Uncovering Feature Interdependencies in Complex Systems with Non-Greedy Random Forests

Delilah Donick, Sandro Claudio Lera\textsuperscript{a,b,}\textsuperscript{*}

\textsuperscript{a}Southern University of Science and Technology, Shenzhen, China
\textsuperscript{b}Massachusetts Institute of Technology, Cambridge, USA

Abstract

A “non-greedy” variation of the random forest algorithm is presented to better uncover feature interdependencies inherent in complex systems. Conventionally, random forests are built from “greedy” decision trees which each consider only one split at a time during their construction. In contrast, the decision trees included in this random forest algorithm each consider three split nodes simultaneously in tiers of depth two. It is demonstrated on synthetic data and bitcoin price time series that the non-greedy version significantly outperforms the greedy one if certain non-linear relationships between feature-pairs are present. In particular, both greedy and a non-greedy random forests are trained to predict the signs of daily bitcoin returns and backtest a long-short trading strategy. The better performance of the non-greedy algorithm is explained by the presence of “XOR-like” relationships between long-term and short-term technical indicators. When no such relationships exist, performance is similar. Given its enhanced ability to understand the feature-interdependencies present in complex systems, this non-greedy extension should become a standard method in the toolkit of data scientists.

The random forest (RF) has become a popular algorithm in data science [1]. RFs are comprised of decision trees (DTs), which are structures used to categorize and sort data. These DTs are conventionally constructed in a “greedy” fashion. In this article, we show that, in certain contexts particularly amenable to complex systems, it is advantageous to replace these greedy DTs with “non-greedy” alternatives. To recognize this, note that complex systems are characterized by the fact that “the whole is more than the sum of its parts” [2]. Accordingly, the non-greedy examination of multiple parts, or features, in concert, rather than the greedy examination of each part, or feature, in silo, may produce more accurate predictions. The results included in this paper suggest that RFs built from non-greedy decision trees (NGRFs) do indeed uncover feature relationships that are overlooked by RFs built from greedy decision trees (GRFs).

We start by reviewing the construction of GRFs, with an emphasis on binary classification. This is subsequently extended to a NGRF implementation that recursively adds non-greedy layers as subtrees of depth 2. We exemplify XOR-type patterns in financial time series with aid of synthetic data on which we train GRF and NGRF binary classification models. Then, we shift our focus from synthetic data to bitcoin, since financial markets are a hallmark example of complex systems [3, 4]. Both synthetic data and actual bitcoin prices demonstrate that when XOR-type patterns are present, NGRF models perform better at (binary) classification tasks. These results support the aptness of NGRFs for uncovering intricate

\textsuperscript{*}Corresponding author

Email address: slera@mit.edu (Sandro Claudio Lera)
feature relationships in financial markets. This suggests that the NGRF is a simple but effective extension of the more common greedy random forest with wide areas of applicability.

**Random Forests with Greedy Decision Trees**

In this section, we provide an overview of how vanilla GRFs and their underlying greedy DTs are built. For convenience, we limit our focus to binary classification. Classification for a larger number of classes and regression variations are similar [5]. For the remainder of this article, we label the two classes as “+” and “−.” In the context of price-movement and the included examples, “+” represents a future upward movement in price and “−” represents a future downward movement in price. The reader familiar with GRFs can skip the remainder of this section.

A RF is trained on a training dataset of \( N \) samples, \( \{X_i\}_{i=1}^{N} \), where the datapoint \( X_i \) is a vector of length \( k \), representing the \( k \) different features that the algorithm may use for classification. To each datapoint in \( X_i \) corresponds a target value \( y_i \) (either “+” or “−”) used to train the algorithm. The RF is made up of a pre-specified number of \( T \) decision trees that recursively split training samples with the objective of disentangling the two classes. There are two important characteristics of a RF that ensure the creation of heterogeneous DTs. First, for each of the \( T \) trees, we sample \( N \) datapoints from \( \{X_i\}_{i=1}^{N} \) with replacement, such that each tree trains on a slightly different version of the data, still of size \( N \). Second, at any split point, only a randomly selected subset of \( \tilde{k} \leq k \) features is considered for the split. Common choices are \( \tilde{k} = \sqrt{k} \) or \( \tilde{k} = \log_2 k \).

Conventionally, DTs are built through recursive binary splitting, often referred to as a greedy approach. To start, the randomly selected training samples are initially contained in the “root node” at the top of the DT, as can be seen in Figure 1.

From the root node a split forms two tree branches and divides the samples into two child nodes. This split is defined by one of the sample features and a specific value for that feature. The samples that were originally in the root node are separated based on whether their values for that feature are greater or less than the indicated split-value. The root node chooses this feature and value to minimize the classification impurity of the samples that will be divided into each of its two separate child nodes. We use the Gini-score to measure classification impurity. Using other impurity measurements like entropy is similar. For binary classification, with the classes “+” and “−,” the Gini-score, \( G \), of a node is defined as:

\[
G = 2P_+(1 - P_+) = 2P_-(1 - P_-)
\]

(1)

where \( P_+ \) and \( P_- \) are the respective proportions of “+” and “−” observations present in the given node,

\[
P_+ = 1 - P_- = \frac{\text{num."+" samples in node}}{\text{total num. samples in node}}.
\]

(2)

The DT construction is described as greedy because each split is determined by optimizing the classification of the immediate split without taking into consideration the implications for future splits. That is to say, a greedy approach dictates that for a parent split-node \( S_0 \), one feature is selected from the available \( \tilde{k} \) features, with a corresponding split-value \( V_0 \) so that the sample-weighted average Gini-score \( G_W \) is minimized. \( G_W \) is defined as

\[
G_W = \frac{n_1G_1 + n_2G_2}{n_1 + n_2}
\]

(3)
For binary classification tasks, the sample is classified as “+” if \( P_+ > 0.5 \) and as “−” otherwise. These RF predictions are the basis for the trading strategies presented below.
Random Forest with Non-Greedy Decision Trees

In this section we discuss the construction of the non-greedy DTs. As discussed above, a near-sighted, greedy approach to DT construction only focuses on the two immediate child nodes to determine a split. In contrast, non-greedy DTs are constructed by considering multiple or all split nodes simultaneously. There can be two drawbacks to using non-greedy DTs over greedy alternatives. The computational complexity of training non-greedy DTs grows exponentially with number of nodes, as opposed to linearly for greedy ones. Non-greedy DTs also possess a higher risk of overfitting, which is of particular concern when working with financial data where a low signal-to-noise ratio is the norm.

Previous approaches to NGRF construction have solved the non-convex, computationally intensive optimization problem with (approximate) methods like iterative linear programming [6], or stochastic gradient descent [7]. Because both the computational cost of non-greediness and the risk of overfitting can be mitigated by limiting DT depth, we focus on constructing non-greedy DTs of depth 2. Since this means we only optimize across three split nodes at a time, a brute force optimization remains feasible. The DTs are not limited to depth 2 but can be further expanded to greater (even-sized) depths, by recursively appending non-greedy subtrees of depth 2, similar to how the greedy DT appends individual nodes.

Each non-greedy DT (or subtree) of depth 2 has the structure shown in Figure 1. It consists of 3 split-nodes that are jointly determined. As in greedy construction, each split node only sees a random subset of $\tilde{k} \leq k$ features, giving rise to at most $\tilde{k}^3$ different feature combinations (or less due to symmetries). To determine the split-value of any given feature, we discretize the feature’s value range into $B \leq N$ equal-sized buckets based on the sample quantiles to avoid susceptibility to outliers. The larger $B$ is, the more granular the resolution of potential splits, with an upper bound at $B = N$ (the case in which each feature value is unique and considered as a potential split point). (In subsequent sections, we have set $B = 30$ to mitigate the chances of overfitting and reduce computational complexity. Varying this parameter did not change the nature of our results and even setting it up to $B = N$ is feasible.) Even for applications where $k$ reaches the hundreds, and $N$ is in the thousands, the evaluation of all $B \tilde{k}^3$ possible DT structures, we select the one that minimizes the cumulative sample-weighted Gini-score

$$G_C = \sum_{i=0}^{i=3} n_i G_i$$

where $n_i$ is the number of samples and $G_i$ is the Gini-score (1) of leaf $L_i$. If the maximum depth of the DT is greater than 2, the above process is repeated recursively to grow subtrees in steps of depth 2 until a stopping criteria is met (not enough datapoints in a leaf, all datapoints in a leaf belong to the same class, etc.). Apart from the construction of its decision trees, a NGRF functions the same way a GRF does. It likewise makes classification predictions by averaging the probabilistic class predictions across its DTs. In the next two sections we will train GRFs and NGRFs on synthetic and real data and evaluate their relative performances.

**XOR as Synthetic Complex Data Patterns**

In this section we look at synthetic data patterns that highlight the advantages of the presented NGRF variation over a traditional GRF.
An “exclusive disjunction” (XOR) is a logical operation that returns True when its two inputs differ (one True, one False) and False when they are the same (both True or both False). A stylized XOR-like feature-interdependency between two features $F_0$ and $F_1$ is depicted in Figure 1 for 100 sample points. Both $F_0$ and $F_1$ have values that range from 0 to 1. All samples which have both $F_0, F_1 \leq 0.5$ belong to the “−” class and otherwise belong to the “+” class.

Such XOR-type relationships can be important in the description of economic and financial data. For example, consider the relationship between a country’s possible fiscal policies and the resulting stability and resiliency of its economy. There are two possible sets of fiscal rules that can be imposed: A) strict debt and budget balance rules, and B) strict expenditure and revenue rules. To achieve a stable economic regime, a country can implement either A or B, but not both. If both A and B are implemented, the excessive fiscal regulation can impede a country’s ability to react to economic turbulence and will result in an unstable economic regime [8]. Inference of such and similar interdependencies directly from data is vital for modern risk modeling. In the next section we will see another instance of an XOR-relationship between a short-term and a long-term technical indicator.

Ideally, a DT of depth 2 would recognize the structure of the data depicted in Figure 1. It should determine that to best separate the data into two classes, $F_0$ should have a split value $V_0 = 0.5$ and $F_1$ should have a split value $V_1 = 0.5$. However, if the DT attempts to determine the best split value for each feature in isolation of the other, it will likely not arrive at this result. For example, as shown in Figure 1, the DT could split on $F_0$ at $V_0 = 0.25$, $V_0 = 0.5$, or $V_0 = 0.8$. In all these cases, the ratio of “+” samples to “−” samples, and hence the Gini-score (3), remains (on average) the same within both of the distinct regions. This would likewise be true if the DT were to first split on $F_1$. Myopically, the greedy DT discovers no significant classification advantage by splitting at any particular split value for either $F_0$ or $F_1$ and consequently selects a globally suboptimal split.

We test the performance of the GRF and NGRF by generating 2,000 XOR-like datapoints à la Figure 1, split 80%/20% into training and test data respectively. For the remainder of this paper we use Scikit Learn’s RandomForestClassifier and underlying DecisionTreeClassifier classes [9] as the benchmark GRF implementation. The targets $\{y\}$ are encoded by setting “+” to 1 and “−” to 0. Next, we use this training dataset to train GRF and NGRF models, which are each made up of 100 DTs of depth 2. Once trained, we use these two models to make classification predictions for the samples contained in the test dataset. For the test set, the GRF and NGRF models achieve prediction accuracies of 87.5% and 98% respectively. The NGRF achieves its greater classification accuracy through its ability to consider $F_0$ and $F_1$ simultaneously. With 100 decision trees, the GRF still demonstrates proficient classification. However, this is primarily true because the ideal split value of 0.5 is also coincidentally the average expected value of numbers randomly selected from between 0 and 1. Therefore, as the number of DTs in the RF increases, the average of randomly selected split values for both $F_0$ and $F_1$ converge towards 0.5. It is worth acknowledging that a GRF model may demonstrate a higher prediction accuracy at greater depths. In the interest of consistency, we have chosen to use a maximum depth of 2 for both RF models. In the next section we will also consider RF models of depth 4 when applied to bitcoin.

In practical applications of financial engineering, one often considers fundamental, technical, and macroeconomic features in combination. The previous example of XOR-type data has only two features and a clear structure that enables a highly accurate classification of the dataset. To make the data more realistic, we again consider the same 2,000 samples from above, but now include eight additional features $F_2, \ldots, F_9$ that are (Pearson) correlated with the target at 25%. These features act as noisier features that obfuscate the still highly structured XOR-type pattern. We set $k = \sqrt{k}$ so that each split node is only able to consider a random subset of 3 of the $k = 10$ possible features. As in the previous example, we split
this data and train GRF and NGRF models, each with 100 decision trees of depth 2. The GRF model prediction accuracy drops to 62.2% while the NGRF model prediction accuracy remains comparatively-high at 88.2%. Because $\tilde{k} \approx 3$, it is inevitable that many DTs do not contain both of the XOR-features $F_0$ and $F_1$ and will be forced to split on one of the available less predictive features. Accordingly, compared to the previous example, the classification performances of both the GRF and NGRF decrease. Greedy-trees are misled by the 25% correlation of the features $F_2, \ldots, F_9$ and more frequently disregard the features $F_1$ and $F_2$ upon their first split. Because the non-greedy DTs are able to better recognize the benefits of splitting on $F_1$ and $F_2$ in combination, the NGRF’s classification performance suffers relatively less.

Finally, we consider synthetic dataset samples with the XOR-type relationship encoded $F_0$ and $F_1$ removed. The 2,000 samples are now composed of only 8 features $F_2, \ldots, F_9$, all of which are correlated with the target at 25%. We once again set $\tilde{k} = \sqrt{k} \approx 3$ and train GRF and NGRF models, each with 100 decision trees of depth 2. Upon running multiple iterations of the above simulation, we find that both the resulting GRF and NGRF models consistently demonstrate almost equivalent prediction accuracies that normally range from 65-75%. A binomial test with a $p$-value well below 1% confirms that neither the GRF nor the NGRF models consistently outperforms the other.

The XOR-inspired data explored in this section are simplified examples of the complex patterns that exist in financial data. In both cases, by construction, the NGRF is able to better capture these important feature relationships and consequently make more accurate classification predictions. At the same time, when the dataset is stripped of any XOR-type, or otherwise definitive feature relationships, the GRF and NGRF models achieve approximately equivalent prediction accuracies. Next, we extend these findings to a financial time series to uncover and isolate feature interdependencies that help improve classification of price returns.

**Trading Bitcoin With Random Forests**

Consider the following trading strategy based on a binary RF classifier that predicts the sign of an asset’s daily price returns. Prior to each (out of sample) trading day, all 100 DTs of the trained RF classifier return a binary prediction - whether the following day’s price return will be positive or negative. If, for instance, 60 out of the 100 DTs predict that tomorrow’s return will be positive, that translates to a signal strength of 60%, and so forth. For the following trading day, we are long/short the asset if the signal is $\theta\%$ above/below the neutral 50%. If it is not, we take no position for that trading day. The threshold $\theta$ is a strategy meta-parameter whose possible values are 1%, 2.5%, 4%, and 5%. There are two RF meta-parameters: maximum DT depths of 2 and 4, and minimum percentages of the total samples required in a leaf of 2%, and 10%. In total, there are thus $4 \times 2 \times 2 = 16$ different combinations of meta-parameters. Of those, we select the one with the highest Sharpe ratio on reserved cross-validation data. (Strategies must trade at least 20% of all eligible trading days to be considered. This may effectively remove some of the 16 combinations ex-ante.)

To extend a trading strategy over longer time periods, we repeat the above process in rolling windows. Here, we consider rolling windows that consist of 500 trading days and are rolled in steps of 75 trading days. For a given window, we denote $t_0$ as the first trading day of the corresponding out-of-sample prediction dataset. Binary RF classifiers are trained on the previous 500 days, $t_{-500}$ through $t_{-1}$, of which 50 data-points, $t_{-275}$ through $t_{-226}$, have been reserved for cross-validation. Because we train on single day price returns rather than overlapping multiple day returns, we omit embargo zones [10]. This rolling window structure is depicted in Figure 2.
Figure 2: The top plot includes the cumulative returns from April 9, 2016 through October 5, 2019 for the different trading strategies. The dark grey line represents the benchmark “buy and hold” strategy for bitcoin. The solid and dashed lines plot the returns of the optimized GRF and NGRF-based trading strategies trained upon ten and two features, respectively. The legend shows the average cumulative annual growth rate (CAGR), annualized Sharpe ratio, success rate (SR), maximum drawdown (MDD), and the fraction of time that a long (L) or short (S) positions is held. The NGRF-based trading strategies outperform their greedy counterparts across various metrics. The shaded background colors depict one of the rolling windows for in-sample (IS) training, cross-validation (CV) and out-of-sample (OS) prediction. These windows are rolled in steps of 75 (bitcoin trading) days. The bottom right heatmap visualizes the “probabilistic” classification (fraction of DTs that predict at positive return) of a NGRF trained on just two features (cf. axis labels). This structure is akin to the XOR-pattern of Figure 1.

This trading strategy framework is now applied to trading bitcoin (BTC/USD). Bitcoin was not selected at random, but rather through a careful process (detailed in Appendix A) with cognizance of the common overfitting trap of “trying numerous strategies until one works.” We have used eight basic technical indicators derived from the asset’s open, high, low, close and volume price data, as defined in Appendix B. We also assume instantaneous trade execution, as we initiate trades based upon close prices that are also included in the technical indicators used for the signal generation. We do not account for slippage or trading costs. Because we are focused on relative rather than absolute performance of the two types of RFs, these simplifications do not affect the analysis presented in this section. Our objective is to evaluate the relative effectiveness of this paper’s NGRF implementation compared to a GRF, not to build
out viable trading strategies.

From April 9, 2016 through October 5, 2019, the benchmark 100% long "buy & hold" strategy has an annualized Sharpe ratio of 1.27 and an average annualized return of 133%. Over that same period, the GRF-based strategy achieves an annualized Sharpe ratio of 0.69 and an average annualized return of 35.5%. The NGRF outperforms both of these benchmarks with an annualized Sharpe ratio of 1.74 and an average annualized return of 184% while staying out of the market more frequently. More performance measures are summarized in Figure 2.

Given the magnitude of price speculation and debate regarding the fundamental value of bitcoin, it is not surprising that the basic price and volume indicators considered here are able to provide some explanatory value of its price movement. To better understand the NGRF outperformance, we investigate the feature combinations that best explain the classification of the dataset. Feature importance practices play an important role in any thorough machine-learning model development process. Although more statistically rigorous methods exist [11], here we start with a basic feature count. Upon examination of the DTs used in the NGRF-based strategy, we identify the five most frequently selected feature combinations that are used by the nodes to split the data. For each of these five feature combinations, we train a single non-greedy DT on all of the available data and visualize its probabilistic prediction using heat-maps. We only consider the first batch of in-sample data to avoid a subsequent lookahead bias.

Amongst these, we find that the 5-day sign-correlation and the 20-day volume-based Z-score most demonstrate an XOR-like relationship, as is shown in the heatmap included in Figure 2. We can use this heatmap to attempt to understand how these features interact in a manner predictive of price movement. The 5-day sign-correlation indicates the consistency of an asset’s price movement over the short-term. The 20-day volume-based Z-score indicates whether present trading volume is low or high relative to its longer-term levels. When the 5-day sign-correlation is greatest and the 20-day volume-based Z-score is lowest, the probabilistic predictions are generally indicative of a price increase (the green lower-right ‘quadrant’ of the heatmap). There exist multiple possible interpretations of this relationship, such as the following. When short term daily price movements correlations are higher, it leads to more certainty about market direction. At the same time, if trading volume tends to mean-revert, and is currently relatively low, it is expected to increase in the near future. Together, this increase in trading volume, coupled with confidence amongst market participants suggests a price appreciation. Deeper examination of this feature relationship is beyond the scope of this paper. The key takeaway is that the NGRF is able to uncover such feature interdependencies.

The 5-day sign-correlation and the 20-day volume-based Z-score are now isolated as the lone features for GRF and NGRF-based trading strategies. We only consider these \( k = 2 \) features and set \( \tilde{k} = k = 2 \) so that both features can be considered at each split point. The strategies are also constructed according to the previously-mentioned rolling windows process and their returns are also included in Figure 2. While, unsurprisingly, the performances of both these strategies suffer as compared to those of the 8-feature versions, the disparity between the GRF and NGRF models’ performances is dramatically accentuated. The GRF-based strategy Sharpe ratio shrinks to -0.09, while the NGRF based strategy is still able to achieve a Sharpe ratio of 0.83. Similar outperformance pertains as measured in terms of success rate. This suggests that the non-greedy DTs effectively account for the XOR-like relationship while the greedy DTs do not.

**Conclusion**

The premise of data science relies upon the existence of patterns in complex data. As exemplified in this article, XOR-like relationships are prominent among such patterns (Figures 1, 2). We have thus
introduced the non-greedy random forest (NGRF) to address the shortcomings of its greedy counterpart. The NGRF’s (step-wise) non-greedy DTs are optimized for the global Gini-score (4) at a fixed depth of 2, and can recursively append additional non-greedy layers as subtrees of depth 2. We demonstrated that the NGRF significantly outperformed the GRF when XOR-type structures are present, and that they otherwise perform similarly. This was evident both for synthetic data and for the bitcoin price time series. The NGRF can therefore be viewed as a valuable replacement for the GRF when examining feature interdependences, as it offers potential performance enhancement with little downside.

References

[1] G. Biau and E. Scornet. A random forest guided tour. Test, 25(2):197–227, 2016.

[2] P. W. Anderson. More is different. Science, 177(4047):393–396, 1972.

[3] F. Patzelt and K. Pawelzik. An inherent instability of efficient markets. Scientific reports, 3:2784, 2013.

[4] P. W. Anderson. The economy as an evolving complex system. CRC Press, 2018.

[5] G. James, D. Witten, T. Hastie, and R. Tibshirani. An introduction to statistical learning, volume 112. Springer, 2013.

[6] K. P. Bennett. Global tree optimization: A non-greedy decision tree algorithm. Computing Science and Statistics, pages 156–156, 1994.

[7] M. Norouzi, M. Collins, M. A Johnson, D. J. Fleet, and P. Kohli. Efficient non-greedy optimization of decision trees. In Advances in neural information processing systems, pages 1729–1737, 2015.

[8] U. Hackett. But not both: the exclusive disjunction in qualitative comparative analysis (qca). Quality & Quantity, 49(1):75–92, 2015.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. Journal of machine learning research, 12(Oct):2825–2830, 2011.

[10] M. L. De Prado. Advances in financial machine learning. John Wiley & Sons, 2018.

[11] M. B. Kursa and W. R. Rudnicki. Feature selection with the boruta package. J Stat Softw, 36(11):1–13, 2010.

[12] D. H. Bailey and M. L. de Prado. The deflated Sharpe ratio: correcting for selection bias, backtest overfitting, and non-normality. The Journal of Portfolio Management, 40(5):94–107, 2014.

Appendix A. Asset Selection

When picking an asset to investigate the performance of the NGRF relative to the GRF, we must be careful to not try a myriad of assets at random until one (essentially out of luck) returns the desired results. With this in mind, we first narrow the eligible universe by eliminating (A) all indices and instruments that do not include volume data, (B) all assets and ETF’s that split or pay dividends, (C) all assets that
seem too illiquid to trade without impact, (D) all assets that are so liquid that return prediction based on simple technical indicators seems out of reach. Given these constraints, we (somewhat randomly) select bitcoin and continuously rolled front month futures for coffee and wheat. For those three assets, we follow the strategy explained above but with GRFs only. We set \( \tilde{k} = 0.5k \) so that each DT is allowed to consider only 5 randomly selected features at each split node. The asset on which the GRF-based strategy performs best is bitcoin and we accordingly narrow our focus to bitcoin. We find that the two commodity-based strategies have Sharpe ratios that hover around random values between -0.2 and +0.2 for different intervals. To account for trying 3 different assets and hence potentially overfitting, we also calculate Deflated Sharpe ratios [12] which leads to corrections of our reported annualized Sharpe ratios of 0.25 at most. This discount does not affect the quality of our results.

Appendix B. Technical Indicators

We describe the technical indicators implemented in section b.

- RSI (Relative Strength Index):
  is a momentum indicator that tracks the recent price changes of an asset. It oscillates between 0 and 100. The 5-day and 20-day RSI’s have been used in section b.

- Sign correlation:
  is the correlation coefficient between the past returns for \( n \) days. The correlation coefficient ranges from -1 to 1. The 5-day and 20-day Sign Correlations’s have been used in section b.

- Overnight-gap:
  is the percentage difference between an asset’s opening price and its previous day’s closing price.

- Close location value:
  is used to measure the closing. It ranges in value from -1 to 1 and is calculated to measure the relative difference between an asset’s closing price and its day’s high and low prices.

- Volume-based Z-score:
  is the difference between an asset’s volume and its trailing exponential moving average for the past \( n \) days divided by the standard deviation of the volume of the past \( n \) days. The 5-day and 20-day Volume-based Z-score’s has been used in section b.

BTC trades around the clock. For the purpose of generating technical price indicators, we mark the daily BTC/USD “OHLCV” data based upon U.S. equity market hours (EST).