Refining Proxy Symbol Selection for Distilled Impact

Allison Bishop Matt Schoenbauer

1 Introduction

Typical metrics for execution performance of trading algorithms in US equities are fundamentally based on price. The trade prices achieved by the algorithm are compared to various benchmarks, such as volume-weighted average price (VWAP) and arrival price. Understanding and modeling market impact, whether for pre-trade predictions or for after-the-fact analysis also demands grappling with the dynamics of price. However, price data is incredibly noisy. The price is changing rapidly at all times, and due to the regular and significant variance in price it is challenging to figure out what drives this change. This poses problems for all of the above tasks, and careless scientists are bound to see relationships and structures that are merely due to random chance. In order to enable robust data science in this domain, we need to cut away at the unwieldy level of noise in price data as much as we can.

We made some first steps in this direction when releasing our initial design for Distilled Impact [1]. This paper laid the conceptual framework for the reduction of market noise in price data for US equities and tested the framework with a few basic models. Here, we will review this framework and further develop it, focusing more extensively on models and proxy symbol selection.

Summary In this paper, we quickly review the goal of Distilled Impact as described in [1], and provide additional motivation for pursuing this research path. Then we pick up where [1] left off, and consider a larger collection of distilled impact models and model selection procedures.

The models we consider are linear models with features given by relative price changes of a collection of ETFs, larger than the collection considered in [1]. We consider several different methods of feature selection and ultimately conclude that the model that consistently performs better than any others is a simple model that simply consists of a single feature, the price changes of the ETF with the strongest correlation with the symbol in question.

2 The Distilled Impact Framework

The broad goal of our Distilled Impact research agenda is to understand and separate the drivers of price change of a symbol from within the tick data from that symbol and from elsewhere in the stock market. This will have implications for our algorithm design, as an effective distilled impact tool will allow us to more effectively determine how our actions influence the market. In addition, this will be helpful for transaction cost analysis, since good distillation of market forces may help us grade our performance more effectively than standard metrics like slippage vs. VWAP and slippage vs. arrival (see [1] for more information).

The basic price modeling framework laid out in [1] for a stock symbol S during time period t (usually hours, days, or 10 minutes) is given by the following equation:

$$M(S,t) = A(S,t) + O(t) + C(S,t) + N(S,t). \tag{1}$$

Here,

1. $M(S,t)$ is the relative change in the price of a stock over the time period t. This quantity will either be defined as
$$M(S,T) := \frac{L(S,T) - F(S,T)}{F(S,T)} \tag{2}$$

or

$$M(S,T) := \frac{VWAP(S,T) - F(S,T)}{F(S,T)} \tag{3}$$

where $L(S,T)$ is the last trade price of S during t, $F(S,T)$ is the last trade price of S during t, and $VWAP(S,t)$ is the market volume-weighted average price for S over all of t. We will usually use definition (2), but will specify whenever necessary.

2. $A(S,t)$ is the impact on M due to activity in the symbol S over T. This is a challenging quantity to estimate on its own, so we will seek to quantify it in terms of the other terms in (1).

3. $O(t)$ denotes the overall market movement over the time period t.

4. $C(S,t)$ denotes the impact on $M(S,t)$ arising from activity in areas of the market outside of the market for S. Much of our work in this paper will be focused on building better models for this term.

5. $N(S,t)$ is a noise term that captures the extent to which all the other terms on the right side of (1) fail to together approximate $M(S,t)$. In particular, good models of A, O, and T have smaller values of N.

Our goal is to build tools that allow us to approximate A, O, and C, from historical market data. Since M is clearly defined by market data we can obtain information about a missing term if we have good models for the rest of the terms and $N(S,t)$ is on average small. As indicated above, we will set A aside and work to determine good models for O and C, in hopes that we can use this information to better understand A.

Before we can begin thinking about building models for these terms, we need to understand how the models ought to be graded. Clearly we wish to minimize the size of the N term, but we will need a way to grade our model without having to approximate A. To do this, we will simply seek to minimize the size of the expected sum of A and N. Since A does influence M, there will be limits to the extent to which we can minimize this quantity. Nevertheless it will be a useful grade for how accurate our representations of O and C are. Our metric will be determined by the square sum of N and A, taken for a random dollar traded in the market for the set of trading periods in the "orders" (denoted by τ) in the testing set:

$$\sum_{\tau} (N(S,t) + A(S,t))^2 \cdot NV = \sum_{\tau} (M(S,t) - O(t) - C(S,t))^2 \cdot NV \tag{4}$$

Here NV denotes the total notional value traded in symbol S over the time period t. All the terms on the right side of (4) are either measurable or good candidates for modeling.

In [1], we modeled $O(t)$ and $C(S,t)$ as linear combinations $\sum_i w_i(S)c_i(t)$ of other stock prices $c_i(t)$ during the time period t. We tested a few basic models, and settled on a combination of ETFs–SPY, XLE, XLF, XLK, XLV, and VB–as our first pass approximation of $O(t) + C(S,t)$. As a linear model on external prices is nicely simple and likely still underutilized, we will stick with that framework throughout this paper.

3 An Alternative Description

We can also rephrase this problem in a different light that better relates the program to transaction cost analysis and highlights some of the issues with causal directionality that we will discuss later.

When we look to grade ourselves on our trade execution, we wish to compare our volume-weighted execution price against some benchmark. As described in [1], most of the common metrics have obvious flaws. In particular, slippage vs. VWAP is a somewhat circular metric, as we are influencing the quantity that we are grading ourselves against. The slippage vs. arrival metric avoids this self-referential problem, but it can be incredibly noisy, as our execution prices will be subject to external market forces that continue beyond the time of arrival.

In a perfect world, we would be able to grade ourselves against the volume-weighted average price of all the trades in the specified symbol and time period *in a world where we were not present*.

We cannot determine exactly what this price would be, but the framework described in Section 2 gives us a method of approximating it. In particular, if we carefully choose proxy terms c_i that we are unable or unlikely to influence significantly with our trading, then these terms can be used in a prediction model for the VWAP price in our absence that can be trained on historical market data. Borrowing from Section 2, we can take this model as

$$O(t) + C(S, t).$$

and grade it using the loss function in (4). In particular, for a randomly chosen dollar traded in our historical dataset, we want to minimize the expected square difference between the observed price and our predicted price.

This poses the problem as a standard machine learning model, with a slight twist. We want to choose features and models that allow us to minimize a squared-error loss function, but we should be careful to not use features that could be influenced too much by our own trading.

To illustrate with a specific example, consider the task of building a model for the price of Southwest Airlines during a 10-minute interval using data from other symbols. We might use the price of Delta Airlines to predict this value, and this would likely allow us to reduce the loss in (4) more effectively than we could with a model without any other airline stocks. However, such a model would not serve well when, during transaction cost analysis, we seek to understand the price of Southwest if we were not present. It is quite possible that our trading activity in Southwest would increase the price of both Southwest and Delta, and that our model would be muddied by our own activity. This is similar to the problem with the slippage vs. VWAP metric, although perhaps on a lesser scale.

To avoid this problem as best as possible, we will work to be parsimonious with our choice of symbols in our model, and we will be willing to give up some loss reduction from (4) for more confidence in a lack of causal contamination.

4 A New Distilled Impact Model

The set of possible models encapsulated by (1) is incredibly large. In our attempt to predict $M(S, t)$, we could include many different features from concurrent and historical prices and sizes from both trade and quote data. The only data that is out-of-bounds in the prediction of $M(S, t)$ is the trade and quote data for S in the interval t (assuming we want to avoid the extent of the circularity introduced by the VWAP benchmark).

Rather than performing an exhaustive exploration of all kinds of these features, we will build out carefully from what we know to work, and then once we have found a suitable model, remove any complexity that is not absolutely necessary.

In [1], our winning model for $C(S, t)$ and $O(t)$ included a linear combination of six concurrent relative ETF prices, with coefficients determined per symbol. We will expand this by simply allowing for more symbols in the model.

The choice of including new symbols' relative price changes in the model seems like a simple and obvious one, but the question of *which* symbols to include for S is a much more difficult problem. We were already nearing the possibility of overfitting with only six ETFs in our model, and an attempt to simply throw more symbols into the mix would likely result in a statistical disaster. Therefore, if we are going to improve on the original model, we need a process for *customizing* the proxy symbol set (i.e. the symbols used in a linear combination) for each symbol S.

The process for choosing a small subset of a large feature set in a linear model is known in the field of statistical learning as "subset selection" (see [2]). There are many techniques for going about this: forward step-wise regression, forward stag-ewise regression, lasso regression, least angle regression, etc. On our first pass, we focused on a straightforward subset selection method–a variant of forward step-wise and forward stage-wise regression–that seemed reasonable for our purposes and was easy to implement.

For our selection procedure we wanted to include all the best proxies for $M(S, t)$ from our proxy set, but we could not possibly search over all possible subsets of our proxy set with any reasonable computational efficiency. Instead, our subset selection method proceeds as follows. We first choose a set of potential proxy symbols that can be used in the model for $M(S, t)$ for each S. Then, for each symbol S, we go through a step-wise procedure that greedily selects n new symbols to add to the model, and stops after r rounds for a total of $n \cdot r$ new symbols in the model. We refer to n as the *batch size* and r as the *number of batches*. At the first stage, we rank all of the potential proxy symbols S' in terms of the correlation between $M(S', t)$ and $M(S, t)$, with the correlation weighted by the total notional value traded in S in the given time period. From this, we chose the top n to go into the model. At stage $i > 1$, we have a model consisting of $(i-1)n$ symbols. To choose the next n symbols, we compute the *residuals* (i.e. true values minus model predictions) and rank the remaining $M(S', t)$ by their correlation with these residuals. Again the top n symbols get added to the model, and the model coefficients are updated by retraining model with the new symbols included. This selection procedure is displayed in the pseudocode below:

```
def choose_top_n_predictors(proxy_data,target_data,weights,n):
    # Choose the top n predictor columns from proxy_data by correlation with target_data

    R = []
    for single_proxy_name, single_proxy in proxy_data:
        r = weighted_correlation(single_proxy, target_data, weights)
        R.append((r ** 2,single_proxy_name))

    return sorted(R)[:n]

def select_symbols(proxy_data,target_data,weights,n,r):
    # Select n * r symbols for a linear model

    # Compute R square for each linear model fitting proxy symbol data
    # (columns of proxy_data) to target_data
    R = choose_top_n_predictors(proxy_data,target_data,weights,n)

    # coefficients for individual models
    coefs = linear coefficients for model using all the symbols in R and weights

    active_symbols = symbols in R

    for i in range(1,r):

        drop all symbols in active_symbols from proxy_data
        pred = prediction using linear model from coefs
        residuals = target_data - proxy_data
        R = choose_top_n_predictors(proxy_data,residuals,weights,n)
        add symbols in R to active symbols
        refit a weighted linear model with all active_symbols and update coefs

    return active_symbols, coefs
```

A forward selection procedure (i.e. a procedure that starts with a small number of features and sequentially adds new features to the model) is best for our purposes since we know in advance that we will likely not be adding many features to the model. The first batch would be from potential proxy symbols that are most "similar" to S, and then sequential batches would pick up what previous batches "missed" in their predictions of $M(S, t)$. We did not know in advance how large these batches ought to be, so we initially set it as a varying parameter. Before running any tests, we figured that it is possible that the set of "similar" symbols is properly represented by a single symbol, or that it may require a larger number. If the batch size was too large, we would likely overfit. If it was too small, then the model may miss very good predictors of $M(S, t)$ in early rounds if these predictors are not correlated with later model residuals.

We also added a few more dimensions to the space of possible models. We allowed for a set of symbols to be required in the proxy symbol set for each S. This was intended to fill the role of $O(t)$ in (1): a proxy for overall market movement that does not differ symbol-to-symbol. This also allowed the model from [1] to fit in as a subset of this collection of models. We can simply require that all 6 ETFs from the older model be included in the framework and set $n = 0$. We also considered carefully whether to fit an intercept to the model, and if so, whether to include it in the model. The model intercept plays a strange role here, as it indicates a general directionality to the stock's price independent of the proxy symbols. It seems that a model may pick up this term over a fixed time period, but unlikely that this directionality will persist over long period of time. We can address this by either choosing not to fit an intercept to the model, or to fit an intercept and not include it.

Occasionally, a symbol would not have any trade data in a 10-minute interval, and the $M(S, t)$ value would not be defined. In these cases, we set $M(S, t) = 0$. However, the total notional value for that symbol in such time periods is 0, and since we are weighting our correlations, model observations, and errors by notional value, these incidences will not have an influence on our final choice of model regardless of our choice of replacement value.

5 Proxy Symbol Sets

As described in Section 3, we ideally wanted a set of proxy symbols whose prices are not easily influenced by trading activity in other individual symbols. For this reason, we chose a small set of ETFs as proxies in [1]. In an effort to expand this list, we collected a list of 685 ETFs from etfdb.com and restricted this set by a liquidity ranking when choosing our proxy sets.

The list of ETFS come from the headers "Sector", "Industry", "Commodity", "Natural Resources", "Asset Class Size", and "Asset Class Style" in the database. Each header has a list of ETF themes, and each theme has a list of symbols ranked by market cap. We collected symbols from the first page in each theme in each of the headers listed. Technically, there may be existing ETFs that we didn't collect, but given how large the list was at this point it did not seem particularly important to try and gather more. In addition, the database only contains existing ETFs, and our market data queries on past data may be missing ETFs that either changed names or were delisted. We expected that the loss of this small number of symbols was negligible.

To choose subsets of size N from this list of ETFs, we ranked the ETFs by average daily volume traded in the time period in question, and chose the top N. Average daily volume traded serves as a proxy for liquidity, and we figured more liquid symbols were more likely to serve as useful features in our models and less likely to be influenced by trading in other individual symbols.

6 Method of Analysis

In Section 4, we laid out a number of possibilities for a model of $M(S, t)$ as defined in Section 2. Below we list the parameters in the model that we need to choose:

1. The set of proxy symbols to draw from.
2. The number of rounds r to add proxy symbols and the number of symbols n to add per round.
3. Which symbols, if any, to require as proxies for all symbols.
4. Whether to fit an intercept, and if so, whether to include it in the model.

We want to understand how each of these parameters influences the metric defined in (4) when trained and tested on fresh data. We also want to understand how the performance of this model persists over time: Do the model parameters need to be updated after a certain amount of time? If so, how often?

A naive way to answer these questions would be to simply run a very large test that trains models for all reasonable combinations of all of these parameters, tests on fresh data, and then chooses the winner based on the loss function (4). This would make our model selection highly influenced by random characteristics of the particular training and testing sets we chose, and would give us little mechanistic understanding of what each of the parameter selections above actually mean.

Instead, we examined each parameter in the list above one-by-one, and over several different time periods, in order of what we believed to be their importance. Once a parameter was chosen, we held it fixed as we examined other parameters. In the end, we aimed to reduce the model complexity as much as possible without incurring a significant loss in performance.

We should note that this parameter-by-parameter selection method does not guarantee that our final result gives us the minimal value in (4). For that, we would need to perform a grid search. However, we do not have enough confidence that the structure of the loss function will be highly persistent over time and robust to noise in our data, and hence we prefer a more curated strategy.

For each test, we trained the model over a 60 day time period and tested on the following 60 day time period. We began with late 2019 (August 1 - September 29 train, September 30 - November 28 test), and examined other time periods later. We expected that the most important characteristic of a model would be the number of proxy symbols the model includes. For this reason, we first varied r and n as powers of 2 to examine the performance of the model with 1, 2, 4, 8, and 16 symbols, where symbols were collected in batches of 1, 2, and 4 whenever possible. We will compare these models to the model from [1] and the basic model that uses only SPY with coefficient 1 as a proxy. We will refer to the latter model as the "SPY Only" model. After analyzing these models, we will proceed to tune the other parameters and examine the model on different time periods.

We will begin with a modest proxy set- a list of the 50 ETFs with the highest average daily volume over the training and testing time period, and determine good choices for r and n. To begin, we do not include the $M(S, t')$ term or any required symbols, and we fit an intercept, but did not include it in the model. That is, we forced the intercept term to be 0 for generalization purposes when fitting to new data, which essentially amounted to centering the data before training. These options and the possibility of drawing symbols from larger proxy sets are reconsidered later.

We figured that the inclusion of the ETFs in the testing metric would be a bit misleading, since the model on these symbols is quite likely to perform well due to the mere similarity among them. Also, in light of the concerns raised in Section 3 the causal direction between ETF prices is much less clear than the direction of influence from ETFs to non-ETFs. For this reason, we will be excluding all ETFs from our metric calculations, as in [1].

7 Results and Analysis

Our first test on the batch size and fitting rounds used the top 50 ETFs by average daily volume for the list of potential proxies. The results, shown in Figure 1, indicate that the model does noticeably better than both the original model and the SPY Only model, but the the iterative symbol addition method leads to quick overfitting. In fact, the batch size n had much less influence on the generalization error than the number of rounds r did. The error measured is average square prediction error scaled by 10^6. From this point forward, we will refer to this average error on the training set as "train error" and this average error on the test set as "test error".

Here we see a direct inverse relationship between training and testing error, which is of course a sign of overfitting. Our simplest model–1 symbol, 1 batch–had the best generalization error out of all of the options we considered. While the model is simple, it constitutes as much of an improvement from the model in [1] that this model made on the basic "SPY Only" model.

Earlier on in the research process, we found more promising results when considering the top 2,000 symbols by average daily notional value as both the proxy set and the set of symbols to be tested. In the case presented above, we have reduced the proxy set to 50 ETFS to address the concerns in Section 3, which of course might exclude good predictors from the models. We also expect the behavior of symbols with less trading activity and lower prices to be more difficult to approximate through ETFs, so our model performance is likely to degrade due to our inclusion of all symbols in the testing set, and our exclusion of the top 50 ETFs from this proxy set.

To test these hypotheses, we reran the above test with a few parameters varying. We tried including the proxy set of symbols in the metric calculation, expanding the proxy set to all 685 ETFs from our database, and removing the bottom 2% and 5% of symbols by average daily notional value from the metric calculation. In each case, the shape of the plot analogous to Figure 1 was the same- the simplest models performed the best on the test data, and outperformed the old model by approximately the same degree that the old model improved from the SPY only model. The absolute scores did differ, significantly, as indicated in Table 1. All results have the set of 50 ETFs for the proxy set, unless otherwise specified.

Here we can see that the inclusion of the ETFs beyond the original 50 as proxies is not significantly

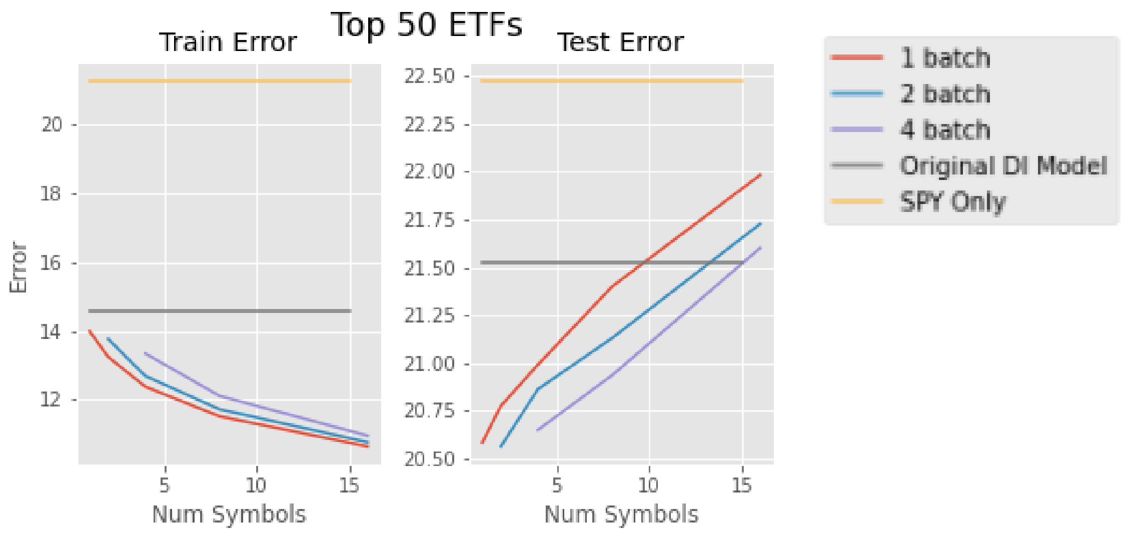

Figure 1: Proxy set size test on top 50 ETFs

Model	Test Error	% Improvement from SPY Only	% Improvement from [1]
50 ETFS	20.59	8.39	4.35
All ETFS	20.47	8.39	4.91
50 ETF Metric inclusion	16.88	11.52	5.7
All ETF Metric inclusion	16.78	12.05	6.26
2% NV drop	17.7	10.16	4.47
5% NV drop	15.18	11.85	5.14

Table 1: Variations on proxy sets and metric sets

helpful. The improvements from the baseline are almost exactly the same, and when including all symbols in the metric, the test error is again almost the same. As indicated in 3, we wanted to choose as small a proxy symbol set as possible, so the model with 50 ETFs fared better here in our minds. We can see also that the inclusion of the ETFs in the metric reduces the average error significantly, as does the removal of low notional value symbols. In addition, the improvement from the SPY Only model is larger in the cases where the metric includes ETFs or fewer low notional value symbols.

It is worth noting that the generalization when adding new symbols to the model actually became *worse* when increasing the size of the proxy set to all 685 ETFs. This plot is shown in Figure 2. This indicates that the correlations found with the additional ETFs were less persistent over time, and that we are best to stick with a smaller set of ETFs.

Next, we set out to determine the remaining parameters for the model. In particular, we still needed to determine whether to set a list of symbols as required proxies for all symbols and whether to fit/include an intercept. At this point, our benchmark test error to beat is 20.59.

Since the current model is so sparse, our first step was to add required symbols to the model to see if we can better account for the the $O(t)$ term in (4). By requiring all the ETFs in the model from [1] as a proxy in addition to the single symbol found in the process described above, the average test error slightly increased to 20.75. By including only SPY, the error increased to 20.67. Since we wanted to keep the model as simple as possible and there was no improvement, we did not consider requiring specific symbols as proxies from this point forward.

As for model intercepts, we tested three options. In the first, our baseline, we fit an intercept but did not include it in the model, and the test error was 20.59. In the second, we did not fit an intercept, and the test error increased slightly to 20.61. We then tried fitting an intercept and including it in the model, and the error increased slightly to 20.68. These are very small changes and perhaps not worth seriously considering

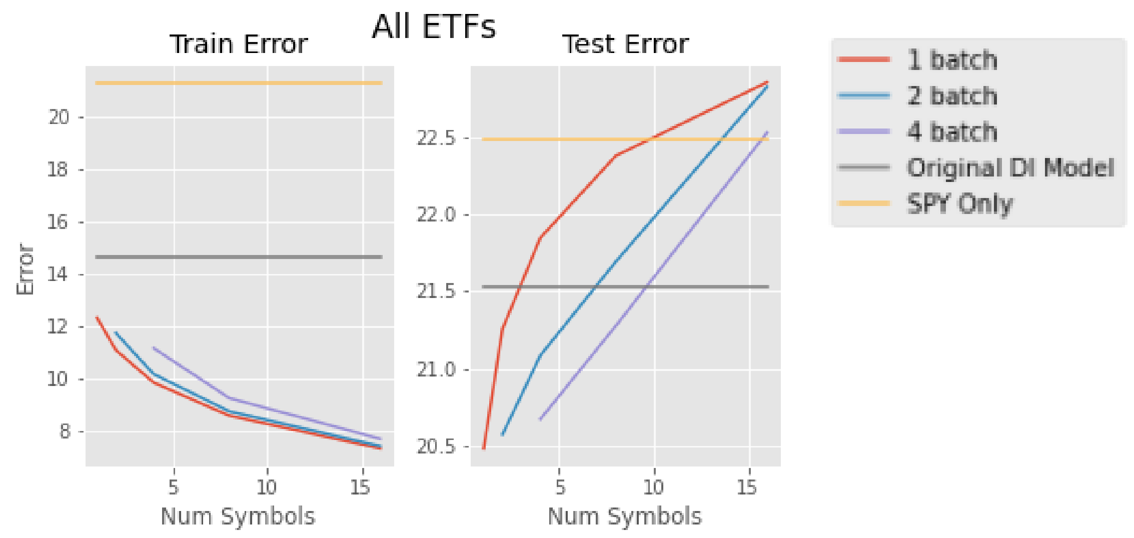

Figure 2: Proxy set size test on top 50 ETFs

when making our decision on the model. As described in Section 4, it is probably best to ignore any global directionality to a stock's price for this model. Because of this and the test error for the model that fit the intercept but did not include it was slightly better, we decided this was the best way to handle the intercept term. As a sanity check, we recreated the plot in Figure 1 with models that fit intercepts, and the general shape is the same, with small improvements in our model errors.

One weakness of the above analysis is that all tests were done on a single time period- early August to late November 2019. These were fairly typical market conditions, and the training data set (from August to late September) was very similar to the test set (late September to late November) in terms of overall market movements. We wanted to see if we achieved similar results when testing the model on different time periods.

We ran the above tests on two other time periods of the same length, one starting on January 1, 2020 and the other on January 1, 2021. We expected all the 2020 models to perform worse overall, since the market during the training period, which ends in early March, was very different from the market during the testing period. The 2021 period was not quite as volatile, so we expected each of the models to do comparatively better there. The results for the batch size test for 2020 and 2021 data are presented below, in Figures 3 and 4, respectively.

These results are promising. The shapes of the plots are the same, which confirms our choice of the 1-symbol model. The 2020 data is clearly more prone to over-fitting, but the simplest model still did quite well relative to our baseline models. In fact, the relative improvement of the 1-symbol model over the SPY-only model compared to the model from [1] actually was noticeably stronger in these cases. The results are presented in Table 2 below. Here "% Improvement" from a baseline score is the difference between our model's score and the baseline, normalized by the baseline.

Model Data Year	Test Error	% Improvement from SPY Only	% Improvement from [1]
2019	20.59	8.39	4.35
2020	82.75	11.06	8.93
2021	83.74	19.25	23.58

Table 2: Variations on time periods for training and testing

We also tested the effect of including required symbols and intercepts on this data, and came to the same conclusion: it is best to fit in intercept but not include it in the model, and not enforce any required symbols.

It is worth noting that the absolute errors in the early 2021 training data were significantly higher than any other time period, including the following time period corresponding to the testing data. This is likely

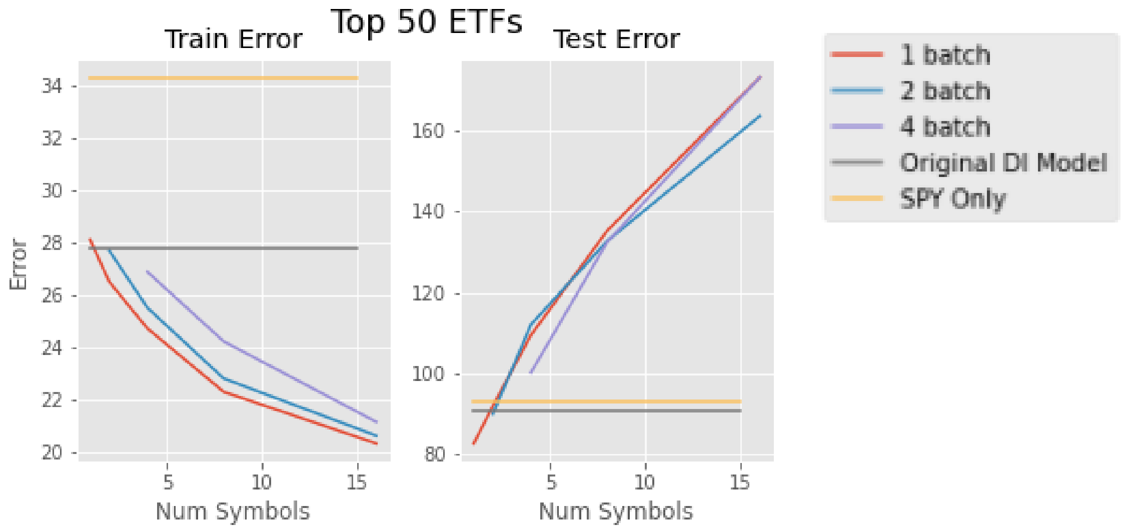

Figure 3: Proxy set size test on 2020 data

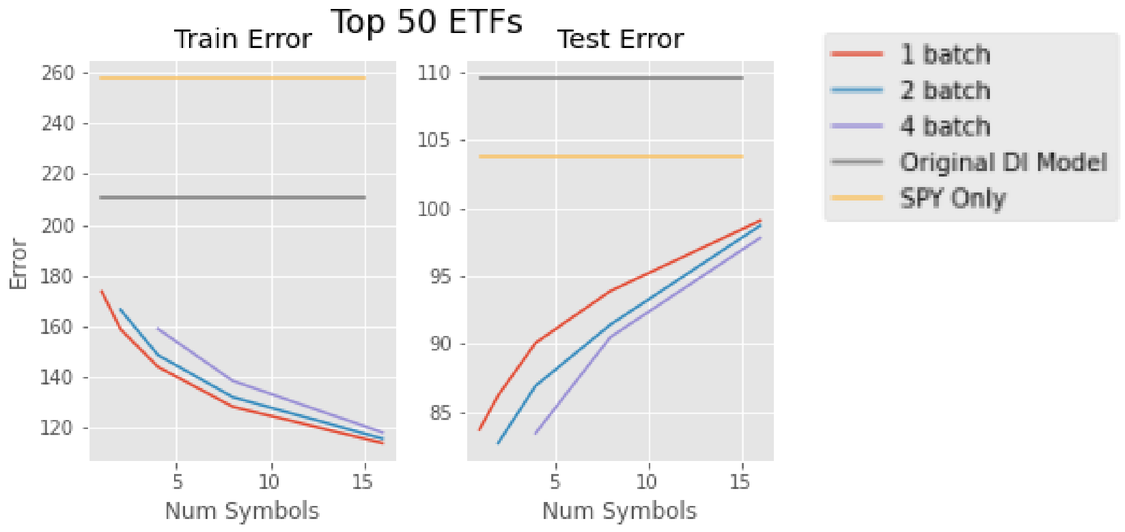

Figure 4: Proxy set size test on 2021 data

due to the fact that the (scaled) notional-weighted variance of $M(S,t)$ during this time period was 264.75. For comparison, the variance during the testing period was 118.10.

In order to determine the robustness of these results, we reran all of the above tests with some slight changes to the setup. First, we replaced the $M(S,t)$ data with corresponding data for the VWAP definition of $M(S,t)$, shown in Equation 3. Second, we changed the method of splitting the data between training and testing. Rather than training the model on data from a contiguous set of 60 days and testing on data from the following 60 days, the training data consisted of all data coming from even-numbered days of each month, and the test data consisted of data from odd-numbered days of the month. This way, we expected the two data sets would not be influenced by significantly distinct overall market trends. In both cases, the resulting data was similar, solidifying the conclusion that a single-symbol model with no intercept term was the best in this framework.

8 Least Angle Regression

The results of Section 7 show that we can improve on the model from [1] by customizing proxy sets per symbol. However, the iterative selection method we chose was not as effective as we would have liked. Our hope was that as we chose larger sets of predictors, the model would begin to both fit the data better and better until the model became too complex and reached a point of over-fitting. What we found was that we hit the point of over-fitting immediately. Without further investigation, we did not know if this was something inherently limiting about the data, or whether our symbol selection procedure was sub-optimal.

As stated in Section 4, the problem of carefully choosing a small set of linear predictors out of a wide selection of options is well-known to statisticians. Our case is a specific version of this, where all the predictors, or features, correspond to the price changes of ETFs. In an attempt to find a better symbol selection procedure, we considered a more complex general feature-selection procedure for high-dimensional linear models known as *least angle regression* (LARS) [3]. The LARS method starts similarly to our method by choosing the most correlated feature and adding it to the model, except with coefficient 0. Features are then iteratively added to the model as follows: Once n features and their coefficients have been selected, they are smoothly increased in their least squares direction until another feature not yet in the model is as correlated with the residual as the model predictions are correlated with the true values. That feature is then added to the model with coefficient 0.

The approach is clearly similar to ours, and we can choose as before the number of symbols/features to include in the model to test and see what the proper feature size is (this is more difficult with other subset selection methods like lasso regression). In addition, scikit-learn already has a LARS implementation that was well-suited for our purposes, so the implementation was straightforward.

We ran tests to find the best number of symbols to include in the model in a manner similar to those at the beginning of Section 7 with only the top 50 ETFs considered. This time we included our 1-symbol model from the previous section in our collection of baselines. In these tests, we only considered 1, 2, and 4 symbols per model. This was partially because again the test error values for these three symbol set sizes did not indicate that larger models were needed.

The results were not promising for least-angle regression. On each of the three time periods (late 2019, early 2020, and early 2021), the LARS model had a worse training score that our 1-symbol baseline model. The results from late 2019 are shown in Figure 5.

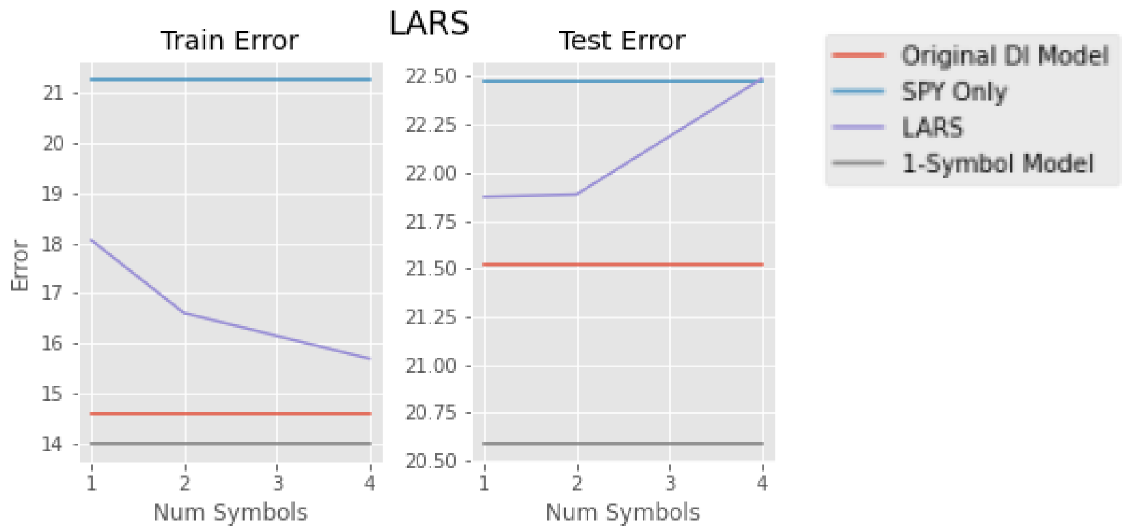

Figure 5: LARS test for late 2019 data

9 Conclusion and Future Steps

In this paper, we explored several options for proxy symbol selection for the distilled impact model. The first was a custom symbol selection method that iteratively selected symbols in batches based on their correlation with model residuals. These tests concluded that the simplest possible model in this framework performed the best under our evaluation metric. We considered some variations on this model, by requiring specified proxy symbols in the model and varying the use of model intercepts, ultimately finding that it was best to fit an intercept but not include it in the model, and not require any specific symbols in the proxy set.

To test the idea that our symbol selection procedure was the source of the poor performance of models with more symbols, we tried using least-angle regression models to find proxy symbol sets. This technique proved less effective than the simple model found previously.

This idea of automated proxy symbol set selection was essentially the simplest "next step" to take after the work in [1]. Our final model consistently improved upon the error in the model from [1] to a similar extent that the model from [1] improved upon its "SPY Only" baseline.

We expect it might be difficult to further improve the distilled impact model in this framework using only linear models with proxy symbol prices as features. To see further improvements, we may need to change one or more of the following: our potential feature set, our restriction of only linear models, or our framework for evaluating the models. The main challenge continues to be the level of noise inherent in the data. Any future improvements would almost certainly need a change in the feature set, but it might be difficult to find useful features since the simplest and most intuitively useful features (i.e. proxy symbol price changes) had such little mileage. Nonlinear models would be easy to test, but without good features, it is highly unlikely that these models will be successful. For now, we leave the search for further improvements as a task for our future research.

References

[1] Allison Bishop. *Distilled Impact.* Proof Trading: https://prooftrading.com/docs/distilled-impact.pdf.

[2] *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, by Trevor Hastie et al., Springer, 2017, pp. 57–93.

[3] Efron, Bradley, et al. "Least Angle Regression." *The Annals of Statistics*, vol. 32, no. 2, 2004, doi:10.1214/009053604000000067.