Predicting Execution Quality on the New York Stock Exchange

Greg Stepaniouk

December 4, 2017

1 Introduction

The financial markets have changed substantially over the years, and the days of the busy stock exchange floor have come to an end. Most trading now occurs electronically. Human buyers and sellers are matched up by complex algorithms that rapidly transact in an increasingly competitive environment where every nanosecond of latency has its impact on profit margins. According to a recent congressional research service report,[1] high frequency trading (HFT) makes up roughly 55% of US equity trading volume. Due to the competitive nature of the industry, the algorithms in use by electronic traders are proprietary and are effectively a black box to the average market participant. We can only observe the result of how markets are behaving.

When an investor sees a price and decides to place an order to buy or sell, there are two costs associated with the transaction. The first cost is deterministic, consisting of the commission paid to the broker (e.g. the $6.95 ETrade will charge to place a trade). The second cost is an implicit transaction cost known as the spread. When a market order is routed to an exchange, the true execution price will not always reflect the publicly quoted price the investor sees on her screen. The goal of this project is to predict the spread an investor can expect to pay when placing an order which is routed to the NYSE.

2 Data

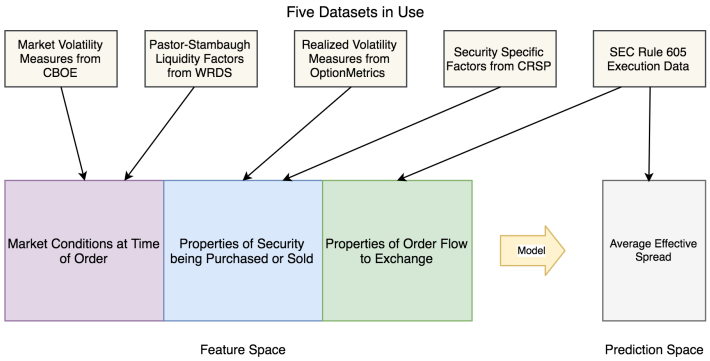
2.1 SEC Rule 605 Data

In 2005, the US Securities and Exchange Commission (SEC) adopted SEC Rule 605 which requires market centers to make available monthly reports in a standardized ASCII format. These reports include information about each market center's quality of executions on a stock-by-stock basis, including how market orders of various sizes are executed relative to the public quotes. These reports must also disclose information about effective spreads (i.e. the transaction costs paid by investors whose orders are routed to a particular market center). In addition, market centers must disclose the extent to which they provide executions at prices better than the public quotes to investors using limit orders.[2]

The New York Stock Exchange is the world's largest in terms of trading volume, so I choose to focus on the Rule 605 disclosures of NYSE ARCA, which is an electronic trading center matching buy and sell orders on behalf of the NYSE. Another important factor is that NYSE ARCA reports are available for several years, while other large market centers seem to only have a few months of disclosures. The availability of data from 2015 is needed to allow merging with the other datasets I selected.

2.2 Additional Datasets

The Rule 605 Data offers plenty of information about transactions on the exchange, but there are no details about the securities being traded or the market conditions at the time of the trades. To expand the feature space, I introduce four more data sets obtained from Wharton Research Data Services.



To assess overall market volatility, I use the VIX index, which is calculated based on the implied volatility of options on the S&P500. As a measure of overall market liquidity, I add the Pastor-Stambaugh Liquidity factors which are derived from their 2003 paper, [3] *Liquidity Risk and Expected Stock Returns*. These three factors are an aggregate liquidity measure (PSL), the innovation of that liquidity measure (PSI), and the premium associated with a portfolio constructed based on the liquidity measure (PSV).

To obtain a range of security-specific factors I use the Center for Securities Research (CRSP) dataset. The CRSP dataset contains daily data for a range of features, including pricing, volume, and shares outstanding. For daily security specific volatility, I use the OptionsMetrics dataset, which contains a 30 day realized volatility measure indexed by stock.

As a consequence of using several datasets, there is a limited period of overlap between all five. The model will need to be trained and tested on the first 8 months of 2015. Though this is a somewhat short period of time, the datasets are still very large. The daily datasets both contain approximately 2 million line items before processing, while a combined 8 months of NYSE ARCA Rule 605 disclosures is approximately 600,000 line items. Once completely merged, aggregated, and processed, the combined master dataset is 167,379 line items with 34 fields.

3 Merging and Processing Datasets

3.1 Monthly Averages and Missing Values

The Rule 605 data only comes in monthly intervals, while the OptionMetrics, CRSP, and VIX data is available at a daily resolution. In these datasets, the features of interest are all continuous, which makes it practical to take monthly averages for each stock. Like most financial data, these datasets have substantial amounts of missing data. In the CRSP dataset, complete cases make up less than 25% of my chosen feature space. As a result, when taking the monthly averages of features, I implement the following logic to ensure I have as many usable line items as possible.

- If there are at least 3 days in a month when the feature is not NA: Take the monthly average over the number of days available.
- Else: Set the monthly average to NA.

3.2 Merging

To facilitate the merge, I assign each line item in the security specific datasets a unique string identifier consisting of the stock ticker and the month. In the Rule 605 files, stock tickers and months are repeated because a specific ticker will generally have documented transactions every month. I index the security specific datasets to the sequence within the merged 605 files and combine everything into one DataFrame, with the addition of the monthly average VIX and Pastor-Stambaugh liquidity factors matched to the respective month of each transaction-level line item.

3.3 Features and Transformations

I choose my starting set of 14 features as follows:

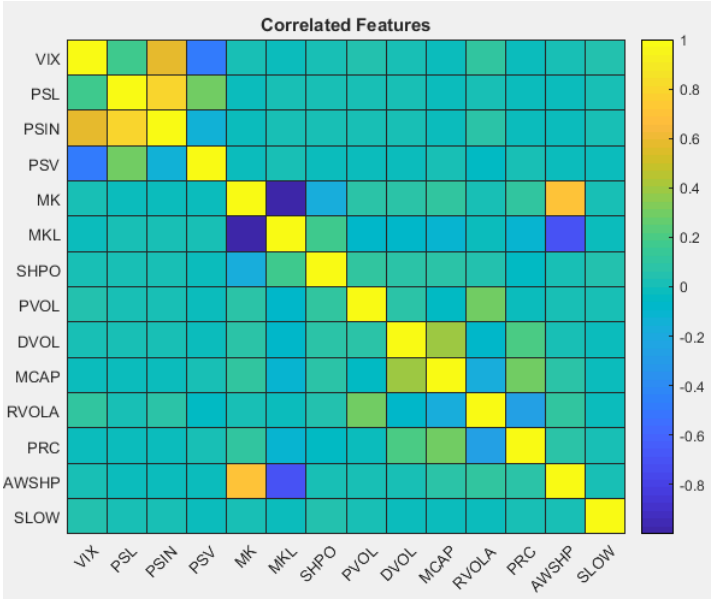
- Average Monthly Market Conditions
  - VIX - Average daily volatility index
  - PSL - Aggregate monthly liquidity
  - PSI - Monthly liquidity innovation
  - PSV - Monthly traded liquidity
- Average Monthly Stock-Specific Factors
  - DVOL - Average dollar volume
  - PVOL - Avg. share volume/shares outstanding
  - RVOLA - 30 day realized volatility
  - MCAP - Market capitalization
  - PRC - Average price
- Transaction Level Factors
  - MKT - One-Hot encoding for market order
  - MKL - One-Hot encoding for marketable limit order
  - SHPO - Shares per order
  - AWSH - Fraction shares executed away
  - SLOW - Fraction shares executed in over 10 Seconds

Most of these features are transformations of other features. For example, the share execution times are encoded in the original data as five individual columns. Over 94% of covered orders were executed in the first 10 seconds, and anything above 29 seconds represents less than 0.1% of the dataset. Instead of having extremely sparse features that only exist to emphasize outliers, I merge the 4 columns above 10 seconds into one proportional measure in the form of the SLOW field.

Most of the stock specific factors and market conditions are selected based on traditional empirical finance knowledge of liquidity and volatility being positively and negatively correlated with transaction costs respectively. The transaction level factors represent the entirety of the fields which I consider usable in the 605 data.

There exist fields such as total shares executed outside the quote, but using these figures would impact the predictive value of the model. An investor placing an order should not already know how many shares will be executed outside the quote. A similar argument could be made about the SLOW feature, but an investor can always cancel her order if she realizes 10 seconds are about to elapse.

All features are standardized as well as the Y output before fitting.I plot a heatmap of correlations within the feature space to see if any features are redundant. As expected, there are unit negative correlations between the one-hot encodings. There are some other strong correlations among the market climate factors, but I expect regularization to take care of these.

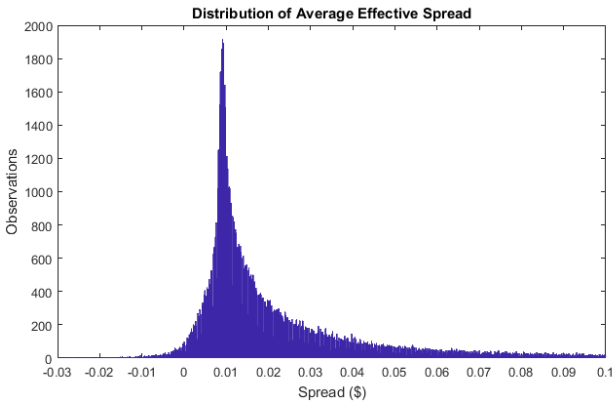


4 Linear Fitting

4.1 Prediction Space and Choice of Loss Function

The goal is to predict average effective spread, which is the difference and the midpoint of the publicly quoted na-

tional best bid or offer (NBBO), and the the average price per share at which investors had their orders executed. If the investor paid more than the NBBO midpoint to buy, or sold for less, the average effective spread will be positive. I plot a histogram of the AESP (before standardization) and run some summary statistics.



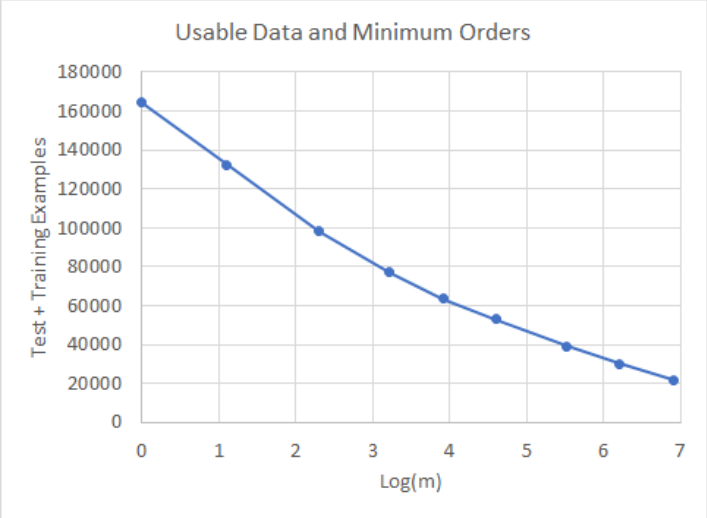
Basic Measures		Quantiles	
Max:	6.9953	5	0.0036
Min:	-9.8236	25	0.0091
SD:	0.08751	50	0.014
Mean:	0.028816	75	0.0295
Skew:	-0.87643	95	0.0954
Kurtosis:	2800.446	99.5	0.3192

The high kurtosis, variance, and heavy tailed distribution of the prediction space all points to the presence of many outliers. A robust regression is desireable in a situation like this, so I will choose the L1 loss as a loss function. The model will be much more stable if outlier errors are not quadratically penalized. An argument can also be made for the Huber loss, but with this particular dataset it is still computationally feasible to implement a non-differentiable loss function and use the proximal sub-gradient method.

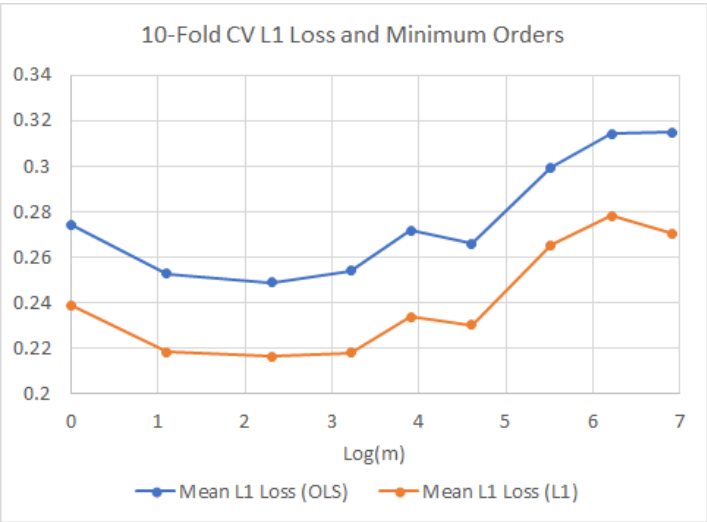
The L1 loss is also a logical performance measure in this case, as to an investor a transaction cost does not have quadratic implications. We would rather save more dollars on average than optimize to save dollars-squared. If we consider the model in the context of a trading firm, many trades are being made and outlier transaction costs will inevitably be negated by outlier savings over a large sample, so we do not want to focus on fitting these outliers with any sort of quadratic loss.

4.2 Minimum Covered Orders

The Rule 605 data is presented in buckets such that each line item represents a basket of orders for a particular month, order type, order size, and stock ticker. Statistics for a basket are given as averages of all orders matching the criteria. Financial data is riddled with outliers and noise, and any basket with only one or two orders is more likely to be noisy. There is a choice to be made for which baskets to consider. I want to keep the dataset as large as possible, but I also want to reduce the number of outliers. The following graph illustrates how the dataset shrinks as minimum orders per basket are increased.



If the dataset becomes too small, more complex models are unlikely to generalize well. If I leave the dataset too noisy, I am unlikely to obtain any sort of meaningful fit. Additionally, a smaller test set size will make it harder to make conclusions about the performance of the model. To make the decision of minimum orders per line item in a quantitative manner, I will refer to the minimum orders parameter as m and cross-validate over it. The following charts show the results for ordinary lease squares and L1 regression. The x-axis is log-scaled for interpretability.



We can see that initially L1 losses are higher, presumably due to excessive noise. The minimum is at m = 10, and as m increases from there we see an increasing mean L1 loss. This is likely due to the inability of the model to generalize in cross-validation as the dataset shrinks. Setting m at 10 is an attractive choice because the dataset still contains 98,268 usable examples, and our cross-validation error is minimized.

4.3 Sparse Fitting

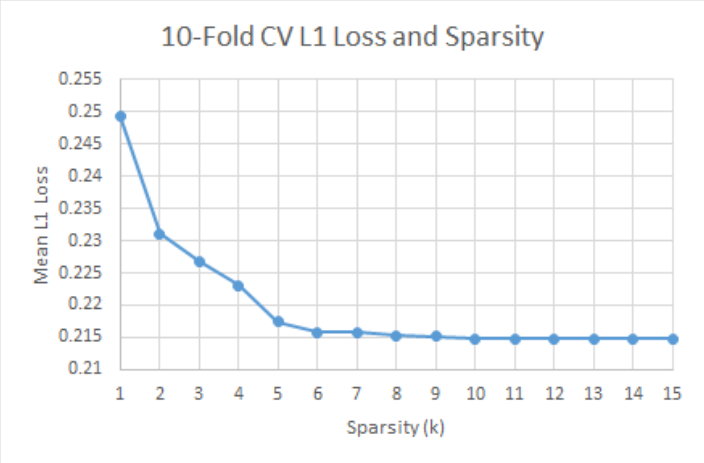
After trying out L1 and L2 regularization, I realized these regularizers were not impacting the model coefficients even at very high λ (i.e. $\lambda = 10{,}000$), and thereafter they only

seemed to be severely worsening the cross-validation error. In order to make the model generalize better, I opt to control coefficients via the more extreme k-sparse regularization. The goal is to minimize the function:

$$\sum_{i=1}^{n} |y - Xw| + 1_k(w) \tag{1}$$

$$1_k(w) = \begin{cases} \inf & \text{if } nnz(w) > k \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

I perform 10-Fold cross-validation over the degree of sparsity to determine which features are not significantly improving the mean L1 loss of the model. The following plot contains the resulting mean L1 cross-validation errors.



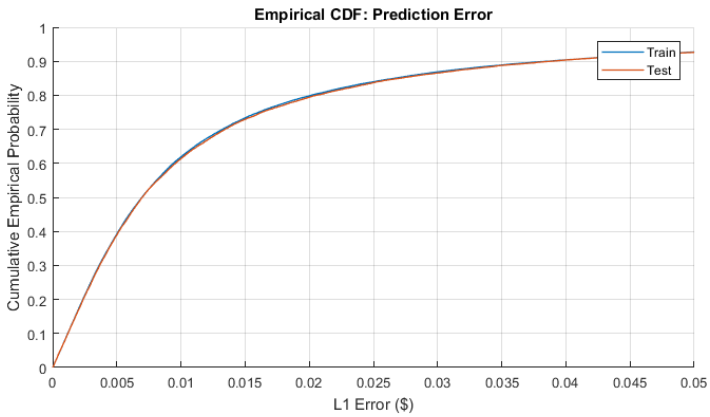
Clearly, once the model has 5-7 features, the marginal benefit of adding more is significantly diminished. Removing the unnecessary coefficients should help the model generalize much better on the test set. Across the degree of sparsity k, I chart the average w obtained by averaging the 10 coefficient vectors fit to each cross validation fold.

k	OFF	VIX	PSL	PSI	PSV	MKT	MKL	SHP	PVO	DVO	MCP	RVO	PRC	AWS	SLO
1	-.169														
2	-.125												.135		
3	-.125					-.035							.145		
4	-.119					-.038						.038	.169		
5	-.116					-.027					-.064	.034	.198		
6	-.114					-.027	.012	-.019			-.060	.036	.195		
7	-.114					-.019	.019	-.019			-.060	.036	.195		
8	-.114			.001		-.019	.019	-.018		-.023	-.053	.036	.198		
9	-.114			.006		-.019	.019	-.018		-.027	-.052	.036	.198		
10	-.113			.006		-.018	.018	-.017	-.010	-.023	-.055	.039	.200		
11	-.113	.005		.003		-.019	.019	-.017	-.010	-.023	-.055	.039	.200		
12	-.113	.004	-.002	.005		-.019	.019	-.017	-.010	-.023	-.055	.039	.200		
13	-.113	.004	-.005	.008	.003	-.019	.019	-.017	-.010	-.022	-.055	.039	.200		
14	-.114	.004	-.005	.008	.003	-.018	.018	-.017	-.010	-.023	-.055	.040	.201	-.003	
15	-.114	.004	-.005	.008	.003	-.018	.018	-.017	-.010	-.023	-.055	.040	.201	-.003	.000

Interestingly, at k = 6 and k = 7, both vectors have 7 nonzero coefficients, which means that at least one of the cross-validation folds at k = 6 had a different nonzero mapping than the others. The figure above suggests k = 5 or k = 6, but I choose to allow k = 7 sparsity for the final model because of how the average w vector itself behaved in cross-validation.

4.4 Test Set Performance

The training set mean L1 loss of the 7-sparse model is 0.2168. For the test set, this figure is 0.2179. Converted from standard deviations to dollars (unstandardized), the mean test error is \$0.0191. The standard deviation of the unstandardized average effective spread is \$0.0875 and the mean is \$0.0288. I plot the empirical CDF of the unstandardized prediction error.



As can be seen from the plot, 62% of the absolute errors are below \$0.01 and 80% are below \$0.02. The model appears to have generalized quite well to the test set, with the empirical densities of the errors matching almost exactly.

5 Fitting Regression Tree Models

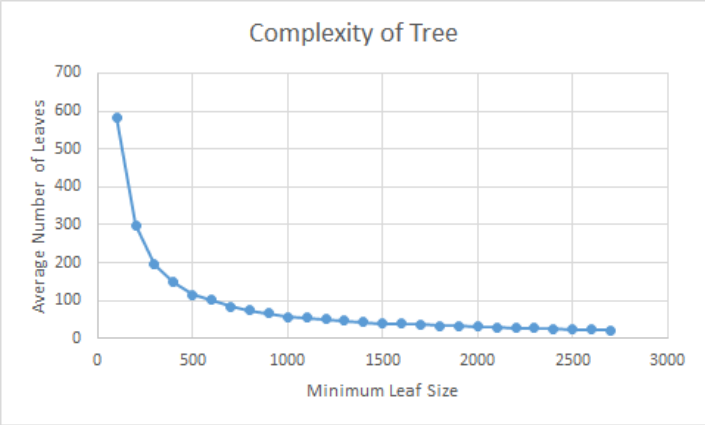
5.1 Motivation

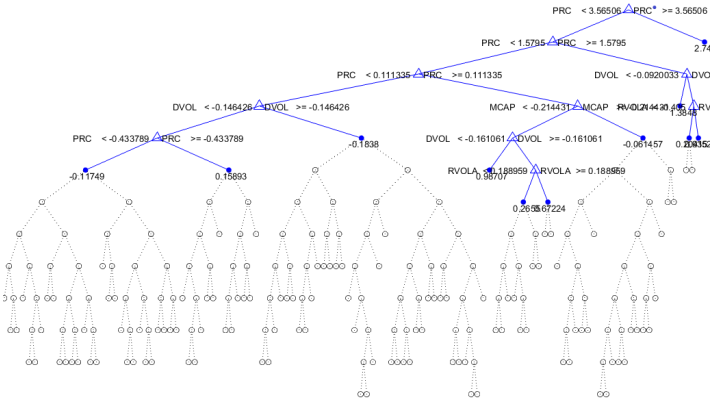
Although the linear model fits with reasonable quality, it clearly suffers from high bias and low variance. The unavoidable error due to noise is also likely to be a significant factor, but based on how closely the test and training set error distributions match up, I suspect this model is underfitting. The errors in the sparsity cross-validation indicate that there may not be too much room for improvement by adding or transforming more features.

A regression-tree based model may be able to capture relationships between the features and predictor space which cannot be fully described by a linear model.

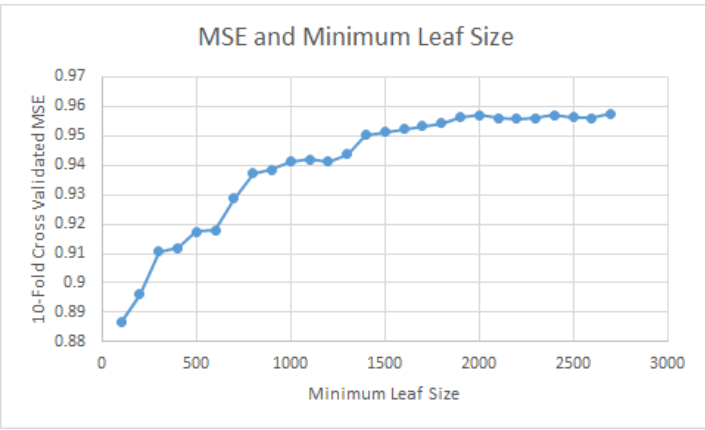
5.2 Determining Leaf Sizes

Before moving to the random forest, I start with a single tree. I control the complexity of the tree by setting the minimal leaf size. This is the minimum number of examples allowed to be stored in the terminal leaves of the tree. If the recursive binary splitting algorithm determines an optimal split which would push the resulting leaves below the minimum, the fitting is over. I plot average number of leaves in a set of 10-Fold cross-validated trees for a range of minimum leaf sizes. The average number of leaves serves to represent the complexity of the tree.





The complexity of the tree grows exponentially with decreasing minimum leaf size, but appears roughly linear in the region greater than 1000 examples per leaf. I also plot 10-Fold cross-validated MSE against minimum leaf size to see how the complexity-error trade-off behaves.



I choose a minimum leaf size of 500 for my single tree model. Although the cross-validated MSE continues to drop at a similar rate with MinLeafSize, MinLeafSize = 500 is the point at which the complexity starts to increase the most rapidly. Since I am fitting a single tree and not an ensemble, avoiding extreme complexity is necessary to prevent overfitting. I will also fit a tree with MinLeafSize = 50 to verify whether the complexity will cause a lack of generalization on the test set.

5.3 Single Tree Performance

The single tree with MinLeafSize = 500 has mean L1 error of 0.1920 on the training set and 0.1949 on the test set. Our most basic tree has generalized very well and outperformed the linear model. The test set error of this tree is lower than the *training* set error obtained by the unregularized L1 linear model.

The MinLeafSize = 50 tree has a mean L1 loss of 0.1573 on the training set, and a test error almost exactly the same as the MinLeafSize = 50 tree: 0.1947. As predicted during cross-validation, this outcome is a clear case of overfitting.

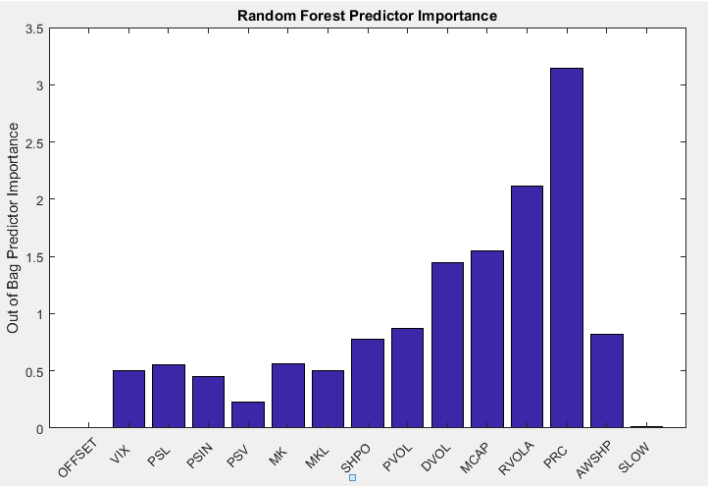
A look at our MinLeafSize = 500 tree shows that some of the earliest splits are on dollar volume, a variable which is completely ignored by the sparse regression until k = 8. Because the recursive binary splitting algorithm is greedy and chooses features and splits which reduce MSE by the most first, this may imply some sort of significance for this feature which the linear model overlooked.

5.4 Random Forest Implementation

The next step to add complexity while minimizing impact on generalization is implementing a random forest. The random forest will consist of 100 regression trees, with each tree training on a random subset of features and examples. The count of features to sample is the integer $m/3$ where m is the number of features. The prediction of the model on new data is the average of the predictions of the 100 trees.

I use MinLeafSize = 50 for trees in the random forest. This is not necessarily optimal, but I am faced with computational restrictions. Even training in parallel on 8 core laptop with 16 gigabytes of RAM takes almost 20 minutes. As seen in the complexity graph in Section 5.2, even a modest decrease in MinLeafSize makes the trees exponentially more complex and take even longer to train.

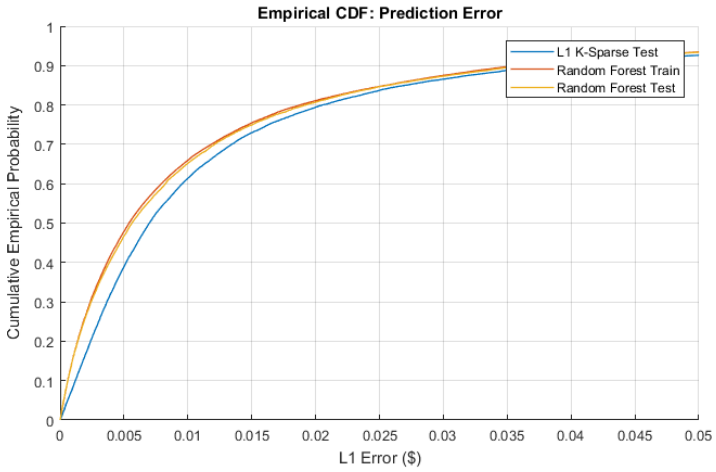
5.5 Random Forest Predictor Importance



It is possible to determine the importance of predictors to the random forest by using a permutation test.[4] This is done by measuring the mean decrease in prediction accuracy when a the order of a predictor is randomized. We can see that the 7 most important variables for the random forest are price, realized volatility, market cap, dollar volume, shares per order, proportion of shares executed away, and MKT (the one-hot encoding indicating a market order). For the 7-sparse linear model, the 7 nonzero coefficients are price, realized volatility, market cap, shares per order, MKT, MKL, and offset. The two features that stand out as different are dollar volume and shares executed away from market center.

5.6 Random Forest Performance

The random forest achieves a training set mean L1 loss of 0.1493 and a mean test set error of 0.1586. This is the strongest model by far, and the test set error only exceeds the training error by 6.23%, which means our random forest has generalized well despite its complexity. The test set error corresponds to a true unstandardized mean L1 loss of \$0.0139. I compare the empirical CDF of the random forest L1 errors to those of the 7-sparse linear model.



We can see that the random forest outperforms the k-sparse model, especially in the early stages of the curve. The random forest makes a prediction that is within \$0.005 of the true effective spread 47% of the time, and one that is within \$0.01 66% of the time.

6 Conclusion

In summary, using a big, messy combination of datasets, I have iterated through several model classes in an effort to create predictive value. Working with financial data is never easy, and it is not uncommon for the variance to be so high that only the most basic classes of model will generalize. I think that by combining multiple datasets and preprocessing my data extensively, as well as cross-validating to address uncertainty about parameters, I have created a usable model to obtain effective spread predictions. A few cents less error between model classes may seem insignificant, but if an order of 1 million shares is being placed, every cent matters.

SUMMARY			
MODEL/ERROR	Training Error	Test Error	Test Error (\$)
Least Squares	0.249	0.250	0.0219
L1 NoReg	0.216	0.217	0.0190
L1 7-Sparse	0.217	0.218	0.0191
Single Tree (50)	0.157	0.195	0.0171
Single Tree (500)	0.192	0.195	0.0170
Random Forest	0.149	0.159	0.0139

7 Bibliography

1. High Frequency Trading: Overview of Recent Developments https://fas.org/sgp/crs/misc/R44443.pdf
2. SEC [Release No. 34-43590; File No. S7-16-00] https://www.sec.gov/rules/final/34-43590.htm#seciiib
3. Liquidity Risk and Expected Stock Returns, Ľuboš Pástor and Robert F. Stambaugh, Journal of Political Economy 2003 111:3, 642-685
4. DECISION TREES, Lior Rokach, Department of Industrial Engineering, Tel-Aviv University http://www.ise.bgu.ac.il/faculty/li