. This is possible if the method of communication employed by this trader is faster than the method of communication between the trading venues. This is what we call stale quote arbitrage, and we believe that its presence is undesirable in the marketplace.

To prevent this scenario from occurring, IEX created the "speed bump," a mechanism for delaying inbound orders by 350 microseconds. Under normal circumstances, it takes roughly 300 microseconds for IEX to learn about a price change on another market and update our pegged orders accordingly, so 350 microseconds is a sufficient buffer to ensure that even if a trader instantly learns of a price change anywhere in the market and immediately submits an order to IEX, all of the pegged orders on IEX will already be updated by the time the incoming order is processed.

And yet, even with this speed bump in place, we began to see an increase in adverse selection experienced by resting orders on IEX. Trades were often executing at prices just before IEX observed a change in the NBBO that would have been favorable to the pegged resting order. But how could this happen? The speed bump was still doing its job - it was preventing someone from quickly reacting to an NBBO change and picking off a resting order on IEX, but it was not preventing someone from *anticipating* an NBBO change. We hypothesized that traders were building probabilistic models to predict price changes far enough in advance to circumvent the protection of the

speed bump. Naturally their predictions would not be perfectly accurate, but they could likely gain an edge by predicting some price changes say 1 or 2 ms before the actual change solidified. This is made possible by the fact that changes in the NBBO are not atomic events, but are often the cumulative result of a sequence of events at different trading venues that span a small but significant window of time, typically on the order of a millisecond.

If we return to our example of a stable NBBO that is currently $10.00 by $10.02, we might imagine that the buy interest at $10.00 is spread over several trading venues. A seller or sellers may arrive at those venues one by one, first exhausting the buy interest at $10.00 in one place and then the next. We call this situation a "crumbling quote" - it doesn't change all at once, it changes gradually. Nimble observers who are watching these dominos fall need not wait for the last order at $10.00 in the last venue to be exhausted. Instead, they can make a reasonable guess earlier in the process as to what the short term outcome is likely to be. This may allow them to exploit resting orders probabilistically, even with a speed bump in place.

To combat this situation, a natural impulse is to increase the length of the speed bump to cover the typical length of a quote change process across venues. This would require a very large increase: perhaps from 350 microseconds to a few milliseconds. Such a large speed bump would be a more difficult pill to swallow, as it is of a higher magnitude than the inadvertent delays on order receipt and processing that inevitably result from communicating across significant physical distances and in an environment with high computational demands. It would also be a rather blunt and inflexible instrument to address what is a delicate and constantly evolving problem.

Our approach instead is to shape the solution to match the problem: we can fight math with math! If others are leveraging short term prediction models to anticipate NBBO changes, than we can build such a model ourselves and deploy it to protect resting orders. One might worry that this will devolve into an arms race, with traders furiously constructing better models, and IEX similarly racing to improve its own prediction until mere microseconds and nearly imperceptible statistical edges separate the top competing models. But there are reasons not to expect this. Traders seeking structural arbitrage opportunities and IEX as an exchange are not symmetrically positioned in this contest. With the speed bump in place, even if the IEX model makes a prediction say 100 microseconds later than a fast trader, this prediction combined with the speed bump is sufficient to protect a resting order on IEX. Secondly, the cost incurred by the arbitrage-seeking trader when they make a trade under a wrong prediction of their model may be far greater than the opportunity cost incurred by a resting order on IEX when it delays execution for a short window due to a prediction by IEX's model. This may allow IEX to make predictions more aggressively than traders, and hence provide greater protection. Thirdly, it is not really necessary to stamp

out every last instance of structural arbitrage. Making it more rare and less profitable on our marketplace will provide a strong disincentive that should improve our market quality even though our model may be imperfect.

This paper documents the development of the IEX crumbling quote model, henceforth called "the signal," a method for detecting moments when a NBBO change is likely imminent. We detail the origins of the signal, its behavior on our market, and our continual efforts to improve it.

2 Developing a Model of a Crumbling Quote

As a consumer of market data, IEX maintains a view of the NBBO. More precisely, IEX observes a series of events, where each event is an update to the best bid and/or offer in a particular symbol on a particular venue. These events are time-stamped according to when they are received by IEX.

We can calculate many relevant features from this data. These features will look more granularly at changes to the National Best Bid (NBB) and National Best Offer (NBO) that together comprise the NBBO. For instance, we can determine how many venues are on the NBB and NBO at any given moment in time (from the viewpoint of IEX). Each time a new market data update is received, we can evaluate the current state of the NBBO and compare it to past states in order to form an opinion about whether a quote is likely to be crumbling. Intuitively, if many venues are deserting the NBB, we may expect that a downward price change is imminent. Conversely, if we are observing many venues deserting the NBO, we may expect that an upward change is imminent.

To make things more rigorous, let's first define some variables that we will use to make our prediction. We define *bids* to be the number of exchanges currently displaying a quote at the NBB, excluding IEX. This will typically be a number between 1 and 12. (Note: this model was originally developed before IEX became an exchange, so we will leave IEX out at this point for historical accuracy. We will also ignore the rare cases where *bids* may be 0 due to there being no bids anywhere, etc.) We similarly define *asks* to be the number of exchanges currently displaying a quote at the NBO (also a number typically between 1 and 12). We define $bids5$ to be the number of exchanges who were displaying a quote at the NBB 5 ms ago, and $asks5$ to be the number of exchanges who were displaying a quote at the NBO 5 ms ago.

A simplest first attempt at a crumbling quote signal might go something like this:

$$\text{If } bids5 - bids \geq 3, \text{ predict downward price tick.}$$

$$\text{If } asks5 - asks \geq 3, \text{ predict upward price tick.}$$

This attempts to capture the intuition that if enough venues are backing off of one side of the NBBO within a short amount of time, this may

indicate the remaining ones are soon to follow. But this first try is a bit too simplistic. When there has already been a price change within the last 5 milliseconds, the variables $bids$, $asks$, $bids5$, $asks5$ are still well-defined, but comparing $bids$ and $bids5$ for example does not really make sense. This comparison is only apples-to-apples if the price level has remained constant over the preceding 5 milliseconds. This suggests a refinement of our basic rule, namely that it should be applied only when the price has not moved in the last 5 milliseconds.

There are a few other reasonable restrictions we might impose on our predictions. We might require $bids < asks$, for example, in order to predict a downward price tick and analogously require $asks < bids$ in order to predict an upward price tick. We might require that the current spread is not larger than the average spread (in that particular symbol). In fact, we will initially be even more restrictive and require that the current spread is one cent, and the bid price is at least one dollar. To express this slightly more refined definition of a crumbling quote signal, we will introduce a few more variables. We let bpx and apx denote the current NBB and NBO prices, respectively. We let $bpx5$ and $apx5$ denote the NBB and NBO prices as of 5 milliseconds ago. We note that the current spread can be then expressed as $apx - bpx$. We may then define a potential signal as:

$$\text{If } bpx = bpx5, apx = apx5, bpx - apx = \$0.01, bids < asks, bpx \geq \$1,$$

$$\text{and } bids5 - bids \geq 3, \text{ predict downward price tick.}$$

$$\text{If } bpx = bpx5, apx = apx5, bpx - apx = \$0.01, asks < bids, bpx \geq \$1,$$

$$\text{and } asks5 - asks \geq 3, \text{ predict upward price tick.}$$

The careful reader may notice a slight ambiguity here. When we write a condition like $bpx = bpx5$, do we intend to merely check that the bid price 5 milliseconds ago was the same as it is now, or to enforce the stronger condition that the price has *remained* the same throughout the preceding 5 milliseconds? The answer does not matter all that much. The number of times where $bpx = bpx5$, for example, but the bid price has actually shifted and then shifted back within the previous 5 milliseconds is relatively small, so our choice on this minor implementation detail does not cause a significant effect.

The careful reader may still protest, however, that "predict downward or upward price tick" is not a precise specification. If we suspect a tick is imminent, when exactly do we expect it to be? Naturally, if we wait long enough, some tick will eventually occur, and we could declare a rather meaningless victory. As an initial setting to keep us honest, let us consider a 10 millisecond interval after each prediction. When a tick is predicted, we will say the signal has "fired". For this "firing window" of 10 ms following the prediction, we will say the signal is "on." This is a long enough time

window that if the quote is legitimately in transition, the transition should complete within this window. However, it is not too long, so when we are incorrect and make a false prediction, the wrong prediction is only in force for a relatively short span of time.

As an initial benchmark, let's test the potential of this simple proposal on a day's worth of data. We will use market data over the course of Dec. 15, 2016 for this. (A quick technical aside: we will often return to the same testing day throughout this paper as an example, so that we can make clear apples-to-apples comparisons. Of course, it is not good statistical practice to test things on only one day, and while performing this research we tested everything on many different days to check that the behavior we were seeing was indeed representative. Also, keep in mind that many of these test simulations do not correspond to what actually happened on IEX on Dec. 15, 2016, since there was a particular model used in production on that day and we will simulate the results of many different possible models throughout this paper.) On that day, our simple signal would produce about 570,000 correct predictions, which we'll call true positives, and 860,000 incorrect predictions, which we will call false positives. This actually isn't too bad: we can correctly predict about 570,000 crumbling quotes, and the total time wasted in false positive prediction intervals is only $10 * 860,000 = 8,600,000$ milliseconds, which is about 8,600 seconds, or roughly 143 minutes. Note that this is aggregated over all 8,357 symbols for the day, so the average time spent in false positive intervals per symbol would be roughly 1 second. This was actually the original version of the crumbling quote signal that accompanied the initial deployment of the discretionary peg order type on IEX in November 2014. It was in production until September 2015.

It turns out that looking back 5 milliseconds is longer than we really need. Instead, let's define the variables $bids1$ and $asks1$ to be analogous to $bids5$ and $asks5$, except only looking back 1 millisecond into the past instead of 5. In other words, we define $bids1$ to be the number of exchanges who were displaying a quote at the NBB 1 ms ago, and $asks1$ to be the number of exchanges who were displaying a quote at the NBO 1 ms ago. We similarly define $bpx1$ and $apx1$ to be the prices 1 ms ago. It also turns out that the restriction $bpx \geq \$1$ does not have a large impact. If we rerun our simulation on data from Dec. 15, 2016 using the same criterion as above except replacing all of $bpx5, apx5, bids5, asks5$ with $bpx1, apx1, bids1, asks1$ and removing the restriction that $bpx \geq \$1$, we get roughly 540,000 true positives and 711,000 false positives. This seems like a decent tradeoff: we see a comparatively small decrease in true positives in return for a much larger decrease in false positives. Going forward, we will stick with looking only 1 millisecond into the past, as this seems to perform better than 5 milliseconds and the conditions $apx = apx1$ and $bpx = bpx1$ are less restrictive than their counterparts $apx = apx5$ and $bpx = bpx5$.

To see how we can improve upon this, we'll start by trying logistic re-

gression over these features of $bids, asks, bids1, asks1$. We will walk through the rationale for this step by step. First, we suspect that our choice of criterion $bids1 - bids \geq 3$ (or $asks1 - asks \geq 3$ respectively) may not be an optimal choice. We ask, is there some other function of the features $bids, asks, bids1, asks1$ that is a better choice? A natural place to look is linear functions. For example, we could consider a weighted average of the values $bids, asks, bids1, asks1$ such as $0.4bids - 0.4bids1 - 0.1asks + 0.1asks1$, in deciding whether or not to predict a downward price tick. But how should we interpret the number that results from such a calculation, and how should we choose the coefficients?

Linear functions model continuous, real number outputs, and the thing we are trying to predict is a binary outcome: either a tick is imminent or it is not. Logistic regression is a general method for making binary predictions out of a linear combination of numerical features. The core idea is to model the probability of the outcome rather than the outcome itself. But this still leaves a slight disconnect: a probability is always between 0 and 1, while a linear combination of numerical features can take on values outside of this range.

For this reason, we would like to use a function that maps an arbitrary real number to a value that is guaranteed to be in the range from 0 to 1, so that we can always interpret it as a probability. In mathematical terminology, this is to say that the domain of the function will be $\mathbb{R}$, the set of all real numbers, while the range of the function will be $[0, 1]$, the set of real numbers between 0 and 1. There are infinitely many choices of functions that satisfy this, but logistic regression happens to choose

$$f(x) = \frac{1}{1 + e^{-x}}.$$

This choice of function has several nice properties. It varies smoothly between 0 and 1 as x varies between $-\infty$ and ∞. In this way, larger values of x correspond to higher probabilities, while lower values of x correspond to lower probabilities.

In our case, the real value x will be computed as a linear function of our variables, $bids, asks, bids1, asks1$. This means we want to choose real number coefficients c_0, c_1, c_2, c_3, c_4 and compute x as $c_0 + c_1 bids + c_2 asks + c_3 bids1 + c_4 asks1$. Plugging this into function f to transform it to a probability, we can write down our model as:

$$p := \frac{1}{1 + e^{-(c_0 + c_1 bids + c_2 asks + c_3 bids1 + c_4 asks1)}}$$

where p is our estimate of the probability of a downward price movement.

There is a symmetry in our problem that is worth noting: the role of $bids, bids1$ in predicting a downward price tick is analogous to the role of $asks, asks1$ in predicting an upward price tick. So we expect that our same

coefficients $c_0, \ldots, c_4$ can be used to predict an upward price tick, just with $bids$ and $asks$ interchanged, and $bids1$ and $asks1$ interchanged. Experiments on our historical data corroborate this intuition. For simplicity, we will always describe our formulas in the form for predicting downward ticks, but the upward prediction formulas can be inferred analogously.

What remains is to make good choices for the coefficients $c_0, \ldots, c_4$. This process starts by forming a definition of "good." Given a candidate setting of values for $c_0, \ldots, c_4$, we can compare its probability predictions to ground truth on a training data set pulled from our historical data. More precisely, we can view each time we received an update to NBBO as a fresh opportunity to make a prediction, and we also know whether a tick occurred in the following 10 ms. This means we can measure the magnitude of the errors produced by our predictions if we were to use these candidate values of $c_0, \ldots, c_4$ on this data set. To be conservative, we pruned our data set to include only moments when $apx = apx1$, $bpx = bpx1$, and the current spread is less than or equal to the average spread in that symbol (calculated in our system as a moving 30-day average). A typical software package with logistic regression built in (such as R) can find coefficients that minimize a particular measure of error on our data set. Employing this, we arrived at the following coefficients:

$$c_0 = -2.39515, \;\; c_1 = -0.76504, \;\; c_2 = 0.07599, \;\; c_3 = 0.38374, \;\; c_4 = 0.14466.$$

These numbers themselves may seem a bit mysterious, but they do have some features that are intuitive. The fact that c_1 is negative for instance, means that all other factors being equal, we are more likely to predict a downward tick when $bids$ is smaller. It is also intuitive that the far side variables ask and $ask1$ have coefficients that are smaller in magnitude than the near side variables $bids$ and $bids1$, meaning they have less impact on the prediction.

Now that we have a probability estimate, we need to turn it into a binary decision: do we fire the signal or not? We impose some restrictions in line with our choice of training data: we only fire to predict a downward tick, for example, where $bpx = bpx1$, $apx = apx1$, the current spread is less than or equal to the average spread in that symbol, and $bids < asks$. To decide whether or not to fire when all of these conditions are satisfied, we will set a threshold. If the probability estimate is above this threshold, we will fire. If it is below this threshold, we will not. If we set the threshold very high, then we will have a high probability of our signal fires being accurate (meaning that the predictions are likely to be true), but we will predict only a small percentage of price movements. If we set the threshold very low, then we will have a relatively inaccurate signal (meaning that the predictions are unlikely to be true), but we will predict a large percentage of price movements. A good choice of threshold will balance accuracy with coverage, so we want to

choose a threshold that predicts as many ticks as possible while maintaining reasonably good accuracy. We have found experimentally that roughly 50% accuracy is a sweet spot: if we try to be much more aggressive and allow significantly more false positives than true positives, we quickly reach a point of diminishing returns, where the small number of additional true positives we get is quickly dwarfed by the vastly growing number of false positives. For this reason, we tune our threshold to achieve an approximate balance between true positives and false positives.

For these particular coefficients, this happens to result in a threshold of 0.32. One might wonder: why not 0.5? We may wonder, of course, how far our estimates drift from the true probabilities, and if this is leading to a weird threshold calculation. But actually, we should expect a threshold under 0.5. The reason is: if we fire whenever the probability is ≥ 0.5, we are often firing when the probability is 0.6 or 0.7 for example, resulting in relatively more true positives. So we can afford to lower the threshold below 0.5 to reach an equal number of true and false positives. In this case, the balance point appears to be approximately 0.32.

It is important, however, not to ascribe too much meaning to the exact numbers. For instance, we can write the same firing criterion in many equivalent ways:

1. $\frac{1}{1+e^{-(c_0+c_1 bids+c_2 asks+c_3 bids1+c_4 asks1)}} \geq 0.32$,
2. $c_0 + c_1 bids + c_2 asks + c_3 bids1 + c_4 asks1 \geq -0.7537718$,
3. $10c_0 + 10c_1 bids + 10c_2 asks + 10c_3 bids1 + 10c_4 asks1 \geq -7.537718$.

We can get from the first representation here to the second by simply rearranging and taking the natural logarithm of both sides of the inequality, whereas we can get from the second to the third by multiplying everything by 10. This is shown merely to drive home the point that it is really the *ratios* between the coefficients and the threshold that matter, not the exact numbers themselves. It is also natural to guess that the level of precision of our coefficients is beyond what's really necessary. We wouldn't really notice the difference on real data if we changed -0.76504 to -0.765, for instance. But we might as well leave things at the level of precision that R provides.

Since we have a formula and a threshold, we can see how this signal performs in action! We simulated its behavior again on data for Dec. 15, 2016 to have an apples-to-apples comparison with our simpler initial attempt. On this day, this formula produces about 1 million true positives and 850,000 false positives. In other words, the signal fires a total of about $1,850,000$ times, and in about 1 million of these cases, the predicted tick actually occurs within the 10ms firing window. We can compare this to our prior result of 540,000 true positives and 711,000 false positives. Clearly the formula is an improvement: we are getting many more true positives while

only incurring a slight increase in false positives. This version of the signal was employed in production at IEX from September 2015 to August 2016.

3 Employing the Signal in an Order Type

Now that we have a basic version of a crumbling quote signal to work with, we can consider how to deploy it on our market. Historically, we have embedded it inside our discretionary peg (dpeg) order. This is a dark order type that rests on the book at the near side, but can exercise "discretion" to trade at the midpoint of the NBBO when the crumbling quote signal is not on. In this way, discretionary peg orders behave less aggressively while the signal is on, seeking to avoid bad trades at the midpoint just before the NBBO moves in their favor.

In December 2016, for example, discretionary peg orders accounted for 32% of the volume on IEX and 50% of the midpoint volume. This represents about 1.5 billion shares. As a group, these shares experienced fewer instances of being "picked off" than standard order types: 7% were picked off, versus 8.5% for standard order types.

Just this month, IEX gained approval from the SEC to release a new version of our primary peg order type which will also incorporate the signal. This order type will rest one tick outside the NBBO and exercise discretion to trade at the NBBO while the signal is off. This is not yet available in production, but we are optimistic that it will perform well once it is deployed.

4 Refining the Crumbling Quote Model: Part I

When we seek to improve our model of a crumbling quote, there are several avenues we can consider. The process of selecting a model involves multiple stages, and the decisions we make at each stage can be revisited as potential opportunities for improvement. To make this more transparent, let us summarize the choices we made in designing our model:

1. We defined our prediction target as price moves within the next 10 milliseconds.

2. We defined the features $bids, asks, bids1, asks1$ to be used in prediction.

3. We defined a class of functions over these features, namely linear functions.

4. We chose a set of training data, imposing some restrictions on our data points.

5. We chose a way of measuring how well a particular linear function of our features does in predicting the log odds of our target variable in our training set.

6. We chose a way of computing a linear function that performed well according to this measurement.

Often we do not have to explicitly think through these last two steps, as standard choices can be made for us in software packages such as R. But it can nonetheless be helpful to remind ourselves that such choices are present, and adjusting them is a degree of freedom we have and may someday want to use. In particular, we may feel that false negatives are worse than false positives, and we may want to adjust how these two kinds of errors are treated when we evaluate how well a particular model "fits." However, all of the different combinations of choices we could consider jointly for steps 1 through 6 above can quickly become an overwhelming, high-dimensional mess, and we want to focus in on the steps where we are likely losing the most ground.

So how do we know where we are losing ground? There are a few ways we can get a broad sense of this. As an illustrative example, let's focus in on step 3. above, where we chose the class of linear functions. If we keep our feature selection of $bids, asks, bids1, asks1$ as fixed, we might note that the number of common value combinations for these features is not too large. The value of $bids$, for instance, is between 1 and 12 (since we exclude IEX itself from our calculations of these variables). The same is true for $asks$, $bids1$, and $asks1$, making a grand total of $12^4 = 20,736$ possible value combinations across the four features (again ignoring very rare events where one of these might be 0).

To get a sense of how much raw predictive power these features have before we impose a linear form on our model, we can test a brute force approach. Going over a portion of historical data, we can simply count how many times each combination of feature values appeared and how many times it was followed by a price tick. As long as our data set is sufficiently large in comparison to our 20,736 possibilities, we will get a pretty good estimate of the probability of a price tick following each feature combination, except possibly for very rare combinations, which do not have a large impact anyway. We can turn this collection of probability estimates into a prediction model by choosing a probability threshold. We can simulate a signal on fresh data then by firing whenever we see a combination of features whose estimated probability of an impending tick is above our chosen threshold. As with a linear model, as we adjust this threshold we see a spectrum of tradeoffs between accuracy and the coverage of true positive events.

What we find, in this case, is that the imposition of a linear function is not costing us much. For this limited feature set at least, the brute force

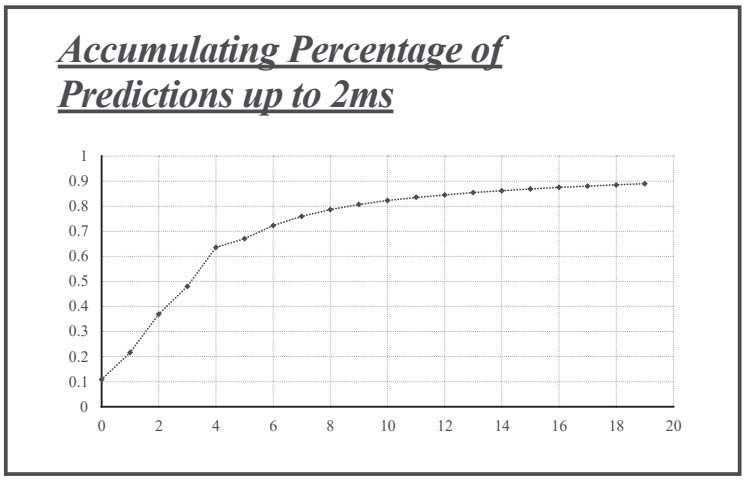
model does not give us a considerably better performance than the best fitting linear function, though this is a step we should revisit whenever we change earlier choices in the modeling process.

Even though it does not seem to be a big loss in this case, we might still ask: why consider imposing a linear function in the first place? If we can get reasonably reliable estimates of the probabilities of ticks following various feature combinations, why not just use these directly and bypass logistic regression? For one thing, a table of 20,736 probabilities is an inconvenient object to include in a rule filing. But more importantly, it does not give us much human-digestible insight into the structure we are modeling. Some might not see this as a problem: if the model works well, do we really care about understanding *why* and *how* it works? Well, we will probably care if it stops working as well. If we cannot succinctly describe and explain our model, it is going to be hard to fix it and improve it over time. It is a common sense rule of good statistical practice: never use an overly complex model when a simpler one will do. Simplicity is a form of protection from over-fitting to training data and overreaction to minor deviations in live data on a day-to-day basis. We should not give it up in exchange for meager returns.

Instead, we will start searching for improvements by revisiting our very first choice: defining 10 ms as the target prediction range. Let's take a closer look at how well that is working for us. For each correct prediction we make, we can examine the gap in time between the market data update that triggered the prediction and the time of the predicted tick. Both times are relative to IEX's view, so more precisely we mean the gap in time between IEX's system processing the market data update that triggered the prediction and processing the market data update that notified IEX of the tick. We note that this ignores the time it takes us to compute the prediction and notify the matching engine, but this is typically quite fast (this could change if we make our model too complicated, but we are currently very far away from that concern).

It turns out that nearly 90% of our correct predictions are actually made in the 2 ms preceding the tick. The following chart illustrates this for a particular day, which happens to be Dec. 15, 2016 again. For each correct prediction, we took the amount of time between the prediction and the tick and rounded it down to the nearest multiple of 100 microseconds. We can then examine the accumulated percentage of predictions as we increase the time scale: the horizontal axis here is in units of 100 microseconds, and the vertical axis is a cumulative percentage of the time gaps between our correct predictions and the corresponding ticks. For example, the y-value of 48% corresponding to the x-value of 3 tells us that 48% of our correct predictions were made less than 400 microseconds before the tick. We can see that the percentage grows rather slowly between 1 and 2 ms, and when we look at the remaining time from 2 ms out to 10 ms (which is not pictured here as

it is too boring), we also see a rather slow growth from 90% to 100%.



This suggests that 10 milliseconds is not an optimal choice for the amount of time our signal should stay in force. In comparison to a 2 ms window, we are leaving it on 5 times longer and only getting a roughly 10% gain in correct predictions. This does not feel like a very good trade off. Predicting a tick 10 milliseconds away was a fine aspiration, but since we are falling well short of this, we might as well embrace the range of predictive power we do have and not waste too much additional time waiting for ticks beyond this. For this reason, we changed our signal to be in effect for only 2 ms. This also changes our predictive target: we are now trying to predict the event of a tick happening within the next 2 ms, not the next 10 ms.

Let's see what happens if we keep the same formula and probability threshold and simply shorten the duration of the signal from 10 ms to 2 ms. Simulating this behavior over market data for the same day, we see that the signal would now fire about 1,970,000 times, consisting of approximately 950,000 true positives and 1,020,000 false positives. To understand why the signal fired a bit more frequently when we shortened the duration, we note that while the signal is on, we choose not to reevaluate the formula to allow new predictions. This is why leaving it on for less time can lead to a few more fires in our implementation. It is important to note here as well that even though our count of false positives has grown, the total time wasted in false positives has been dramatically reduced from about $850,000 * 10$ ms to $1,020,000 * 2$ ms. This is a reduction from about 142 minutes aggregated across all symbols to about 34 minutes.

This is a worthwhile reduction in the overall cost of false positives, though it comes at the expense of losing a small number of true positives. Not to worry though, we can regain those lost true positives and a bit more by finding some additional features to enhance the fit of our signal.

In other words, we will consider augmenting our selection of features in step 2 from our model selection steps listed above. As in our choice of a linear function, we favor simplicity here in feature selection, but limiting ourselves to these four basic features is a bit too conservative. For example, we are currently treating all venues the same, even though desertion of the near side by certain venues may be more indicative of an impending tick than other venues. Fitting separate coefficients for each venue is a bit unwieldy, but it appears experimentally that giving some additional weight to BATS, EDGX, and NASDAQ can be useful. For this, we define the following feature:

$$D := \mathbb{I}[BATS] + \mathbb{I}[EDGX] + \mathbb{I}[NASDAQ],$$

where $\mathbb{I}[BATS]$ is an indicator variable that equals 1 if BATS was present on the near side 1 ms ago but is not now, and equals 0 otherwise. $\mathbb{I}[EDGX]$ and $\mathbb{I}[NASDAQ]$ are defined analogously, so the value of D is always between 0 and 3.

Another thing we may consider is the ordering of events. So far, we have only considered cumulative changes in how many venues are on the best bid and offer, regardless of the order in which these individual adjustments occur. The same values of $bids, asks, bids1, asks1$ may describe significantly different sequences of events. For example, consider a case where $bids1 = 3, asks1 = 3$. From this point on, the following may happen: BATS leaves the bid, Arca leaves the bid, NASDAQ joins the ask, NASDAQ joins the bid, EDGX leaves the bid. At this point, we have $bids = 1$ and $asks = 4$. But there are many other scenarios that yield this same result. For example, perhaps BATS joins the ask, BATS leaves the bid, EDGX leaves the bid. We might suspect that if the *most recent* events have been venues deserting the near side, than a tick is more likely to occur. For this reason, we define another feature E, which $= 1$ if the most recent two events have both been venues leaving the near side, and $= 0$ otherwise.

We now have 6 features to include in our logistic regression model to predict a downward tick: $bids$, $asks$, $bids1$, $asks1$, E, and D. We thought up and tried several other features like these at this stage of our research, but these six seemed to capture the bulk of the predictive power without introducing unnecessary levels of complication. So we again used historical data to find the best choice of coefficients for weighting these factors in our signal, as well as a threshold to apply on the result. This produced the following model:

$$\text{fire when } \frac{1}{1 + e^{-(c_0 + c_1 bids + c_2 asks + c_3 bids1 + c_4 asks1 + c_5 E + c_6 D)}} \geq 0.6$$

$$\text{where } c_0 = -1.3493, \;\; c_1 = -1.1409, \;\; c_2 = 0.2671,$$

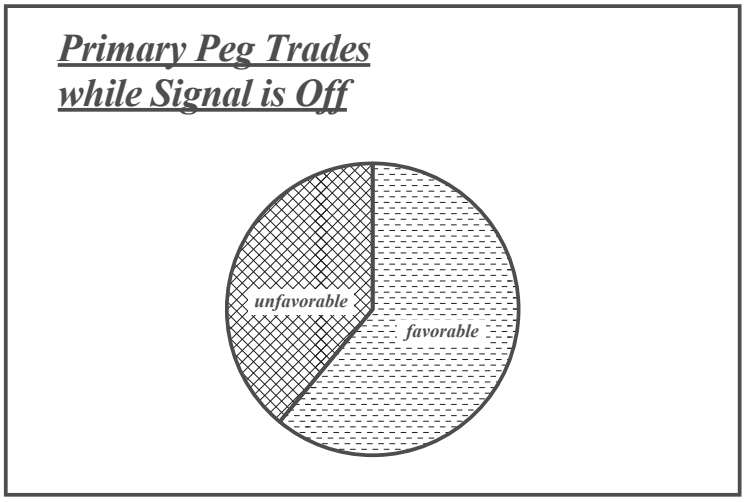
$$c_3 = 0.5141, \;\; c_4 = -0.190, \;\; c_5 = 0.1347, \;\; c_6 = 0.6862.$$

Again, we note that scaling all coefficients and the threshold 0.6 by the same constant factor would result in exactly the same criterion, so we shouldn't read too much into the particular number of 0.6. This formula overall is not less aggressive than the previous one. The particular number 0.6 arose because we calculated our coefficients in R using weights on our training data that made false positives errors less costly than false negative errors, which means we should no longer interpret this threshold 0.6 directly as our probability estimate of the likelihood of a tick. Simulating its behavior on our test data from Dec. 15, 2016, we see that this new criterion results in roughly 1,060,000 true positives and 975,000 false positives. This is with a 2 ms firing window, so it should be compared to the 950,000 true positives and 1,020,000 false positives that we obtained using the prior formula with a 2 ms firing window. The new formula is a clear win by this metric: we get more true positives and less false positives. This version of the signal was employed in production at IEX from August 2016 until March 2017.

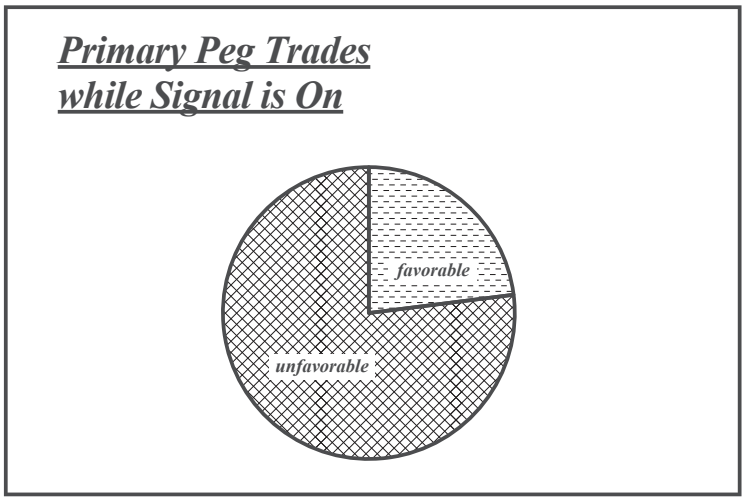
5 Behavior on Our Market During the Signal

Let's take a brief detour from working to improve our signal and pause to examine how it corresponds to behavior on our market. Note that the formula we arrived at in the previous section is indeed the version of the signal that was employed in production at IEX in December 2016, the month that we will take as an illustrative example here. We look at the data set of primary peg trades that happened on IEX during that month. We classify each trade as favorable or unfavorable from the perspective of the resting order based on a 1 second markout. More precisely, we compare each trade price to the midpoint of the NBBO 1 second later. If the trade price was better (from the perspective of the resting order), we call this "favorable". If the trade price was worse or the same, we call this "unfavorable." For each trade, we also check whether the signal was on in that symbol at that moment.

For context, we note that the signal typically fires 1.6 to 2 million times a day, so the total time the signal is on is usually less than $2,000,000 * 2$ ms $= 4,000,000$ ms $= 4,000$ seconds a day, which is an average of less than half a second per symbol. Surprisingly, about 21% of the volume of trades in our data set occur in this very small portion of time while the signal is on. Less surprisingly, there is a systematic difference in quality between trades that occur when the signal is on versus when the signal is off. Here is what the distribution of trades (by volume) looks like from the perspective of the resting order when they occur during signal off periods:



Overall, this behavior looks pretty appealing. A strong majority of the time (about 61% of volume), the 1 second markouts are favorable. However, if we look at trades that occur while the signal is on, we see a very different picture:



Here, only 23% of the trading volume is favorable and the remaining 77% is unfavorable. This clearly demonstrates that the signal provides a meaningful protection against unfavorable trades, at least at this time horizon. It is important to note that the choice of an appropriate time horizon is crucial. For example, if we were to choose 2 ms as a time horizon, we would not get very meaningful results. By design, the trades that occur while the signal is on are highly likely to be deemed unfavorable by this metric, as a correct signal prediction means the price will move into the resting order within 2 ms. But by looking at 1 second markouts here instead, we do learn

something new, namely that our signal can be leveraged to improve the experience for resting orders at this greater time horizon. These results give us promising insight into how our new version of primary peg may perform.

One might also wonder what happens if we divide the trades based on signal accuracy. The 21% of trading volume in our data set that occurred in periods where the signal was on breaks down into 14% while the signal was on and correct (true positives) and 7% while the signal was on and incorrect (false positives). For trades that occurred during true positive signals, 12% were favorable (by volume) while 88% were unfavorable. During false positive signals, 47% were favorable while 53% were unfavorable. There are likely many contributing factors to why the effect on 1 second markouts is much stronger during true positive signals. One is that false positives by definition mean the price didn't move into the resting order within 2 ms, so this selection of events will have some lingering correlation with the 1 second markouts. Secondly, traders may be acting less aggressively than our signal since they may stand to lose more from false predictions, so they may only act when they are more confident a price change is coming, which would correlate more with our true positives. Presumably, many of our false positives occur in times when our prediction probability was fairly near our threshold, so even if a trader was acting on the same probability estimate, he or she might choose not to trade in these moments. What is perhaps most interesting here is that the 1 second markouts of trades during false positive signals are significantly less favorable than those that occur while the signal is off. This suggests that even when the signal is wrong, it does significantly correlate with trading activity. This might suggest that traders are basing behavior on models that are similar to or at least correlated with our signal model.

6 Refining the Crumbling Quote Model: Part II

Though we have seen that our crumbling quote signal performs reasonably well already, we suspect it can still be dramatically improved. One might hypothesize, for instance, that the patterns that characterize a crumbling quote differ substantially based on characteristics of the symbol being traded and/or the current trading conditions. We might wonder if the model should vary based on time of day, or average price of a symbol, or average daily volume of a symbol, etc.

To test the influence of time of day, for instance, we took training data only from a particular hour of the trading day and performed logistic regression again, recalculating the coefficients to perform well on this more targeted data set. We then tested the performance compared to the current model on fresh test data taken from that same hour on a different day. We did this separately for various hours, such as the first hour of trading, the

last hour of trading, and several hours in between. We similarly tried many ways of grouping symbols into a small number of categories, and trained separate models for these categories, then tested them against our current model.

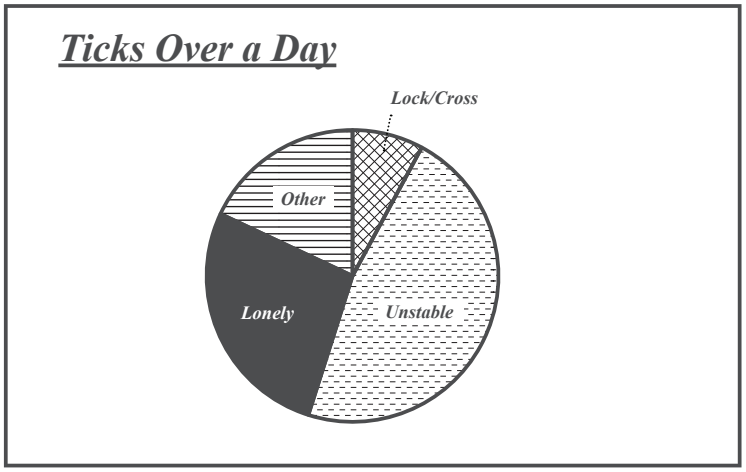
It is easy to get lost in the universe of all possible groupings and features that could be tested, so in designing and executing these tests, we tried to maintain focus on the forest through the trees. Some basic patterns did emerge. First, time of day did not appear to be significantly helpful, at least not at this level of granularity. Second, reasonable ways of grouping the symbols into a small number of clusters did result in meaningful results, but perhaps not in as many ways as we expected. The basic shape of the signal did not seem to meaningfully change when we optimized our coefficients to a particular class of symbols, but the optimal probability threshold depended on spread. More precisely, the "sweet spot" of roughly equal true and false positives moved around a bit in a way that was correlated with the average spread in a symbol. We found that we could fire at a lower, more aggressive threshold in high spread symbols (those with average spread above \$0.035), obtaining more true positives without accumulating too many more false positives.

However, all of the potential gains we discovered in these kind of tests were fairly modest, all combined amounting to about a 25% increase in true positives while maintaining approximate equality with false positives. To go in search of *immodest* gains, we will now take a look at the basic anatomy of price ticks throughout a single day and look for clear targets for improvement. We will classify ticks into a few categories.

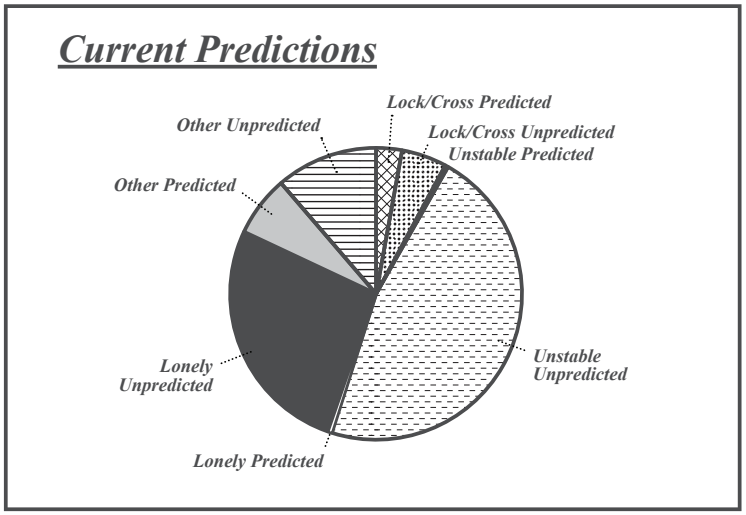
We will design our categories to capture a few of the possible explanations for why we might fail to predict a tick. One obvious case is ticks that occur in relatively unstable conditions, where our features of $bids1, asks1$ cannot be meaningfully calculated. We will classify a tick as "unstable" when: for the last NBBO update preceding the tick (our last potential chance to predict it), the price stability condition over the preceding millisecond is unsatisfied. Another problematic class of ticks are those that occur when a single venue has been alone at the NBB or NBO for awhile and then suddenly moves. Trying to predict exactly when such ticks will occur would seem to require true black magic, and we shouldn't judge ourselves too harshly for missing them. To make this precise, we will classify a tick as "lonely" when: for the last NBBO update preceding the tick, the price has been stable for the past 1 ms, but the number of venues at the near side has been $= 1$ for that interval. It is also helpful to identify classes of ticks that are not as valuable to predict, like those immediately following a locked or crossed market. Since IEX already prevents dpeg orders from exercising discretion during a locked or crossed market, predicting these ticks with our crumbling quote signal is not necessary.

As an illustrative example, we again consider ticks over the course of the

trading day for Dec. 15, 2016. There were approximately 11,245,000 times over this day where either the NBB moved down or the NBO moved up. This is the total set of ticks we will work with. If we break them down into our categories, we get the following distribution:



We can see here that about 8% of ticks occur immediately after a locked or crossed market, 47% of ticks occur during unstable conditions, 27% of ticks occur with one venue alone at the near side for a significant stretch of time, and 18% of ticks occur outside of these categories. Let's now look at how our signal predictions overlap with this:



In total, we are predicting about 10% of the day's ticks. In the unstable and lonely cases, we are predicting essentially no ticks (this is why you cannot even really see their slices in the pie chart). This is no surprise, since we do not allow our signal to fire in unstable conditions, and lonely

ticks are rather unpredictable by nature. In the lock/cross category, we are predicting a significant fraction of ticks, but this is not very important. We are predicting about 37% of the ticks in the "other" category, which is where we expect our useful predictions to fall.

We can hope to improve upon this 37% of course, but we can also see from this breakdown that imposing our 1 ms price stability condition is very costly. Right off the bat, we give up on even trying to predict nearly half of a day's ticks!

To have a fighting chance at predicting ticks in less stable conditions, we need to define new features that do not depend upon 1 ms of stability. As a first attempt, let's return to basics and consider the form of the market data we are receiving. What we receive in market data is of course more granular than mere snapshots of the NBBO at particular moments in time: it reflects the full sequence of moves that individual venues make as they update their best quotes. If we are watching the NBBO updates in a certain symbol and we freeze at a moment in time, we can look back to the most recent price tick and examine what has happened between then and now. It may be that many things have happened - someone joined the bid, then someone left the bid, then someone joined the ask, etc. We can encode this sequence of events like so: $+1bid$, $-1bid$, $+1ask$, etc.

The length of the sequence of events since the last price change will vary. Looking at very long sequences is likely to complicate the model more than it helps - for now we will cap the length of the event sequence to 4, meaning that when there have been more than 4 events since the last price change, we only consider the most recent 4. We start by also considering the current number of bids and asks, as well as the direction of the last price change (bid up, bid down, ask up, or ask down). This means we associate each NBBO update with the following data:

- bids: the number of venues currently on the bid
- asks: the number of venues currently on the ask
- direction of last price change: one of (bid up, bid down, ask up, ask down)
- ordered sequence of ≤ 4 most recent events (within current price level) encoded as $+1bid$, $+1ask$, etc.

These features can form the basis of a new model that attempts to use this data to predict a price tick within the next 2 ms, potentially even in moments of relative instability. To get a general sense of how well a model based on these features might perform, we can start with a brute force approach. For each possible combination of values for the features above, we can count how many times it occurred over the course of one day, and look at how many of those instances were followed by price ticks in each

direction with the next 2 ms. For example, let's suppose there were 100 times where the current number of bids = 2, the current number of asks = 5, and the recent event sequence was: $+1ask$ then $-1bid$ then $-1bid$. Let's further suppose that out of these 100 times, 60 of them were followed by a reduction in the bid price within the next 2 ms. Upon seeing this, we might reasonably say that our signal should predict a down tick whenever it sees this particular combination of values. Looking over a full day's worth of data across all symbols, we can make a table of all the combinations of these features that occur, and for each calculate what percentage of the time is this combination followed by a down tick or up tick within 2 ms. We can then pull out all of the combinations where this percentage is above a threshold, and declare these to be times when our signal should fire. We will use 42% as a threshold for now because it seems to be the sweet spot roughly equalizing true and false positives.

We should note that we are not advocating using this kind of look up table as a signal definition in practice. We are merely using this as an intermediary step to see how much information is contained in these features before trying to fit a simpler model to them. This way, as we fit the model, we can understand how much of its success/failure is due to the model fit as opposed to the feature selection.

We now take this table of feature combinations that represent a reasonable likelihood of impending ticks for our training day, and simulate what would happen on a fresh day's worth of data if we fired the signal exactly when these combinations occur. The training data this time was from November 18, 2016, while the testing data was from November 16, 2016. Our previous signal formula produced roughly 990,000 true positives and 870,000 false positives on that day. This new look up table approach produced roughly 2.6 million true positives and 2.7 million false positives.

As a revealing point of comparison, if we had simply taken the current formula and lowered it's firing threshold to 0.2 from 0.6 to make it much more aggressive, we would have produced 2.1 million true positives and 5 million false positives. So the look up table on these features represents a far superior fit, at least in terms of counts of true positives versus false positives.

However, the true positive and false positive numbers don't tell the full story. In signal modeling as in life, it is dangerous to focus single-mindedly on one metric of success. If we do, we are likely to optimize it too far, at the expense of other important things. In this case, the important thing we've neglected is time.

In an initial search for structure in the subset of event patterns in the table that were most commonly followed by ticks, we noticed a simple feature. Most of them were predicting impending ticks at the very last opportunity. More precisely, the predictions for downward price movements were largely coming only when the number of bids had dwindled all the way down to 1,

and similarly upward price movements were only predicted after the number of asks had dwindled all the way to 1. To get a rough sense of this effect, we compared it against the behavior of the current signal formula over one day's worth of data (namely Dec. 8. 2016). On that day, the current signal formula made about 1,058,000 true positive predictions, and roughly 608,0000 of those correct predictions came in moments where the crumbling side had already dwindled down to 1. That left about 450,000 true predictions that were made at earlier points during the crumbling process. This means about 57% of predictions came very late in the crumbling process, while about 43% came a bit earlier in the process, giving us more of a head start before the impending tick. Using the look up table approach previously discussed, we would have gotten about 3,090,000 true positives for this same day, but a whopping 2.8 million of these predictions came only when the crumbling side reached 1. This is not only a vastly higher percentage (92% instead of 57%) but even the absolute number of earlier predictions has fallen to about 250,000, so we have lost at least 200,000 opportunities to predict a tick earlier in the crumbling process.

Now, we might note that predicting a tick at the last possible moment is better than not predicting it at all (though even this may be false, as predictions that are unlikely to traverse our system in time really just clog up our internal processing without serving a purpose). But naturally we do not want to lose ground on ticks we were already predicting correctly. Predicting the same tick a bit earlier is a benefit to us that the true positive count doesn't capture, since the earlier we can predict a tick, the more bad trades we can prevent in the time between the signal and the tick. In many cases, it may be that there simply isn't enough information to predict a tick much in advance, but this of course can't explain lost ground between the current formula and the lookup table.

So how do we approach the higher true positive count of the lookup table without losing ground on prediction times? One easy approach would be to use both methods together. We could fire the signal when either the current formula OR the table predicts that a tick is imminent. Something like this is certainly an option, but it's not ideal for several reasons:

1. If the true positives of the two approaches correlate more than the false positives, we'll need to re-tune the parameters to get the true and false positives to stay roughly equal, and this will likely degrade the total true positive count we can capture as a result.

2. More importantly, blindly gluing together two models that differ without understanding how and why they differ is unlikely to be the best long term approach. Nor does it help us demystify the look up table and distill it into a more palatable computation.

Instead let's ponder - why might looking at the pattern of the most

recent events around the NBBO lead to later predictions than looking at a time slice exactly 1 ms ago? One possibility is that when our current signal makes early predictions, these predictions are not as accurate as the probability threshold of 0.42 that we imposed in selecting patterns from our look up table. But this doesn't seem to be the case. For the test data, the current signal had a 45% accuracy on times when it fired without waiting for the near side to crumble all the way to 1. Also, if we relax our 0.42 probability threshold in selecting patterns from our table to something a little bit lower like 0.4 or 0.38, it doesn't seem to resolve the issue.

What this seems to suggest is that time is a relevant feature: lumping in events that are tightly clustered in time with events that are more spread out in time into the same pattern loses vital information that can be leveraged to make earlier predictions. In getting rid of the 1 ms stability condition, we've gotten rid of timing information completely in our approach, and perhaps have thrown the baby out with the bath water.

This is easily fixed. We can build features that still take time into account without relying on a fixed window of price stability. Here's a first attempt: whenever we receive an NBBO update, we look back over a window of time that ends either 1 ms ago or at the most recent price tick, whichever we encounter first. For example, if the last price change was 3 ms ago, we will look back over the last 1 ms. If the last price change was 0.5 ms ago, we will look back only over the last 0.5 ms.

Now we face the question, what do we do with the pattern of events that we observe in that window of time? If the window is cut short by a recent price change, it doesn't make much sense to compare the number of bids now to the number of bids right after the price tick, for instance. So instead of relying on a feature like $bids1$ (the number of bids 1 ms ago) to predict a downward price movement, we define a feature called BL that represents the current number of bids minus the peak of the number of bids over the window we are considering (this is a measure of bid loss). We can analogously define features that capture gain in asks, gain in bids, and loss in asks. We can also define features like EP (an abbreviation for "Event Positive"), which $= 1$ if the most recent event was someone joining the bid, and $= 0$ otherwise. Similarly, we can define EN (an abbreviation for "Event Negative"), which $= 1$ if the most recent event was someone leaving the bid, and $= 0$ otherwise.

To predict a downward price movement for example, we will try examining the following set of features:

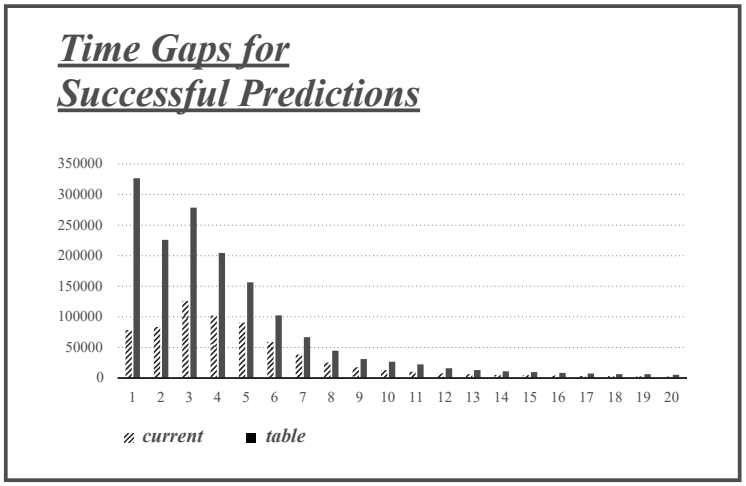
- bids: the current number of venues at the NBB
- asks: the current number of venues at the NBO
- BL: current bids - max number of bids over the window
- AA: current asks - min number of asks over the window

- EP: = 1 if the most recent event was a venue joining the NBB, = 0 otherwise
- EN: = 1 if the most recent event was a venue leaving the NBB, = 0 otherwise
- EEP: = 1 if the 2nd to most recent event was a venue joining the NBB, = 0 otherwise
- EEN: = 1 if the 2nd to most recent event was a venue leaving the NBB, = 0 otherwise.

We will do things analogously for predicting upward ticks, just reversing the roles of bids/asks in all of the features. Now that we have a candidate list of features, we can again consider linear models, look up tables, or other classes of functions over them. We'll start with the look up table approach to get a raw sense of the predictive power of these features. For this, we can go over a day's worth of training data, simply counting how many times each combination of values for the features occurred, and how many of those occurrences were followed by ticks. We then selected from the table all of the combinations of values that were followed by a tick more than 40% of the time.

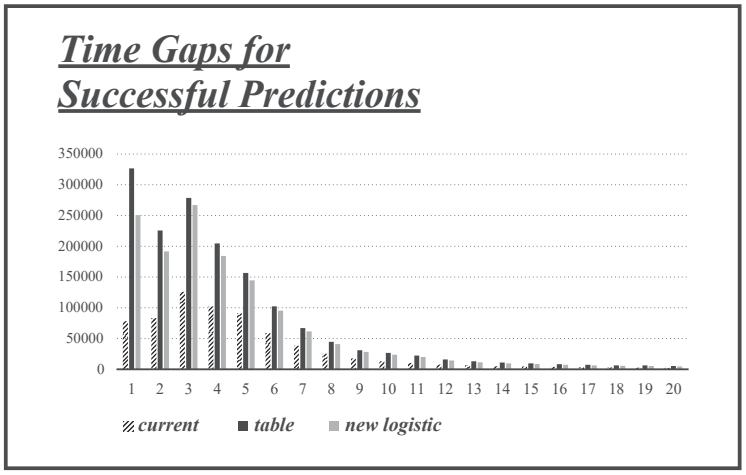
We can use this as a look up table to decide when to fire a signal and test its performance on a fresh day of data. For comparison purposes, we test it on data from Nov. 16. On that day the current signal formula resulted in roughly 990,000 true positives and roughly 870,000 false positives, while our previous look up table based on time-insensitive patterns of length $\leq$ resulted in 2.6 million true positives and 2.7 million false positives. Our new look up table produces 2.2 million true positives and 2.3 million false positives on this day. So this is slightly less than the previous look up table approach, but again, this does not tell the full story. The hope is that this new set of features that also incorporates time without requiring stability can achieve the best of both worlds: a high true positive count without sacrificing accuracy or the timeliness of predictions.

Here's a chart that suggests it does pretty well at achieving this goal:



This chart compares the current signal to this new table look up over a fresh day's data, in this case Dec. 20, 2016. The horizontal axis here represents 100 microsecond buckets. For each correct prediction, the time gap between the prediction and the tick is measured in microseconds, rounded up to the nearest multiple of 100, and placed in appropriate bucket. The vertical axis represents a count of how many predictions fall into each bucket. We can see that the new predictions made by the table based approach are often made with a slim time margin, but we haven't lost ground in any of the time buckets. So at every time range, the number of correct predictions made by this new table-based approach outnumber the correct predictions made by the current formula. We can also consider the interesting metric discussed above, namely how many correct predictions are made before the near side has crumbled to 1. For the current formula on this day, about 290,000 of the 680,000 true positives were fired when the near side was still > 1. This is roughly the same 43% we saw before. For the table based approach, about 280,000 of its 1,570,000 true positives were fired when the near side was still > 1. This is a much smaller percentage (18%), but at least the absolute number has not dropped too considerably.

Interestingly, this number 280,000 jumps up to about 370,000 if we replace the look up table by a logistic regression trained on the same 8 features instead. The logistic regression achieves about 1.4 million true positives (with false positives roughly equal), which is not that far below the look up table's 1.6 million for this day (also with false positives roughly equal). If we throw in this new logistic model to the time gap comparison, we see it still does quite well:



This shows that a lot of the true positives we miss by moving from a look up table to a logistic regression in these new features are not that costly to miss, because we were only predicting them with an extremely tight time margin anyway, making those predictions relatively less valuable. Taken all together, it seems we've hit upon a new set of features that can achieve about twice the true positives of our current model while the false positives stay roughly in line, and the time gaps for our predictions at least don't get worse for the ticks we were already predicting, and we are still getting some considerable gains even pretty far out on the time margins. This is great news!

There are still several tweaks we can consider on top of this new set of features. For one, now that IEX is an exchange, we might consider incorporating our own presence or absence at the NBB,NBO into the calculation of variables like *bids* and *asks*. When we tried retraining and testing our model with this change, we found it made the signal slightly less accurate. There are several reasons why this may be happening. For one, the speed bump is doing its job, so IEX quotes may not crumble in the same way that other quotes do. And of course, the mix of participants on IEX and how they are using the venue may systematically differ from the compositions of participants and strategies that inhabit other venues. Whatever the reasons, including IEX in features like $bids, asks$ appears to slightly dilute their predictive power.

This raises a natural question: are there other venues who inclusion in variables like *bids* and *asks* is either counterproductive or largely pointless? It turns out the answer is yes. When we limit our features to only considering eight venues, namely Arca, BATS, BATS BYX, EDGA, EDGX, NASDAQ BX, NASDAQ, and NYSE, the predictive power of the model is essentially the same. Interestingly, when we narrow things to these eight venues and retrain the coefficients, the false positives seem to increase slightly, but the correct predictions sometimes occur a tiny bit earlier, giving us a bit more

breathing room between the prediction and the tick. Both effects are barely noticeable, so it's a close call. But we prefer to err on the side of leaving things out of the formula rather than in (a good general rule of thumb for statistical practice), so we have decided to exclude the venues outside of these eight for now. Conversely, we find once again that it is helpful to give a little extra weight to the venues BATS, EDGX, and NASDAQ.

We also previously observed that there may be benefits to tuning our firing threshold according to spread. To incorporate this cleanly into our new model, we decided to make the firing threshold a function of the current spread. This is a bit more robust than clustering symbols by average spread over a given period, as it eliminates edge cases such as what to do for a new symbol that does not have a well-defined average spread yet.

To find a good fit between spreads and thresholds, we segmented our training data roughly according to spread and separately calculated a threshold for each segment that resulted in an equal number of true and false positives. The overall result of all this tinkering is the following criterion for firing the signal (this is for predicting a downward tick, the case of an upward tick is analogous):

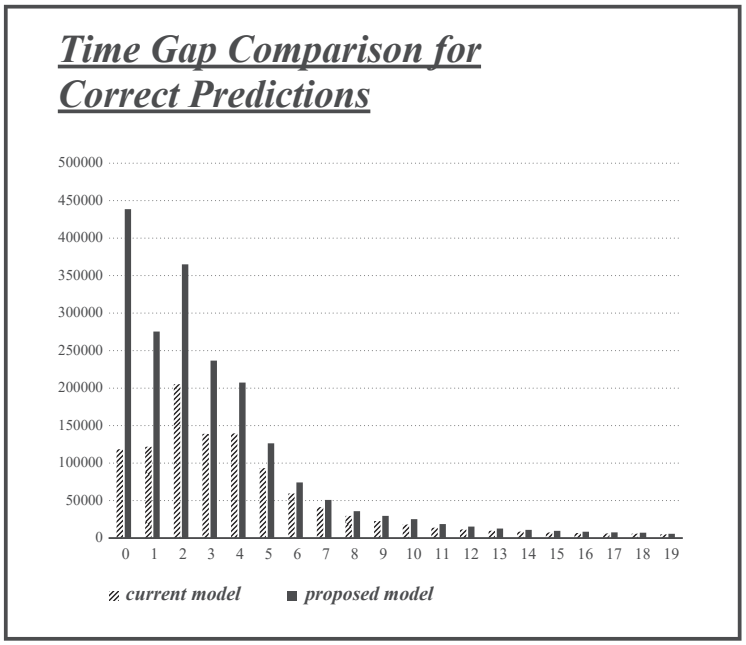
$$\frac{1}{1+e^{-(c_0+c_1 bids+c_2 asks+c_3 BL+c_4 AA+c_5 EP+c_6 EN+c_7 EEP+c_8 EEN+c_9 D)}} > f(apx-bpx)$$

where $c_0 = -1.2867$, $c_1 = -0.7030$, $c_2 = 0.0143, c_3 = -0.2170$, $c_4 = 0.1526$, $c_5 = -0.4771$, $c_6 = 0.8703$, $c_7 = 0.1830$, $c_8 = 0.5122$, $c_9 = 0.4645$,

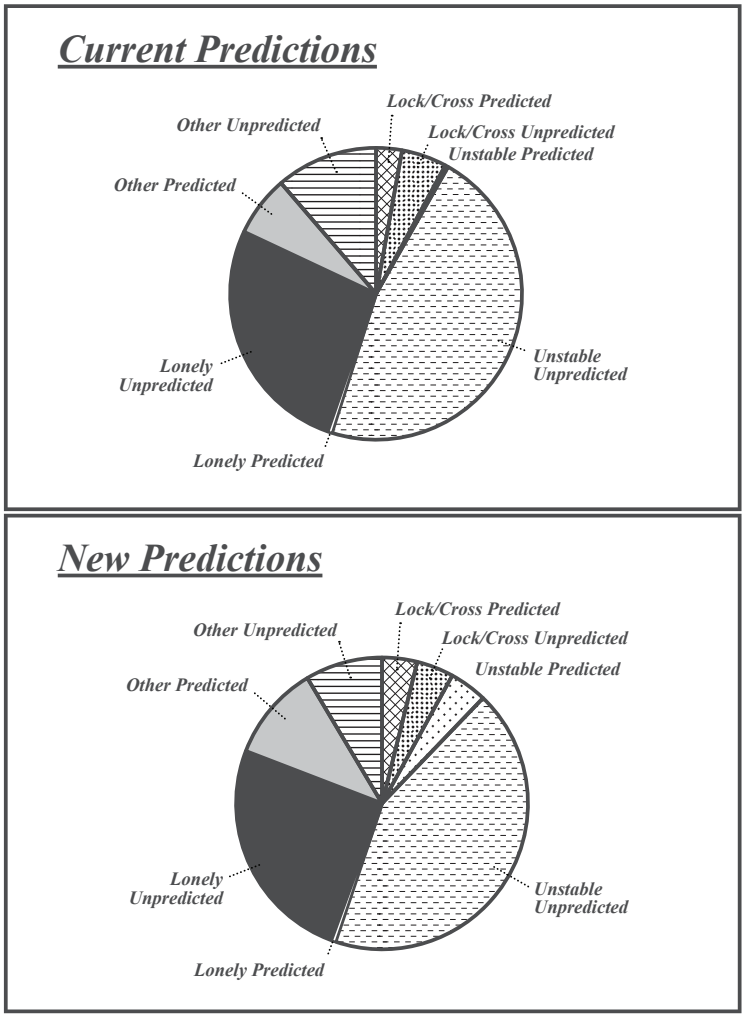
$$\text{and } f(apx-bpx) = \begin{cases} 0.39, & \text{if } apx - bpx \leq \$0.01; \\ 0.45, & \text{else if } apx - bpx \leq \$0.02; \\ 0.51, & \text{else if } apx - bpx \leq \$0.03; \\ 0.39, & \text{otherwise.} \end{cases}$$

The features *bids* and *asks* are now always numbers between 1 and 8, as they are calculated only by looking at the eight venues Arca, BATS, BATS BYX, EDGA, EDGX, NASDAQ BX, NASDAQ, and NYSE. More precisely, the variables *bids* and *asks* are now calculated by counting how many of these 8 venues are at the best bid/offer among these venues, which sometimes differs from the true NBBO calculated across all venues. In our experiments, it seems this is a rare enough occurrence that it's effect on performance is negligible. The features BL and AA also consider only these venues in their calculation. The features EP, EN will be 0 unless the most recent event among these 8 venues was a join or desertion of the best bid among these. The features EEP, EEN behave analogously for the 2nd most recent event among these 8 venues, but with the additional restriction that they will be 0 if that event occurred outside the time window being considered. Similarly, D is a count of how many of the three venues BATS, EDGX, NASDAQ were on the best bid at some point in the window under consideration and have since left.

Let's take a look at how well this new signal candidate performs and compare it to our current version. On our example day of December 15, 2016, our current formula resulted in about 1 million true positives and 975,000 false positives. This new candidate formula would have produced about 2 million true positives and 2.1 million false positives. So we are correctly predicting roughly twice as many ticks, without compromising too far on the accuracy (our false positives still are not too much greater than our true positives). In addition, we are making more correct predictions at every time scale. In the chart below, the y-axis indicates the number of correct predictions made at each time scale, while the x-axis is indexed by increments of 100 microseconds. Here we took each time gap between a correct prediction's generation and the anticipated tick and rounded this down to the nearest 100 microseconds:



We can breakdown the ticks we are predicting into the categories we considered above, and compare the current signal's correct predictions to this new candidate:



Overall, we are predicting about twice as many ticks, now roughly 17% of the ticks on this particular day. We can see here the gains we've made in predicting ticks that occur in unstable conditions, as well as in the "other" category. And if we ignore the lonely ticks that are inherently unpredictable as well as the ticks following locked or crossed markets which are unnecessary for us to predict, the gain is even clearer. If we sum the predicted ticks in the unstable and other categories for the current signal, we get roughly 800,000 on this day. For the new model, we get roughly 1,660,000. There are about 7.3 million ticks in these categories over the day, so we are predicting about 23% of these.

We might wonder: why are we doing better even at predicting ticks in stable conditions? Our previous variables $bids1$ and $asks1$ may have been well-defined in these cases, but that does not mean they were an optimal choice. Looking at the maximum drop in the number of bids, say, over a 1 ms window of time, may capture more relevant information for predicting

an impending tick than the more arbitrarily regimented definition of $bids1$.

This new signal is the culmination of our research so far and has just been deployed. We are excited to see how it behaves on our market in practice. Our research into the signal continues as always, and we expect to keep updating and improving it over time.

7 Conclusion

The market for U.S. equities is a diffuse and constantly shifting landscape. Participant strategies, technology, and regulation all evolve in complex and interactive ways, requiring vigilance and providing ceaseless opportunities for innovation. IEX does not seek to rest on our laurels, but rather to remain on the forefront of technological innovations that are designed uniquely to protect investors. Our continuing development of the signal is a key piece of our larger mission: to bring transparency to the marketplace and empower ordinary investors to compete on a level playing field with entrenched special interests. We have achieved great gains already, and we will continue to employ our collective financial and scientific expertise in the service of investors.