Utah State University

DigitalCommons@USU

All Graduate Theses and Dissertations | Graduate Studies

5-2020

Applications of Machine Learning in High-Frequency Trade Direction Classification

Jared E. Hansen
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/etd

 Part of the Statistical Models Commons

Recommended Citation

Hansen, Jared E., "Applications of Machine Learning in High-Frequency Trade Direction Classification" (2020). *All Graduate Theses and Dissertations*. 7764.
https://digitalcommons.usu.edu/etd/7764

This Thesis is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Theses and Dissertations by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.

APPLICATIONS OF MACHINE LEARNING IN HIGH-FREQUENCY TRADE DIRECTION CLASSIFICATION

by

Jared E. Hansen

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Statistics

Approved:

Kevin Moon, Ph.D. Major Professor	D. Richard Cutler, Ph.D. Committee Member
Tyler Brough, Ph.D. Committee Member	Todd Griffith, Ph.D. Committee Member

Richard S. Inouye, Ph.D.
Vice Provost for Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2020

Copyright © Jared E. Hansen 2020

All Rights Reserved

ABSTRACT

Applications of Machine Learning in High-Frequency Trade Direction Classification

by

Jared E. Hansen, Master of Science

Utah State University, 2020

Major Professor: Kevin Moon, Ph.D.
Department: Mathematics and Statistics

The correct assignment of trades as buyer-initiated or seller-initiated is paramount in many quantitative finance studies. A variety of simple decision rule methods have been used for signing trades since many data sets available to researchers do not include the sign of each trade executed. Utilizing these older methods and originally-engineered features as predictors, we have demonstrated that machine learning models outperform existing methods, achieving state-of-the-art results. Out-of-sample predictive accuracies of random forests exceeded those of older methods by 4.5 percentage points overall, were higher for all securities tested, and improved prediction for some securities up to 12.1 percentage points. Since finance and economics departments pay thousands of dollars in annual data service subscription fees they are often reluctant to fund order book data purchase. The use of our best trade signing model as an alternative to the purchase of order book data has the potential to collectively save universities millions of dollars in additional subscription fees, facilitate more reliable research, and remove the burden of order-level data processing for researchers in need of accurate trade signs.

(98 pages)

PUBLIC ABSTRACT

Applications of Machine Learning in High-Frequency Trade Direction Classification

Jared E. Hansen

The correct assignment of trades as buyer-initiated or seller-initiated is paramount in many quantitative finance studies. Simple decision rule methods have been used for signing trades since many data sets available to researchers do not include the sign of each trade executed. By utilizing these decision rule methods, as well as engineering new variables from available data, we have demonstrated that machine learning models outperform prior methods for accurately signing trades as buys and sells, achieving state-of-the-art results. The best model developed was 4.5 percentage points more accurate than older methods when predicting onto unseen data. Since finance and economics departments pay thousands of dollars in annual data service subscriptions they are often reluctant to fund purchase of additional data containing trade signs when methods for predicting these signs exist. The use of our best trade signing model as an alternative to the purchase of additional data has the potential to collectively save universities millions of dollars in additional subscription fees, facilitate more reliable research, and lighten the burden of data processing for researchers.

ACKNOWLEDGMENTS

I am incredibly fortunate to have had the opportunity to work with, get to know, and learn from each member of my committee in a way that I don't think most graduate students do. Dr. Moon, thank you for pushing me hard in your courses while being patient and allowing me to push myself in research. You have instilled in me an appreciation of deeply understanding how things work rather than simply applying them. Dr. Cutler, thank you for your mentorship and wisdom from my undergraduate days through until now. Much of my decision to pursue this degree and field of work is thanks to your fantastic teaching and encouragement. Dr. Griffith, thank you for your help with so many of the finance-related specifics of this work. If it weren't for your advice regarding time management and career choices I might still be writing this thesis and looking for a job. Dr. Brough, thank you for helping me get out of my academic comfort zone and exemplifying an attitude of continual growth and evolution. By exposing me to new ideas, academic and otherwise, you have helped me to adjust my prior and change my outlook on statistics and on life.

To fellow students, thank you for your friendship throughout my graduate experience. Eric, thank you for putting up with my questions and letting me bounce ideas off of you. Matt, thank you for your example and encouragement to get after it. Dan, Sam, and Brandon, thank you for enlarging my perspective and always being up for an engaging conversation. Ronak and Sharad, thank you for your guidance regarding a career in industry and advice on how to set myself up for success.

Gary Tanner, you are the unsung hero of mathematics and statistics graduate students at USU. Thank you for helping me to stay on top of administrative necessities and always taking the time to answer my questions.

Finally, thank you Mom, Dad, Amanda, Natalie, and Rebeka for putting up with me throughout this process. Without your encouragement, patience, comedic relief, and support none of this would have been possible.

Jared E. Hansen

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

AMEX	American Stock Exchange
CLNV	Chakrabarty, Li, Nguyen, and Van Ness
EDA	Exploratory Data Analysis
EMO	Ellis, Michaely, and O'Hara
FOMC	Federal Open Market Committee
HFT	High-frequency trading
KDE	Kernel Density Estimate
k-NN	k-Nearest Neighbors
LR	Lee and Ready
NASDAQ	National Association of Securities Dealers Automated Quotations
NBB	National Best Bid
NBO	National Best Offer
NBBO	National Best Bid and Offer
OOB	out-of-bag
PCC	percent correctly classified
PHATE	Potential of Heat-diffusion for Affinity-based Trajectory Embedding
QDA	Quadratic Discriminant Analysis
RF	Random Forests
SEC	Securities and Exchange Commission
SVM	Support Vector Machine
TAQ	Trades and Quotes
VIMP	variable importance
wab-PCC	weighted-all-but percent correctly classified
WRDS	Wharton Research Data Services

CHAPTER 1

INTRODUCTION

In many quantitative finance studies, the correct assignment of trades as buyer-initiated or seller-initiated is paramount. Various methods have been used for signing trade direction since many data sets available to financial researchers do not include the sign of each trade executed. Most of these methods are simple decision rules, and the only data they rely on are the best quotes on the order book just prior to a trade's execution and the price of a trade relative to prices of previous trades. By utilizing these decision rule methods, as well as engineering new predictive features from available data, we have demonstrated that machine learning models achieve state-of-the-art results for accurately signing trades. Adoption of our best model has the potential to collectively save universities millions of dollars in order book data subscription fees, to facilitate more reliable research, and to relieve the burden of processing order-level data for those in need of accurate trade signs.

1.1 Prominent Prior Studies

Several well-known studies which have explored the problem of trade signing will be referenced throughout this thesis. They will be introduced here and then discussed in further depth in Chapter 3. The seminal work of Lee and Ready (LR hereafter) [1], which primarily relies upon the midpoint of the best quotes for classification, is well-known in financial literature and has been cited over 3,000 times. Ellis, Michaely, and O'Hara (EMO hereafter) [2] used NASDAQ (National Association of Securities Dealers Automated Quotations) data to demonstrate a new algorithm which principally classifies trades based on the price of execution relative to the prevailing best bid and ask. The work of Chakrabarty, Li, Nguyen, and Van Ness (CLNV hereafter) [3] uses prevailing quotes to calculate five ranges of values, using the quote test (see Section 3.2) to classify trades in the second and fourth ranges, and the tick test (also see Section 3.2) to classify trades in the first, third, and fifth ranges.

Rosenthal moved from the simplicity of decision rules to a statistical modeling approach, employing logistic regression to predict trade sign [4].

1.2 Why Machine Learning and Feature Engineering

A key difference between previously-used decision rules and the machine learning methods employed in this work is the degree to which data is fully used. Decision rule methods rely on only three pieces of information: bids, asks, and prices. Our new approaches take advantage of these features as well as others, including: time of trade execution, time between trades (trade distance [3, p. 3808]), time of best quote placement immediately prior to trade execution (quote distance [3, p. 3808]), the size of the order executed (number of shares traded), the size of the best quotes, and the predicted signs of these older decision rules. Additionally, we engineer many new features from these original ones. Rosenthal's model [4] encoded prices and quotes into original and once-lagged metrics for use as predictors, and was influential in inspiring our feature engineering.

Engineered features make more predictive information available, allowing machine learning methods to characterize a more complex model for predicting trade sign [5]. In turn, this results in a higher accuracy for classification of trade signs than the methods used by LR, EMO, and CLNV. While older methods are somewhat successful, they can only correctly classify trades that conform to their rigidly-defined pattern, systematically biasing results [2]. The beauty of the machine learning methods tested here is their flexibility; they can essentially learn to use older methods to sign trades that fit a certain profile, or alternatively use a more intricate method for anomalous trades. However, there is a downside to this intricacy. These more complex models can be difficult to interpret [6, p. 25] whereas the decision rule based approaches are quite transparent. However, the primary interest of researchers is that trades be signed accurately, so lack of interpretability is a negligible concern. Nonetheless, some insights into the behavior of our final model are extricated and discussed in Section 6.3.

1.3 Relevance of This Work

Although many institutional traders subscribe to data services which provide trade direction, some may not. For those who don't, estimation of important pricing models is subject to the noise resulting from less than 100% accuracy as obtained by the methods of LR, EMO, CLNV, and others. This may result in inefficient execution of illiquid orders, with the trader absorbing the cost of buying or selling at sub-optimal prices. As detailed by Rosenthal, a modest improvement in trade signing accuracy, even just one or two percent, would affect estimates of price impact and could potentially results in savings of hundreds of millions of dollars over the course of a year for a large bank [4].

Another motivator of the work in this thesis is the heavy use of trade direction in academic research. Financial market microstructure studies often rely on access to accurately signed trades. Rather than pitting machine learning and traditional econometric modeling against each other, we demonstrate that a symbiosis between the two can be achieved. Machine learning is used here to augment data with more accurate trade directions, which can then be used to create more accurate econometric models and more reliable empirical results. Some examples of the use of trade signing in financial literature include:

- Studies of information asymmetry and inventory-control to characterize specialist behavior [7] and to model bid/ask spread components [8].
- Research related to stock prices, such as price impermanence in large transactions [9], order imbalance [10], and explanation of aberrant closing prices [11].
- The examination of market response to information events, such as differing response of small and large traders to corporate earnings announcements [12], the contradictory behavior of small traders buying even after negative news [13], and informed trading with pre-release information from FOMC announcements [14].
- Liquidity measures [15], modeled spread and impact measures such as realized spread, effective spread, trade-specific price impact, and transaction costs all rely upon trade directions in their respective calculations [16].

- Sophisticated econometric models for prices such as the work of Madhavan et al. [17], variants of a generalized Roll model for prices and trades [18], and a VMA model [18] of efficient prices generalized from Glosten and Harris [11].

As noted by Hautsch, "...a noisy buy-sell identification can have a severe impact on these measures" [16, p. 228]. Therefore, it is of crucial importance to financial research validity that trades be signed accurately.

If having accurate data is of such importance for research, then why are researchers estimating trade signs instead of using databases that already have them available with 100% accuracy? The short answer is **money**. Wharton Research Data Services (WRDS) is used as a research data source by more than 400 institutions throughout the world [19], universities in particular. Although no official figures are publicly available on the WRDS website or otherwise, an estimate from 2005 puts the cost of a base subscription at $35,000 per year [20]. Many economics and finance departments require much more than just the base subscription data, and end up paying far more than this, often eclipsing the $100,000 mark in yearly fees. Despite the high cost, data needed for a wide array of financial research can be obtained via WRDS. A notable exception is trade direction, which is not an included field in available databases. A few services do exist which provide data containing trade directions. NASDAQ provides a data subscription called ITCH which costs over $1,000 per month [21]. Lobster, a service which reconstructs order book data for NASDAQ-listed securities, requires a year-long subscription which costs over $6,200 [22]. A reduced version of Thomson Reuters' Eikon service can be had for $3,600 per year while a subscription to the full service costs $22,000 per year [23].

Since departments already pay thousands of dollars in WRDS subscription fees, they are often reluctant to fund further data purchase when trade signing methods exist, however inaccurate these methods might be. Individual faculty members often do not have the funds necessary to buy order book data from a provider like NASDAQ or Thomson Reuters (see Section 2.2 for an explanation of what a limit order book is). Our approach of applying machine learning techniques to this problem is an ideal solution. It achieves more accurate

results than simpler established methods, enabling more reliable research. Additionally, our contribution has very real potential to save universities and taxpayers millions of dollars in order book data subscription fees.

CHAPTER 2

FINANCE BACKGROUND

Before going into the particulars of methods used for predicting trade direction, it is worth discussing the process that generates this data: the framework for trading itself. For those not familiar with trading, this will provide needed context for the rest of this work, and for all readers it will lend insight into data-generating processes which inform modeling decisions. Discussion here will center around the NASDAQ exchange since this is the source for our data, but much of the information generalizes to other exchanges.

2.1 Traders and Dealers

Trade is a central feature of markets throughout economies. Participants often deal directly with their counter-party, exchanging some good or service for cash or credit at an agreed-upon price. In many financial markets this is often not the case, especially when trading via an exchange such as the NYSE or NASDAQ. Individual investors are typically not allowed to trade directly, and must go through a broker or broker-dealer who is a member of the exchange [16, p. 10]. Brokers trade on behalf of their clients according to client instructions, while dealers trade for their own account. As one might guess, broker-dealers can trade on behalf of others as well as for themselves [24]. Broker-dealers often facilitate trading and increase market liquidity by taking the opposing position of a client's trade or lending shares for short selling. Dealers usually only deal with creditworthy clients with established reputations of reliability, while brokers serve traders without this same level of trustworthiness, guaranteeing credit when trading with dealers on behalf of their clients [16, p. 13].

2.2 Orders and the Limit Order Book

Since investors' participation in trading is facilitated through a broker or broker-dealer,

they must submit an order detailing their intentions; which security they want to trade, the size (number of shares), and the price at which they wish to buy or sell are all necessary for making an order. An order to buy is known as a *bid*, while an order to sell is known as an *offer* or *ask*, and either can be generically referred to as a quote. The best bid on an exchange is the buy order with the highest per-share price and the best offer is the sell order with the lowest per-share price. We will represent the best bid at time k as B_k and the best ask at time k as A_k. At any given time, the difference between the best ask and the best bid gives the bid-ask spread, or just "the spread": $\left(A_k - B_k\right)$. Since different exchanges may have different best bids and asks, the National Market System aggregates this information and records the National Best Bid (NBB) and National Best Offer (NBO), collectively referred to as the NBBO [16].

Each exchange maintains what is known as a limit order book, order book for short, for each security that it lists. Prior to computers, this data was maintained using an actual book (NYSE) or a chalkboard (Tokyo Stock Exchange), but it is now done electronically using a computerized data structure [25]. The order book organizes all orders made by dealers for the given security and is used to match orders for trade execution. A contrived example of a simple order book is given below in Table 2.1. The order book which market participants can see would be similar to this with a few exceptions; it would not show the hidden orders, it would not show the names of other traders, and the information would be given in a stream of messages rather than in a table like this (to be discussed in Chapter 4). It should also be noted that exchanges often will not know who the traders are [25, p. 21]. Names are simply given here to illustrate the concept of various traders participating.

Orders on an exchange's book are arranged according to the (potentially) unique priority schedule of that exchange. Almost always, orders with the best price (high for bids and low for asks) have the best position in the book. After ordering by price, many markets will sort by the time at which the order was received, earlier receipts having higher priority [25, p. 20-21]. A tertiary sorting criterion is order visibility. Some markets allow traders to disguise their strategies by submitting orders that are partially or entirely hidden from

the rest of the market [16]. After sorting by price, visible orders will often have priority over hidden orders even if the hidden order arrived earlier. Quote prices are effectively an advertisement of an exchange's liquidity and a clear indicator of the exchange's viability, explaining this prioritization of visible orders ahead of hidden ones [25, p. 21].

Table 2.1 is a contrived example of a limit order book for the fictitious "USU" stock. (USU was formerly the ticker for Centrus Energy, which now uses the ticker LEU. It is surprisingly difficult to find a combination of letters which isn't already used as a ticker, therefore we will stick with USU since it is a defunct ticker as well as a fantastic institution.) In this table we can see several characteristics of order books as described above. The best bid is from Tony and the best ask is from Meadow. These may not necessarily be the NBB and NBO respectively, but they are the best bid and ask for this market. The exchange listing USU stock here prioritizes orders first by price, then by visibility, then by timestamp. The primary ordering by price is easily observed by the fact that both the bids and ask are strictly ordered by price "best to worst": greatest to least for bids, and least to greatest for asks, with the best prices nearest the center of the book. We can also see a secondary ordering by visibility on both sides of the book. Although Paulie and Janice have both placed bids of $158.47 per share and Janice's bid arrived first, since Janice's order is hidden it is entered below Paulie's. The same holds true for Carmela and Silvio on the sell side of the book. The tertiary prioritization by timestamp can be seen by comparing the last two entries on the buy side of the book. Ralph and Richie both placed un-hidden bids at $158.46 per share; since Ralph's order was received first it is given higher priority.

Many different types of orders exist. We will focus on market orders and limit orders and some of their variants, illustrating how they work using our order book in Table 2.1. A market order is an order to immediately buy or sell at the best possible price(s) [16, p. 11]. If the size of the order (number of shares) is larger than that of the best quote, the trader must "walk down/up" the book in order to fill the order. For example, if Jennifer wants to buy 40 shares of USU stock relative to the order book in Table 2.1 she would have to walk up the book, buying 15 shares from Meadow at $158.52 each, 20 shares from Chris

Order book for USU: market maker view					
	Trader	**Time**	**Qty**	**Visibility**	**Price**
Asks (sellers)	Junior	9:40	40		$58.66
	AJ	9:37	30		$58.65
	Silvio	9:33	50	Hidden	$58.63
	Carmela	9:34	50		$58.63
	Chris	9:31	20		$58.60
	Meadow	9:35	15		$58.52
Bid-Ask spread of $0.02					
Bids (buyers)	Tony	9:30	10		$58.50
	Paulie	9:37	30		$58.47
	Janice	9:33	15	Hidden	$58.47
	Adriana	9:39	25		$58.46
	Ralph	9:40	40		$58.45
	Richie	9:42	50		$58.45

Table 2.1: A contrived order book example from the view of the exchange listing the fictional USU stock. Order precedence is determined by price, visibility, then time, and the concepts of best bid, best ask, and the spread are illustrated.

at $158.60 each, and five shares from Carmela at $158.63 each. Or if Jennifer wants to sell 10 shares of USU stock Tony would fill the order, paying $158.50 per share. In both cases, Jennifer ends up paying the bid-ask spread on top each of these transactions. In order to operate and be profitable, the market (which is often also the dealer) earns the spread for their services [16, p. 13].

An alternative to a market order is a limit order; the trader specifies a limit price, instructing the dealer to buy or sell at a price no worse than the limit price [16, p. 11]. If the limit price is better than the best quote on the opposite side of the book (e.g. a buy order with a limit price greater than the best ask, or a sell order with a limit price less

than the best bid) then the trade executes immediately. Otherwise the order is placed onto the book according to that exchange's priority rules as described above. For example, if Vito places a limit order to buy 12 shares with a limit price of $158.54, his order is filled by Meadow's offer, with Vito paying $158.54, the exchange making the spread of $0.02 per share, and Meadow's remaining three shares staying on the book. However, if Vito placed that same order but with a limit price of $158.48 his order would be placed on the book between Tony and Paulie's respective bids.

2.3 Prominence of High-Frequency Trading

Thanks to advances in computing and telecommunications, trading has evolved from limit orders being recorded on chalkboards and in actual books to limit orders being recorded in high-speed, order book databases. These technological improvements have facilitated an increase in trading activity and complexity via electronic exchanges such as the NASDAQ. The use of computerized algorithms to execute high volumes of trades has become known as high-frequency trading (HFT hereafter). This development has resulted in increased interest in order placement, optimal trade execution strategy, liquidity dynamics, and intraday trading [16] from academics and industry practitioners alike. Highlighting the relevance to this work, study of liquidity dynamics often relies on knowing the signs of trades [15].

As of 2011, it was common to see more than 100,000 trade executions per day for highly-traded, blue chip stocks [16]. Around this same time, roughly 60-70% of U.S. trading was HFT, moving up to 3.25 billion shares of stock per day [26]. Although HFT activity has somewhat declined since this period, it still generates massive amount of granular, high-quality data which academics can study and industry players can use to develop more efficient trading strategies.

2.4 NASDAQ Details

The data used in this thesis comes exclusively from the NASDAQ exchange. (All but two of the stocks used - BABY and CA - are also listed on the NYSE.) NASDAQ is a quote-driven dealer market, meaning that trades can only be executed by traders and bro-

kers through a dealer. Established in 1971, NASDAQ was the world's first electronic stock market [27] and was used to connect geographically disparate dealers. Prior to 1995, competitive quotes from traders were not always displayed publicly on the NASDAQ, affording dealers an informational advantage which they capitalized on by trading for their own account. From 1995 onward the SEC has required that exchanges display the best quotes, including those from inter-dealer markets. This caused bid-ask spreads to drop significantly on the exchange [16, p. 17].

NASDAQ utilizes a form of an electronic limit order book maintained by dealers, facilitating "...trade execution, reporting, confirmation, and interdealer communication" [16, p. 17]. By market capitalization, the NASDAQ is the largest U.S. electronic stock market [16, p. 17] and the second largest U.S. stock market overall [28]. NASDAQ holds a wealth of information related to activity on their exchange, much of which is available for purchase. Since WRDS (the data subscription service mentioned in Section 1.3) does not contain trade directions, we obtained this data via NASDAQ's data product TotalView-ITCH 5.0 (ITCH hereafter) [29].

CHAPTER 3

TRADE SIGNING LITERATURE

We now discuss in detail the trade signing methods introduced in Section 1.1. We use the outputs of some of these methods as features, and their respective approaches substantially inform other modeling decisions that we make and features that we engineer. For the tick test, quote test, LR, EMO, and CLNV methods, we will explain the algorithm along with giving practical, illustrative examples of these methods at work. The work of Rosenthal, and its impact on our models, is also discussed. We mention the test accuracies of some methods within studies but do not compare *across* studies. This is mainly because each study used different data to obtain their results, effectively invalidating any direct comparisons.

3.1 Determining Trade Direction

While numerous order types exist, the elementary understanding of market and limit orders given in Chapter 2 is sufficient for this thesis. Armed with this knowledge, we can understand the most crucial financial definition for this work; the later-arriving order is considered the trade's initiator, determining whether we sign a trade as a buy or a sell [30, p. 262]. In the example from Section 2.2 in which Vito places a limit order to buy 12 shares at a limit price of \$158.54, Vito is the later-arriving order. Since Vito is a buyer, the trade which executes at \$158.54 for 12 shares is a buy. The example in which Jennifer places a market order to buy 40 shares results in three separate trades are all considered buys since Jennifer's later-arriving order is what initiated the trades. It is also imperative to note the behavior of the best quotes during these trades, as best quotes are the bedrock of predictive information when developing models. For Vito's trade, the best ask is Meadow's offer and the best bid is Tony's. For Jennifer's three-trade market order, the best bid remains the same (Tony's), while the best ask moves up the book each time a

trade executes.

3.2 The Tick Test and the Quote Test

The precise origins of the tick test and quote test are a bit murky. Based on mentions in journal articles by Holthausen et al. [9] and Hasbrouck [7], it appears that they were first used for assigning trade direction sometime around 1987. Thorough explanations of the methods can be found in Lee and Ready's exposition of their own new method [1], which is a combination of the two. These tests are absolutely foundational for trade signing in academic financial literature, and both play an integral part in the LR, EMO, and CLNV methods.

For both tests we will define the following:

- Let our trade-of-interest be at time k, executing at price P_k.
- Let P_j be the price of the trade immediately preceding k, occurring at j $(j < k)$.
- Let P_d be the price of the first trade preceding k for which $P_k \neq P_d$ (it may be that $P_d = P_j$).
- Let A_k be the price of the best ask (offer) prevailing when the trade at time k executes.
- Let B_k be the price of the best bid prevailing when the trade at time k executes.
- Let $M_k = \left[\frac{A_k + B_k}{2}\right]$ be the midpoint of the best quotes prevailing at time k.

The tick test is very straightforward, relying solely on the prices of trades in order to predict their respective directions. Trades are first categorized as being an uptick, zero-uptick, zero-downtick, or downtick [1]. For the trade at time k we define the four categories and corresponding trade directions:

$$\begin{cases} \text{uptick} & \textit{buy} & (P_k > P_j) \\ \text{zero-uptick} & \textit{buy} & (P_k = P_j) \text{ and } (P_k > P_d) \\ \text{zero-downtick} & \textit{sell} & (P_k = P_j) \text{ and } (P_k < P_d) \\ \text{downtick} & \textit{sell} & (P_k < P_j) \end{cases}$$

The quote test relies on slightly more information, requiring the price of the trade and the midpoint of the prevailing best quotes for that trade. According to the quote test, trades are signed such that if $P_k > M_k$ it is a buy, and if $P_k < M_k$ it is a sell. If $P_k = M_k$ the quote test fails to make a classification.

A brief illustrative example of these two test is given below in Table 3.1.

Tick Test and Quote Test Results for USU Stock						
Time	$\mathbf{B_k}$	$\mathbf{M_k}$	$\mathbf{A_k}$	$\mathbf{P_k}$	**Tick**	**Quote**
9:36	58.66	58.69	58.72	58.72	N/A	buy
9:37	58.63	58.655	58.68	58.63	sell	sell
9:39	58.63	58.655	58.68	58.63	sell	sell
9:40	58.60	58.62	58.64	58.63	sell	buy
9:41	58.60	58.62	58.64	58.64	buy	buy
9:43	58.60	58.64	58.68	58.65	buy	buy
9:44	58.65	58.69	58.73	58.66	buy	sell
9:45	58.60	58.60	58.62	58.60	sell	N/A
9:46	58.58	58.605	58.63	58.60	sell	sell

Table 3.1: A contrived example exhibiting the tick test and quote test for classifying trades. Until a differing price is observed the tick test is unable to make a prediction (see the first trade). Similarly, the quote test is unable to make predictions for trades occurring at the midpoint (see the eighth trade).

3.3 The LR Method

The method of Lee and Ready has been foundational for trade signing since its publication in 1991, both for practical use in signing trades as well as for comparison with competing methods. Their work demonstrated inefficiencies in using the tick test to classify trades within the spread due to insufficient information from relative prices as the sole predictor of sign. Their solution was to use quote data (the best bid and the best ask) corresponding to a trade to provide further predictive information through the quote test. Another issue raised in their work was the inconsistent timing of quotes relative to corresponding trades due to human clerical entry. This is no longer a large concern thanks to

electronic order books. For the data available at the time, they recommended the use of quotes with a timestamp five seconds prior to the timestamp for a trade when calculating the midpoint of quotes [1]. The LR method signs trades by applying the quote test first; if the quote test fails to make a classification (e.g. the trade executes at the prevailing midpoint), then the tick test is used to make the classification. Table 3.2 gives an example of the LR rule at work in comparison to other methods.

3.4 The EMO Method

The study done by Ellis, Michaely, and O'Hara [2] was the first to examine trade signing using NASDAQ data. The data used consisted of roughly 1.8 million trades and 600,000 corresponding quotes. Using the LR method to sign trades first, they fit a logistic regression model to characterize how features impact the probability of correct classification. They showed that proximity of trade prices to quotes is by far the most important predictor for accurately signing trades. Other factors such as the size of the trade, the size of the firm trading, trade distance, quote distance, and amount of trading activity also affected ability to predict signs accurately. Trades occurring inside of the quotes, e.g. $(P_k > B_k)$ & $(P_k < A_k)$, were particularly hard to classify. The LR method achieved only 55% accuracy on these trades with prices inside of the quotes. However, trades occurring ***at the quotes*** (either $P_k = A_k$ or $P_k = B_k$) were classified at a phenomenal 88.68% accuracy.

Since trades inside of the quotes were poorly classified by LR and trades at the quotes were classified well by LR, the authors proposed a new method for signing trades. Trades were signed such that if $P_k = A_k$ the trade is a buy, if $P_k = B_k$ the trade is a sell, and all other trades' signs are determined using the tick rule; this is the EMO method. This method yielded a solid improvement for classifying trades inside the quotes (went from 55% with LR to 61% with EMO) and the accuracy for trades outside of the quotes went from 64.8% to 65.8%. Overall accuracy barely increased from 81.05% with LR to 81.87% with EMO, but they showed that calculation of the effective spread is much more accurate using EMO-signed trades than using LR-signed trades [2]. An illustrative example of the EMO method is also contained in Table 3.2.

3.5 The CLNV Method

The key improvement of the trade signing method developed by Chakrabarty, Li, Nguyen, and Van Ness is their unique treatment of trades occurring inside of the quotes. They made the keen observation that the quote rule outperforms the tick rule when classifying trades whose prices are near the quotes, but the tick rule outperforms the quote rule for trades near the midpoint. Based on this, they developed the following method for classifying trades:

$$\begin{cases} \text{Above the ask} & P_k > A_k & \text{use tick rule} \\ \text{Top 30\% of spread} & P_k \in \left[0.7A_k + 0.3B_k, A_k\right] & \text{use quote rule} \implies \text{buy} \\ \text{Middle 40\% of spread} & P_k \in \left(0.3A_k + 0.7B_k, 0.7A_k + 0.3B_k\right) & \text{use the tick rule} \\ \text{Bottom 30\% of spread} & P_k \in \left[B_k, 0.3A_k + 0.7B_k\right] & \text{use quote rule} \implies \text{sell} \\ \text{Below the bid} & P_k < B_k & \text{use tick rule} \end{cases}$$

The lower bound for the "Top 30% of spread" range comes from simply adding 20% of the spread to the midpoint and algebraically manipulating that term. Recall from Section 2.2 that the spread $= A_k - B_k$, and from Section 3.2 that $M_k = \left[\frac{A_k + B_k}{2}\right] = 0.5A_k + 0.5B_k$. Therefore we have $\left[\text{midpoint} + 20\% \text{ of spread}\right] = \left[M_k + (0.2)(A_k - B_k)\right] = \left[(0.5A_k + 0.5B_k) + (0.2A_k - 0.2B_k)\right] = \left[0.7A_k + 0.3B_k\right]$. The upper bound for the "Bottom 30% of spread" range is obtained in a nearly identical manner, subtracting 20% of the spread from the midpoint. Although calculations of these ranges aren't provided in Table 3.2, the results of the CLNV method for the sample data are given in the table for comparison to the other methods discussed.

The authors developed the CLNV rule on NASDAQ ITCH data for 750 stocks from April - June of 2005. In a first for trade signing papers, the researchers adopted a commonly used paradigm in statistical and machine learning modeling; they split their data into a training sample (April) used for model development, and a test sample (May-June) for model evaluation and final test accuracy reporting [3, p. 3809]. Although a ratio of 1:2 for training versus test data is considerably different than typically-used splits, this modeling

paradigm is a strength of this study relative to previous works.

Chakrabarty et al. examined the April training data by using the LR, tick, quote, and EMO rules to observe patterns which informed the formulation of the resulting CLNV method. They observed that nearly a third of their trades were executed inside of the quotes. These trades proved harder to sign, making them especially crucial for achieving a higher accuracy. Correctly signing these trades is equally crucial for finance applications such as calculation of effective spread and price impact. The final test accuracies reported were 74.42% for LR, 75.80% for EMO, 75.40% for the tick rule, and 76.52% for CLNV. Considering only trades inside of the quotes (exlcuding trades at the midpoint), test accuracies were 71.85% for LR, 71.35% for the tick rule and EMO, and 76.32% for CLNV.

In order to show the relative importance of variables for correct classification, the authors performed logistic regression [3, p. 3817] just as EMO did [2, p. 538]. First, they signed trades using the CLNV method, creating data whose response was 1 if the CLNV sign agreed with the ground truth and 0 if the CLNV sign disagreed with the ground truth. They used this data with the new response to construct the regression model, obtaining a vector of estimated coefficients B. They characterized each variable's impact on accuracy by the size and sign of its transformed coefficient. This transformation was performed using the equation $\left[\frac{\partial\theta(x)}{\partial x} = \frac{e^{B^T X}}{1+e^{B^T X}}\left(1-\frac{e^{B^T X}}{1+e^{B^T X}}\right)B\right]$ where X is a vector containing the means of each variable, and $\theta(x)$ is the probability of correctly classifying sample x [3, p. 3818]. The analysis of these transformed coefficients substantiated the idea that trades occurring at the quotes are the easiest to correctly classify; this finding agrees with the the work of EMO [2, p. 539]. The authors also found that trades having larger trade distances and larger quote distances are more easily classified. Zero-tick trades (a trade whose price is the same as the trade immediately preceding it), trades which occurred when the spread was tighter, and trades of larger size were all more difficult to correctly classify.

3.6 Rosenthal's Method

Rosenthal was one of the first researchers to use something other than a basic decision rule for classifying trades. He employed a logistic regression model which drew from the

Trades, Quotes, and Predicted Signs for USU Stock								
Time	$\mathbf{B_k}$	$\mathbf{M_k}$	$\mathbf{A_k}$	$\mathbf{P_k}$	**Tick**	**LR**	**EMO**	**CLNV**
9:36	58.66	58.69	58.72	58.72	N/A	buy	buy	buy
9:37	58.63	58.655	58.68	58.63	sell	sell	sell	sell
9:39	58.63	58.655	58.68	58.63	sell	sell	sell	sell
9:40	58.60	58.62	58.64	58.63	sell	buy	sell	buy
9:41	58.60	58.62	58.64	58.64	buy	buy	buy	buy
9:43	58.60	58.64	58.68	58.65	buy	buy	buy	buy
9:44	58.65	58.69	58.73	58.66	buy	sell	buy	sell
9:45	58.60	58.6	58.62	58.60	sell	sell	sell	sell
9:46	58.58	58.605	58.63	58.60	sell	sell	sell	sell

Table 3.2: A contrived example exhibiting the tick test, LR, EMO, and CLNV methods for classifying trades. Note that the LR, EMO, and CLNV methods agree most of the time, generating the same predictions for seven of the nine trades here. Using the output of these methods as predictors will leverage their high performance on well-characterized trades. However, these decision rules fail to reach a consensus on two of the trades in the example. It may be that these trades can be more accurately predicted by a more intricate model.

intuition of previous methods, relying primarily upon the information contained in quotes and prices for making predictions. He also made points about the effects which varying delays between trades and quotes have upon trade direction prediction, proposing a model for predicting these delays.

As seen in previous methods, the quotes matched with each trade and used for prediction are considered some of the most predictive information available for generating classifications. Therefore, good quote information is of critical importance in making accurate, unbiased predictions of direction. Most methods just assume some fixed delay between the trade and its quotes, using the quote values in effect after accounting for that delay. Rather than picking a single bid and a single ask as the best quotes for a given trade, Rosenthal models delays as probability density estimates, using these densities to estimate a single value for the best bid ($\hat{B}_k$) and the best ask ($\hat{A}_k$) for each trade.

In the logistic regression model for predicting trade signs, Rosenthal uses information strength metrics as predictors rather than the actual binary outputs of past rules. He relies

on a midpoint metric, a tick metric, and a quote metric, as well as their respective once-lagged counterparts, as predictors in the model. Additionally, he incorporates an overall effect for time indices (trades during certain times are more likely to be either buyer or seller initiated) and a within-sector effect to account for higher likelihood for trades in a given sector being buyer or seller initiated. Using the terms defined in Section 3.2, the model specification is:

$$
\begin{aligned}
\eta_k =& \big[\beta_0\big] + \big[\beta_1 g(P_k, \hat{M}_k)\big] + \big[\beta_2 g(P_k, P_d)\big] + \big[\beta_3 J(P_k, \hat{B}_k, \hat{A}_k; \tau)\big] \\
&+ \big[\beta_4 g(P_{k-1}, \hat{M}_{k-1})\big] + \big[\beta_5 g(P_{k-1}, P_{d-1})\big] + \big[\beta_6 J(P_{k-1}, \hat{B}_{k-1}, \hat{A}_{k-1}; \tau)\big] \\
&+ \big[\text{ overall effect }\big] + \big[\text{ sector effect}\big] \\
\pi_k =& \log\left(\frac{\eta_k}{1-\eta_k}\right) \\
\pi_k =& \Big[P(h_k = \text{ buy } | P_k, P_{k-1}, P_d, P_{d-1}, \hat{B}_k, \hat{B}_{k-1}, \hat{A}_k, \hat{A}_{k-1}, \hat{M}_k, \hat{M}_{k-1}, \text{overall effect}, \text{sector effect})\Big]
\end{aligned}
\tag{3.1}
$$

where:

- h_k is the ground truth of the sign for the trade at k, 1 for buy and 0 for sell.
- $g(X_k, Y_k) = log(X_k) - log(Y_k)$ is a log-return function.
- $J(X_k, Y_k, Z_k; \tau) = \exp\left(-\left(\frac{X_k - Z_k}{\tau}\right)^2\right) - \exp\left(-\left(\frac{X_k - Y_k}{\tau}\right)^2\right)$ is a proximity function for measuring information strength, where τ controls the width of the function near the Y_k and Z_k values.
- Excluding the bias term β_0, the items on the first line of the η_k expression are the midpoint, tick, and quote metrics respectively.
- The items on the second line of the η_k expression are the once-lagged midpoint, tick, and quote metrics respectively.

The data source for this work was the December 2004 ArcaTrade dataset. December 1 and 2 data was used for training, containing the 2,836 stocks in the June rebalance of

the Russell 3000 index, coming from a wide variety of sectors and encompassing a range of trading frequencies [4, p. 400]. Only trades with timestamps between the hours of 10:00 A.M. and 3:30 P.M. were kept, resulting in 2,178,307 trades in the final dataset. Since AMEX trades were a miniscule proportion of total data (roughly 0.13%), only NASDAQ and NYSE executions were considered.

The last 20 days of the dataset were used for generating test accuracies. For the roughly 15 million NASDAQ trades used for testing, Rosenthal's model achieved 74.3% accuracy, the tick test just 66.7%, LR 71.6%, and EMO 72.3%, while the CLNV method was not used in this study. The EMO method worked best for classifying trades occurring at or below the bid, the LR/tick method outperformed the other methods for trades inside of the spread, and Rosenthal's model achieved the highest accuracies for trades at or above the ask [4, p. 409].

CHAPTER 4

DATA: DESCRIPTION, CREATION, AND EXPLORATORY ANALYSIS

4.1 Original Data Sources

4.1.1 TAQ Data

As mentioned in Section 1.3, many finance researchers use WRDS to obtain data for studies. One of their most commonly-used products is the Trades And Quotes (TAQ) database. TAQ data contains intraday trades and quotes to microsecond granularity for approximately 8,000 stocks listed on the NYSE, AMEX, NASDAQ, and other U.S. equity exchanges [31]. We have noted previously that TAQ data available from WRDS does not contain the directions for each trade. Therefore, researchers must either buy trade sign data (from a source such as Lobster or NASDAQ), or use a method like LR, EMO, or CLNV to sign their trades. Since signing trades is a well-accepted practice in academic circles, most researchers opt to do this rather than buy the data.

The TAQ data we will be using are daily products, and all of them share the descriptive fields of `DATE`, `TIME_M` (timestamp in microseconds), `EX` (exchange), and `SYM_ROOT` (four character ticker symbol). The three types of TAQ files we use are:

- Trade files: going forward, these files will be referred to as Trade files (capitalized) to distinguish between specific trades and the files containing them. Trade files contain all trades for specified tickers and dates. For the 10 tickers we consider, a single day has approximately 30,000 trades. Relevant fields include `SIZE` (number of shares) and `PRICE` (a trade's execution price).

- Quote files: going forward, these files will be referred to as Quote files (capitalized) to distinguish between specific quotes and the files containing them. Quote files contain all quotes for specified tickers and dates. For the 10 tickers we consider, a single day of

trading contains roughly 2,500,000 quotes. Some of the more active trading days have quote files which take up nearly half a gigabyte of disk space in SAS7BDAT file format (roughly equivalent to CSV size). Relevant fields include `BID` (a limit order bid price), `BIDSIZ` (number of shares for a bid limit order), `ASK` (a limit order ask/offer price), `ASKSIZ` (number of shares for an ask limit order), and `QU_COND` (quote condition, used for removing abnormal quotes).

- NBBO files: NBBO files contain sequential NBB and NBO information, updating and creating a new row whenever either the NBB or the NBO quote changes for a given stock. These files are slightly larger than the Trade files at about 300,000 quotes per day for the 10 tickers we consider. In addition to the Quote fields of `BID`, `BIDSIZ`, `ASK`, `ASKSIZ`, and `QU_COND`, NBBO files also contain `BEST_BID`, `BEST_BIDSIZ`, `BEST_ASK`, and `BEST_ASKSIZ` fields.

4.1.2 ITCH Data

Since none of the Trade, Quote, or NBBO files from TAQ contain trade direction we obtained this information via NASDAQ's ITCH product. Having 100% accurate trade signs allows for the creation of training and test data (see Section 4.2.2), as well as a direct comparison of the performance of our models to the predictions of the LR, EMO, and CLNV methods. ITCH data comes in the form of streaming messages in binary format [29]. After converting from binary to alphanumeric characters, we're left with the messages themselves in comma-separated text format. The fields we keep are `symbol` (ticker), `nanoseconds` (timestamp in nanoseconds past midnight of that day), `buysell` ('B' if a buy, 'S' if a sell), `executed_shares`, and `execution_price`.

4.2 Data Engineering and Pre-Processing

4.2.1 ITCH Data Cleaning

ITCH data contains a variety of messages formats, including system event, stock related, add order, modify order, trade, net order imbalance indicator, and retail price improvement indicator messages [29]. These are indicated by a single alphabetic character in the `msg_type` field. We are concerned with only three types of messages: "order executed" messages (`msg_type` = E), "order executed with price" messages (`msg_type` = C), and "non-cross trade" messages (`msg_type` = P). E-type trades occur at the listed `price` of the original add order, and can be thought of as a regular trade. C-type messages apply to trades which execute at a price (`execution_price`) different than the display `price` of the original add order. Both E-type and C-type trades may be split into several orders; an example of this could be if a large market order is placed, requiring the initiator to walk up/down the book. Some C-type trades are called non-printable (marked with the field `printable` = N), and must be discarded to prevent double counting of trades [29, p. 13]. P-type messages indicate the execution of hidden orders (mentioned in Section 2.2).

Our pre-cleaned ITCH data are stored as comma-separated text files, with one text file containing messages for one ticker on one day of trading. These text files are organized by day, with one folder per trading day. Pseudocode for cleaning all ITCH files is detailed below in Figure 4.1.

4.2.2 Study Data Creation

To create correctly-signed data for training and testing models, collectively called "study data", we require four files for each day-ticker combination: the Trade, Quote, NBBO, and cleaned ITCH files. Some additional cleaning is required for the NBBO data as well. In a 2014 study of liquidity measures, Holden and Jacobsen demonstrated that the NBBO is not always complete, containing quotes which are not actually the NBB or the NBO. If the NBB and NBO come from the same exchange the NBBO file may contain one or neither of the actual best quotes [15, p. 1764]. By comparing records from the Quote and NBBO files we are able to recreate the actual NBBO. This is significant; all methods discussed in Chapter 3 rely heavily on quotes to correctly estimate trade direction. Additionally, analysis by EMO and CLNV (mentioned in Section 3.5) showed that

# Let "df" represent a Pandas dataframe containing data for one ticker on one day.
for folder in folders:

    for ticker in folder:

- Subset to trades with `msg_type` in [E, C, P]
- Drop ((`msg_type` == 'C') and (`printable` != 'Y')) trades
- Replace missing `executed_shares` and `execution_price` values with `shares` and `price` values respectively.
- `execution_price` = (`execution_price` / 10,000) # Gives decimal prices in dollars
- Keep cols [`symbol`, `nanoseconds`, `buysell`, `executed_shares`, `execution_price`]
- Create dummy `buysell_S` (1=sell, 0=buy) as new response.

    Concatenate ticker df's into a single df per folder (day), write df to CSV.

Fig. 4.1: Pseudocode for cleaning ITCH data. This process removes redundant messages, correctly replaces and adjusts values for size and price fields, and keeps columns needed to match ITCH data with the corresponding TAQ data.

trade price relative to quotes was the most valuable information used in generating accurate predictions.

In addition to constructing the complete NBBO, Holden and Jacobsen's publicly available SAS code [32] cleans the Trade, Quote, and NBBO files. The code also matches quotes from the NBBO with trades and generates the predictions of LR, EMO, and CLNV for each trade. The authors have the program divided into 13 steps; only the first 9 steps are needed for our purposes. Detailed below in Figure 4.2, the first seven steps primarily clean data while steps eight and nine assign quotes to trades and generate predicted signs.

In addition to these steps, we must match ITCH trade signs with TAQ trades to generate our study data (see Fig 4.2, step 6). Since all other work for this thesis was done in Python, including matching ITCH and TAQ trades, steps one through seven of Holden and Jacobsen's SAS code was translated into Python. Steps eight and nine were performed using SAS. It should also be noted that we removed the first and last five minutes of trading (step 1), similar to Rosenthal's approach [4, p. 400].

```
# Let "df" represent a Pandas dataframe containing one day's data.
for day in days:
```

1. Read in Trade, Quote, NBBO, and ITCH df's. Restrict NBBO and Quotes to entries between 9:00 AM - 4:00 PM, restrict ITCH and Trades to entries between 9:35 AM - 3:55 PM. Convert ITCH timestamps from nanoseconds to microseconds to match TAQ units.
2. Clean NBBO df and calculate spread and midpoint values.
3. Drop NBBO quotes that are substantially outside of large once-lagged spreads.
4. Keep only NBBO rows for which once-lagged values (quotes, sizes) are not equal to current values.
5. Clean Quotes df and calculate spread and midpoint values.
6. Clean Trades df and subset to only NASDAQ executions (`EX` = Q). Confirm that the `PRICE` and `SIZE` columns for the respective Trade and ITCH df's are in complete, simultaneous agreement. If so, append the `buysell_S` column from the ITCH df onto the Trade df.
7. Create the complete NBBO using Quotes df and NBBO df. Write out the complete NBBO df, Trades (now with signs) df, and Quotes df to CSV files.

Fig. 4.2: Pseudocode for steps 1-7 from Holden and Jacobsen's SAS code. This process cleans TAQ trades, calculates values of new fields, and matches trades to cleaned ITCH data so that they have the correct response value (`buysell_S`). Construction of training data will be complete after quotes are matched with trades and a few additional fields are calculated as described in steps eight and nine.

Rather than replicating steps eight and nine in Python, it made most sense to use the available SAS code. The prime reason for this is step 8, which matches each trade with the correct prevailing quote. Any attempt to do this in Python necessitated at least one loop over all trades, as well as performing a search in the NBBO data for every trade. In contrast, SAS's `RETAIN` statement performs this matching with ease.

To match quotes with trades in step eight Holden and Jacobsen assumed a one microsecond delay. More clearly, the NBBO entry matched to a trade will be the first NBBO entry with a timestamp of at least one microsecond less than the timestamp of the trade. One microsecond is added to NBBO quote timestamps, then quotes and trades are merged based on timestamp. The `RETAIN` statement is used to match quotes to trades. Essentially,

for each trade the values of NBB, NBO, NBBqty, and NBOqty corresponding to the nearest preceding quote are retained across data steps and are used for each trade until there is a new quote which applies to the next trade.

Step nine creates a binary variable lock (1 if NBO = NBB and 0 otherwise) and a binary variable cross (1 if NBB > NBO and 0 otherwise). Observing a crossed market is an abnormal and unrealistic condition, as a best bid greater than the best offer would have already resulted in a trade execution. Both locked and crossed markets prevent calculation of liquidity measures and confound trade signing methods which rely on the quote test. Therefore, we drop all trades with locked or crossed quotes as suggested by Holden and Jacobsen [15, p. 1750].

Additionally, step nine creates features to execute different tests. The feature $\left[\texttt{direction} = P_k - P_{k-1}\right]$ (essentially upticks, zero ticks, and downticks) is created, and then coupled with the RETAIN statement to create $\left[\texttt{direction2} = P_k - P_d\right]$ to facilitate implementation of the tick test. To allow signing via the CLNV method, $\left[\texttt{bid30} = 0.3A_k + 0.7B_k\right]$ and $\left[\texttt{ofr30} = 0.7A_k + 0.3B_k\right]$ features are created, defining the upper and lower bounds of the bottom 30% and top 30% of the spread respectively. After calculation of the final midpoint associated with each trade, all trades are signed according to the LR, EMO, and CLNV methods. This is our final study data prior to feature engineering. Table A.1 gives a complete list of features included in the cleaned study data, and Figure 4.3 shows the first few rows in one such dataframe.

It should be noted that the DATE and SYM_ROOT features are only retained for use in exploratory data analysis and feature engineering but are not included in models. Standardization of variables must be done within a single [day + ticker] combination, requiring us to keep these fields to enable this data separation. As with many models, our ultimate goal is to achieve high, out-of-sample predictive accuracy. In this case, out-of-sample data means trades of a different stock (different SYM_ROOT) occurring on a different trading day (different DATE). Therefore, these variables are excluded from all models to prevent data leakage and to more accurately simulate out-of-sample prediction. The EX column is also

TIME_M	SIZE	NBBqty	NBOqty	PRICE	NBB	NBO	midpoint	direction2	ofr30	buysell_S	buysell_CLNV
9:35:27.639672	100	200	300	30.6	30.6	30.68	30.64	nan	30.656	0	0
9:35:27.639676	100	100	300	30.6	30.58	30.68	30.63	nan	30.65	0	0
9:35:29.422193	49	200	400	30.55	30.55	30.62	30.585	-0.05	30.599	0	0
9:35:29.422197	100	100	400	30.55	30.55	30.62	30.585	-0.05	30.599	0	0
9:35:29.422201	100	100	400	30.55	30.55	30.62	30.585	-0.05	30.599	0	0
9:35:29.426187	264	200	200	30.5	30.5	30.56	30.53	-0.05	30.542	0	0
9:35:40.022919	100	100	100	30.41	30.41	30.46	30.435	-0.09	30.445	0	0

Fig. 4.3: The head of a `Pandas` dataframe containing cleaned data. Fields shown are a selection of the basic features listed in Table A.1. The data can now be explored, new features can be engineered, and models can be developed.

dropped since it offers no predictive information (it contains only a single unique value, 'Q', indicating all executions NASDAQ trades).

4.2.3 Train-Test Split

Before moving on to exploratory data analysis (EDA) we must decide how to divide our data for training and testing. As just discussed, a primary objective of our scheme will be to simulate out-of-sample performance. This implies that training and test data be disjoint in terms of tickers and dates. To achieve temporal separation, we reserve the first two days and last two days as test data and use the middle seventeen days of trading data for training and validation. This also achieves a commonly-used ratio of 80:20 for partitioning training and test data. Obtaining separation of tickers for training and test data is further discussed in the beginning of Chapters 5 and 6.

4.3 Exploratory Data Analysis

Rather than jumping immediately from data cleaning into model development, it is advisable to perform some kind of exploratory data analysis. Our general goal is to get

a feel for the data from the perspective of a financial researcher as well as an accuracy-concerned model developer. We are centrally interested in gaining insights into our data which will aid our ultimate goal of high, out-of-sample predictive accuracy. Insights which inform engineering of valuable features are of particular emphasis. For the purpose of modeling, Wickham and Grolemund suggest examining variation within variables as well as covariation between variables [33]. We might also look for anomalous or distinct patterns within data. We perform all of the above tasks through a combination of data visualization and numerical summary.

4.3.1 Financial Descriptions of Data

Our data is comprised of information for 10 tickers from the 21 trading days in March 2018. As mentioned in Chapter 4.1, this includes ITCH data from NASDAQ as well as Trades, Quotes, and NBBO files from the WRDS TAQ database. After constructing study data as described above in Section 4.2 we have a total of 653,974 trades to be used for training and testing.

We want to make the argument that our study data is a somewhat representative sample of broader markets, strengthening the case of models results generalizing well to other data. To this end, as well as to familiarize readers with some details of the examined stocks, descriptive statistics for each stock are given in Table 4.1.

To calculate the quantities given in the table we define the following attributes:

- `BIDLO`: the lowest execution price observed during the trading day.
- `ASKHI`: the highest execution price observed during the trading day.
- `PRC`: the closing execution price for the trading day.
- `VOL`: the cumulative number of shares exchanged during the trading day.
- `BID`: the closing bid price for the trading day.
- `ASK`: the closing ask price for the trading day.

- SHROUT: the number of shares outstanding for the given company. (Recorded in units of 1,000).

The fields given in Table 4.1 are calculated as follows for each individual ticker:

- Market cap (in millions of \$): $= \frac{1}{21}\sum_{i=1}^{21}\left[\frac{\texttt{SHROUT}_i \times \texttt{PRC}_i \times 1,000}{1,000,000}\right]$
- Range-based daily volatility [34]: $= \frac{1}{21}\sum_{i=1}^{21}\left[ln\left(\frac{\texttt{ASKHI}_i}{\texttt{BIDLO}_i}\right)\right]$
- Average quoted spread: $= \frac{1}{21}\sum_{i=1}^{21}\left[\texttt{ASK}_i - \texttt{BID}_i\right]$
- Estimated transaction cost (in \$): $\frac{1}{21}\sum_{i=1}^{21}\left[\frac{2(\texttt{ASK}_i - \texttt{BID}_i)}{\texttt{ASK}_i + \texttt{BID}_i}\right]$
- Average closing price: $\frac{1}{21}\sum_{i=1}^{21}\left[\texttt{PRC}_i\right]$

STUDY DATA: FINANCIAL DESCRIPTIONS						
Ticker	**Market Cap ($ MM)**	**Cap Category**	**Daily Volatility**	**Average Spread**	**Est. Transaction Costs**	**Average Price**
AAOI	525	Small	0.05088	0.0110	0.00041	26.96
BABY	1,099	Small	0.03654	0.0548	0.00166	33.15
CA	14,684	Large	0.02083	0.0100	0.00028	35.24
DAIO	68	Small	0.05812	0.0200	0.00245	8.27
EA	38,444	Large	0.02888	0.0219	0.00017	125.35
FANG	12,490	Large	0.03171	0.0257	0.00020	127.21
GABC	788	Small	0.02479	0.0243	0.00070	34.35
HA	1,870	Mid	0.03448	0.0500	0.00137	36.52
IAC	12,193	Large	0.02821	0.0705	0.00044	158.86
JACK	2,564	Mid	0.02368	0.0167	0.00019	86.84

Table 4.1: Descriptive financial statistics for the stocks sampled for study data. The ten stocks used for study data vary in market capitalization, volatility, and liquidity, acting as a somewhat representative sample of broader markets.

Note that the study data consists of a good mix of small capitalization (cap), mid cap, and large cap stocks. At the time of these trades (early 2018), large cap stocks were counted as having a market cap of \$10 billion or above, mid cap between \$2-10 billion, and small cap less than \$2 billion [35]. The stocks selected also come from a wide variety of sectors in the economy. Applied Optoelectronics (AAOI) is a telecommuncations company,

Natus Medical (BABY) produces medical devices, CA Technologies (acquired by Broadcom later in 2018) creates systems software, Data Input-Output Corp manufactures computing equipment, and Electronic Arts (EA) is a well-known video game company. Diaomondback Energy (FANG) is an oil company, German American Bancorp (GABC) is a bank holding company, Hawaiian Holdings (HA) operates an airline, InterActiveCorp (IAC) is a TV broadcasting and internet conglomerate, and Jack in the Box (JACK) is a popular fast food chain. Having a mix of small, mid, and large cap stocks from varied industries lends confidence to the idea that our models and results will generalize well to other securities and other markets.

Another metric of interest is the volume of trading for our collection of stocks during the sample period of March 2018. Figure 4.4 shows a healthy diversity in relative trading activity. This is an additional piece of evidence bolstering the claim that our models can generalize well to unseen data; in this case, unseen data of varying trading volumes. We observe a roughly proportional relationship between market cap and trading volume. Stocks with a large market cap generally have a higher volume of trade (CA, EA, and FANG), stocks experiencing more middling volume of trade tend to be mid cap stocks, and stocks with a smaller market cap have lower trade volume (DAIO and GABC). Per the work of CLNV, we would expect that the trades of stocks which are traded less frequently are easier to classify [3, p. 3819]. By characterizing the trading activity of the stocks here, we can see if this holds true during our model development by paying close attention to any abnormalities in predicting onto less-traded stocks.

Drawing from the work of LR, EMO, and CLNV, we also examine the position of trades' prices relative to quotes. From their respective work and observations (see Sections 3.4 and 3.5), we know that this might be the most important information in generating accurate predicted trade directions. Figure 4.5 represents this distribution for our study data. Training and test data have very similar distributions with a nearly identical proportion of trades occurring either above the ask or below the bid. The test data has a few more trades

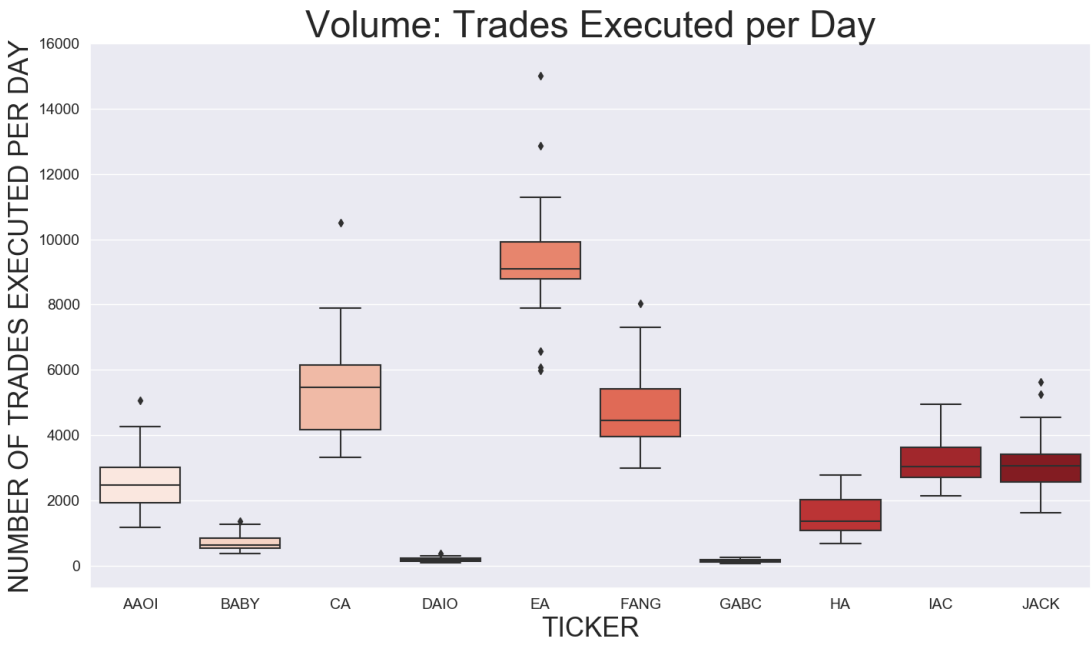

Fig. 4.4: The volume of shares traded per day for each stock represented as boxplots. A broad range of trading volumes present in study data bolsters the argument that our models will generalize well to other stocks, regardless of how often the new stocks are traded.

occurring at the quotes while the training data has a few more trades occurring inside of the quotes, but these differences are quite minor at an approximate difference of 1-3% per category.

The distribution of prices to quotes for the study data is reasonably similar to that of the data used by LR, EMO, and CLNV as summarized in Table 4.2. LR observed far fewer trades outside of the quotes (less than 1% total), about 34% each at the bid and the ask (very similar to our data), and about 30% of trades occurring inside of the quotes (also similar to our data) [1, p. 740]. Of the three older methods, the data used by EMO is the most similar to the data used in this study. Our proportion of trades occurring outside of the quotes and at the quotes is nearly identical. We end up seeing fewer trades at the midpoint (10% versus 6%) and more trades inside of the quotes which aren't at the midpoint (15% versus 22%) [2, p. 534]. On the other hand, CLNV data is somewhat different from our study data, with the concentration of trades shifted away from the quotes and a higher density of midpoint trades. Their percentage of trades at the quotes is only

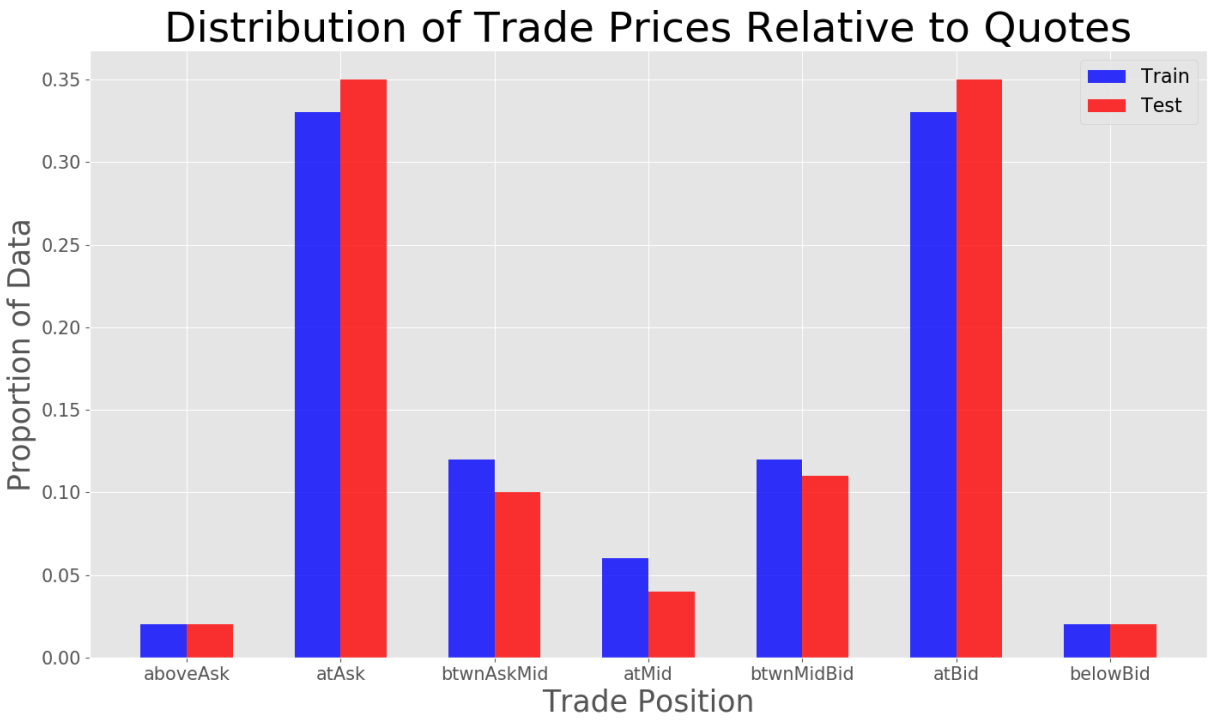

Fig. 4.5: The distribution of trade prices relative to quotes for training and test study data. The training and test data exhibit nearly identical distributions, with a large majority of trades occurring at the quotes. A small proportion of trades occur inside of the quotes, with very few of them at the midpoint. Hardly any trades execute outside of the quotes. The exact percentages represented can be found in Table A.2.

57% compared to our roughly 66%. They also see more trades outside of the quotes (about 5% on either side versus about 3% on either side), and much different behavior inside of the quotes. About 20% of their trades are at the midpoint (only 6% in our data) while the remaining non-midpoint trades are only about 13% of their sample (22% of our study data) [3, p. 3810].

4.3.2 Examination of Data for Predictive Modeling

Attempting to perform a thorough statistical EDA on all 520,000 observations of training data would be infeasible, particularly when creating visualizations. Therefore, we randomly selected a single day of training day, March 12$^{\text{th}}$, for this purpose. However, some additional pre-processing of the data is needed before further inspection. The specific scale of values for some fields differ widely across $\left[\text{day} + \text{ticker}\right]$ combinations ($\left[d+t\right]$ for short).

DISTRIBUTIONS OF TRADE PRICES RELATIVE TO QUOTES				
	LR all data	**EMO all data**	**CLNV test data**	**Our study test data**
Outside quotes	0.5%	4.83%	9.44%	4.19%
At quotes	68.4%	69.89%	57.55%	70.56%
Inside quotes	30.1%	25.21%	33.01%	25.25%
No quotes	1.0%	0.07%	0.0%	0.0%

Table 4.2: A comparison of the distribution of trade prices relative to quotes from the LR, EMO, CLNV study and our study. For the CLNV study and our study, test data was used for the given percentages. Our study data bears a strong resemblance to the data used by EMO.

For instance, the average price of trades in our sample ranges from \$8.27 per share for DAIO stock up to \$158.86 for IAC stock. The price within a ticker may vary widely too, particularly for more volatile securities. For example, the price of TSLA (Tesla) stock as of January 6, 2020 was \$451.54 and by February 4, 2020 had risen to \$901.43. These cases both illustrate the absolute necessity of standardization of variables to consider them as a single collection of data.

Standardization

Therefore, we standardize selected variables within each $[d_i + t_j]$ combination such that each variable has mean 0 and standard deviation 1. For our single day of 03/12 data that leaves $i = 12$ and $j \in \{\text{AAOI, BABY}, \ldots, \text{JACK}\}$. The list of variables for which we will standardize includes: `bid30`, `NBO`, `midpoint`, `PRICE`, `NBB`, `ofr30`, `SIZE`, `NBOqty`, and `NBBqty` (see Table A.1 for a description of these features).

After performing this within-variable standardization, we now have a single set of data containing 29,842 trades which we will use for EDA. A minuscule fraction (just 23 or 0.08%) of these trades contain missing values. The missing values are solely an artifact of the construction of the `direction2` feature. Recall that this feature is the difference between the current trade price P_k and the nearest preceding trade at a different price P_d. Thus, the first trade of each ticker cannot have a `direction2` value since there is no trade

preceding it. Also, if the first trade price of the day is repeated for consecutive trades then none of these repeats will have `direction2` values either. This can then potentially impact the outputs of the LR, EMO, and CLNV methods since their classifications are often determined by the tick test, where `direction2` is a numeric descriptor of downticks and upticks as described in Section 3.2. In total, of the 23 trades that have missing values for `direction2`, 10 of those trades having a missing value for `buysell_EMO` and one trade has a missing value for `buysell_CLNV`. Since they comprise such a negligible proportion of our data, we simply drop these observations leaving 29,819 trades.

Class Imbalance

An elemental consideration in binary classification problems is the relative balance of observations in each class. In cases such as fraud detection or disease diagnosis, the class of interest (fraudulent activity, disease presence) is greatly outnumbered by observations in the other class (regular activity, no disease), sometimes at ratios of 100:1 or greater. Methods for rectifying class imbalance to improve model performance are well-studied ([36], [37], [38], [39]), thus we should determine if such an issue is present in our data before developing models.

Our training data and test data share an identical ratio of buy to sells, with 58% of trades being buys and 42% being sells. This is a fairly balanced distribution of the two classes. However, trading behavior specific to certain tickers may affect their relative proportions of the two classes, potentially making their trades more difficult to classify. Figure 4.6 below illustrates this clearly.

We can see that the median proportion of trades which are sells is roughly around 42% for most tickers. The DAIO and GABC tickers show a distribution shifted substantially below that of the other tickers, having more buys than sells. The BABY and HA tickers appear to be the opposite, with somewhat more sells than buys. We may consider rectifying these class imbalances within tickers when constructing training data to see if it noticeably improves performance.

Variable Correlations

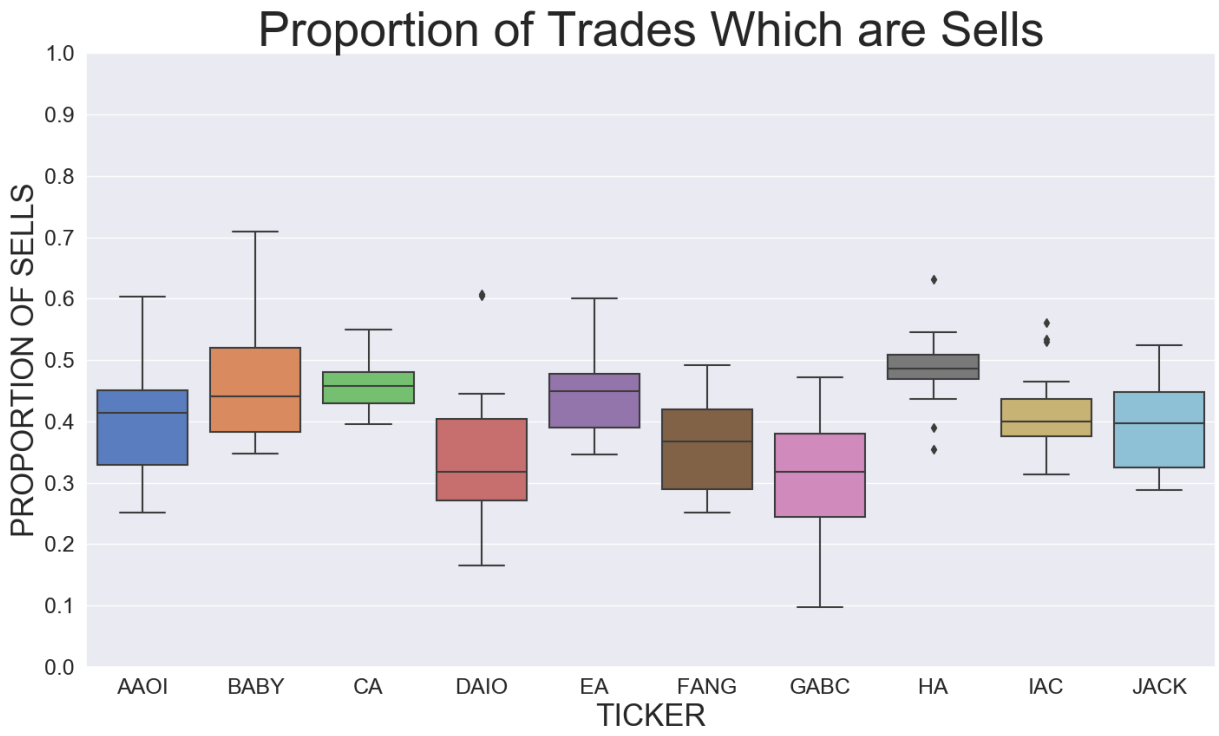

Fig. 4.6: The proportion of trades which are sells for each stock, represented as boxplots. The median proportion of sells for most of the ten stocks is fairly near the overall 0.42 proportion of sells. A few stocks exhibit trading days with far more sells than this (BABY or HA) while other stocks exhibit trading days with far fewer sells than this (DAIO or GABC). Although this graphic does not indicate a severe class imbalance we will explore this during model development.

Wickham and Grolemund suggest examining covariation between variables as a basic consideration in EDA [33]. This may lend insight into which variables are most likely to aid in correct prediction as well as which variables contain similar information. A heatmap of variable correlations is provided in Figure 4.7.

From this heatmap we can observe three distinct groupings of variables: analogues for `microsTIME`, `buysell_S`, and `std_PRICE`. The `microsTIME` and `std_PRICE` groupings all exhibit nearly perfect correlations (see Tables A.3 and A.5). The analogues of `buysell_S` are still noticeably correlated, but not to the same extent. This is actually encouraging; if any of the LR, EMO, and CLNV methods were perfectly correlated with the `buysell_S` variable this would imply they can perfectly predict trade signs. Instead, they do a fairly good job, but still leave room for improvement via modeling. Also, the respective outputs

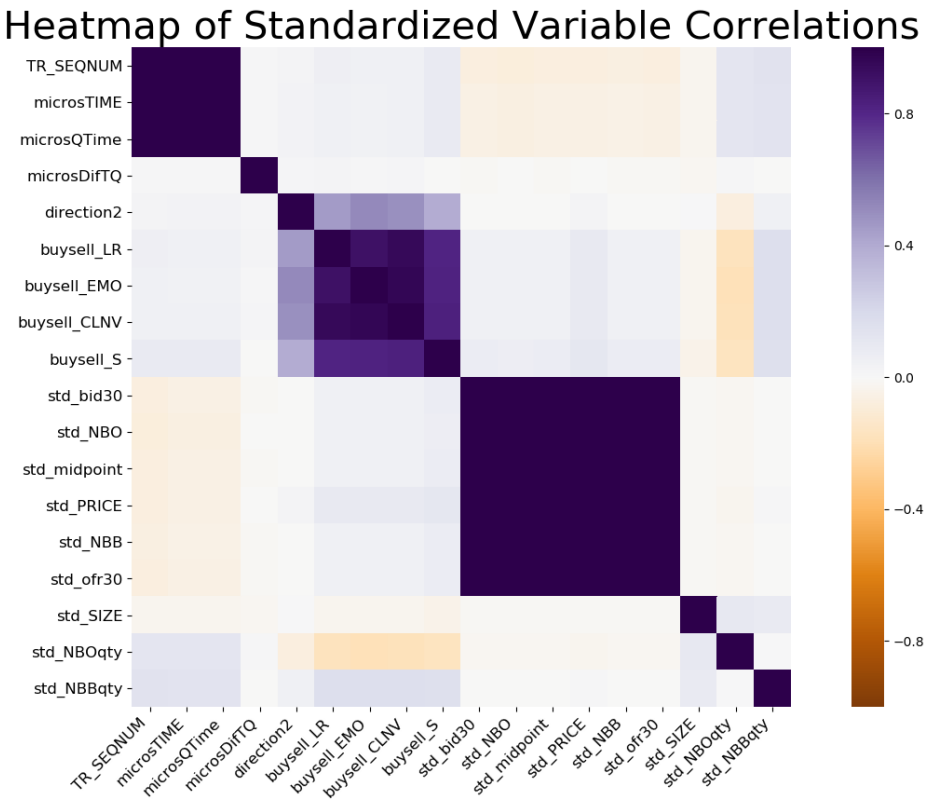

Fig. 4.7: Correlations of standardized variables. Notice the three distinct groupings of `microsTIME`, `buysell_S`, and `std_PRICE` analogues. The numerical pairwise correlations for these groups are found in Table A.3, A.4, and A.5 respectively. Although `SIZE` variables are also analogues of each other they exhibit hardly any correlation.

of the LR, EMO, and CLNV methods aren't perfectly correlated (see Table A.4), which will allow models to use their values as predictors for observations with different profiles.

KDE Plots Colored by Trade Direction

We know from prior studies that the tick test is a crucial predictor of trade direction. Since the `direction2` feature encodes tick values ($P_k - P_d$) used in the tick test, it will likely prove vital for good prediction. After removal of a few outliers (less than 0.4% of the data), we create overlaid KDE plots of this feature for buys and sells as seen in Figure 4.8. We observe a pronounced bifurcation between the distribution of buys and sells according to `direction2`: 77.5% of buys have value less than 0 and 84.3% of sells have a value greater than 0. By construction, no trades have a `direction2` value equal to 0, therefore we can visually apply the tick test via the green dotted line, resulting in an overall accuracy

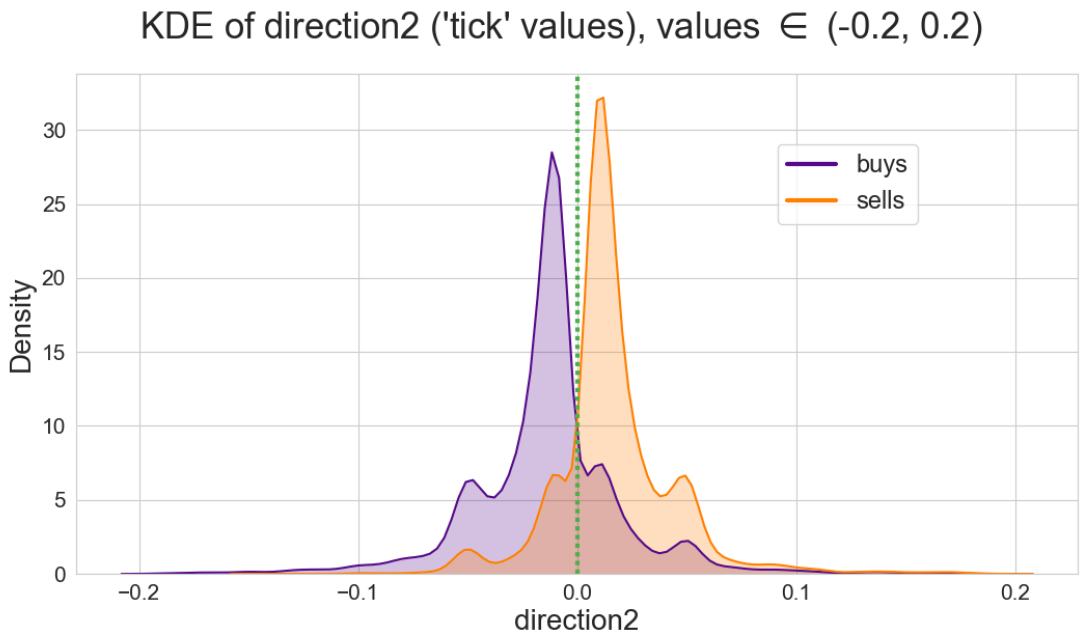

Fig. 4.8: Overlaid KDE plots of `direction2` (non-zero tick) values for buys and sells. Kernel density estimates for each distribution were smoothed with bandwidth parameters of 0.006. A `direction2` cutoff of 0.0, which implements the tick test, decently classifies trades. However, a non-trivial number of trades are misclassified by this simple rule.

of 80.3%. This is encouraging for a couple of reasons. One, we know that this variable contains valuable information and will surely be a valuable predictor for our models. Two, we can see that there is still a non-trivial amount of overlap between the two distributions, meaning that our models can significantly improve upon current methods such as this one.

Another feature worth examining is `SIZE` (now `std_SIZE`), as EMO found that larger trades are more difficult to classify [2, p. 536] and CLNV found that the tick rule performs better on larger trades [3, p. 3815]. After subsetting to trades with `std_SIZE` values less than 5 (leaving 99.6% of the data) we construct the KDE plot seen below in Figure 4.9. From this figure we can see that there is essentially no difference between the distribution of buys and sells relative to trade size. Unfortunately it appears that trade size makes trades more difficult to classify, but we can't directly use this feature to fix the problem and accurately separate buys and sells.

In this same vein, we also investigate the `microsDifTQ` feature. The name of this features comes from taking the difference (`Dif`) in microseconds (`micros`) between the

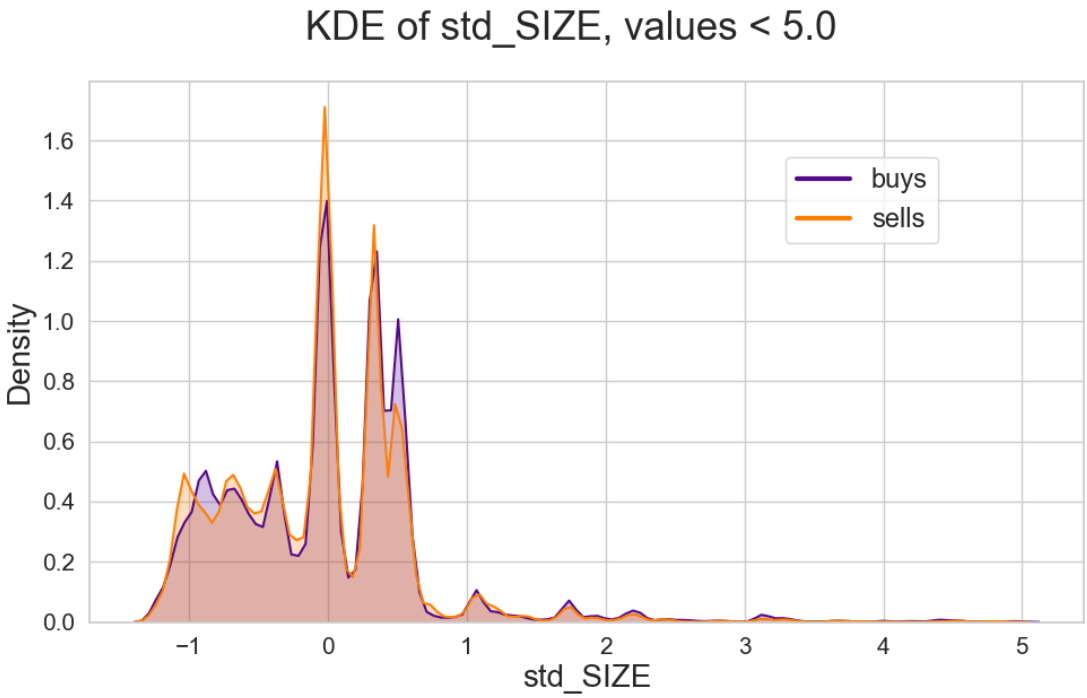

Fig. 4.9: Overlaid KDE plots of `std_SIZE` values for buys and sells. Kernel density estimates for each distribution were smoothed with bandwidth parameters of 0.05. The EMO and CLNV studies cite trade size as impacting how easily a trade is predicted; unfortunately, we see no discernible difference between `std_SIZE` density estimates of buys and sells.

trade time (`T`) and the quote time (`Q`). This is also known as quote distance. Again, we remove a small amount of outliers (here, just 0.1% of the observations) by restricting our data to trades with a `microDifTQ` value less than 100, and then construct the plot in Figure 4.10. As with the `std_SIZE` feature above, we can see no appreciable difference between the distribution of buys and sells relative to quote distance. If anything, there may be a slight shifting of the distribution of buys upward on the scale, particularly for larger quote distances. However, this difference is subtle enough that it gives little additional information which we, as human model developers, can incorporate into our choices of feature engineering.

Dimensionality Reduction Visualization via PHATE

Another EDA tactic for finding patterns in data is to perform dimensionality reduction and visualize the transformed data in a lower-dimensional space. Some forms of this include the use of biplots to represent the first two principal components in prinicpal components

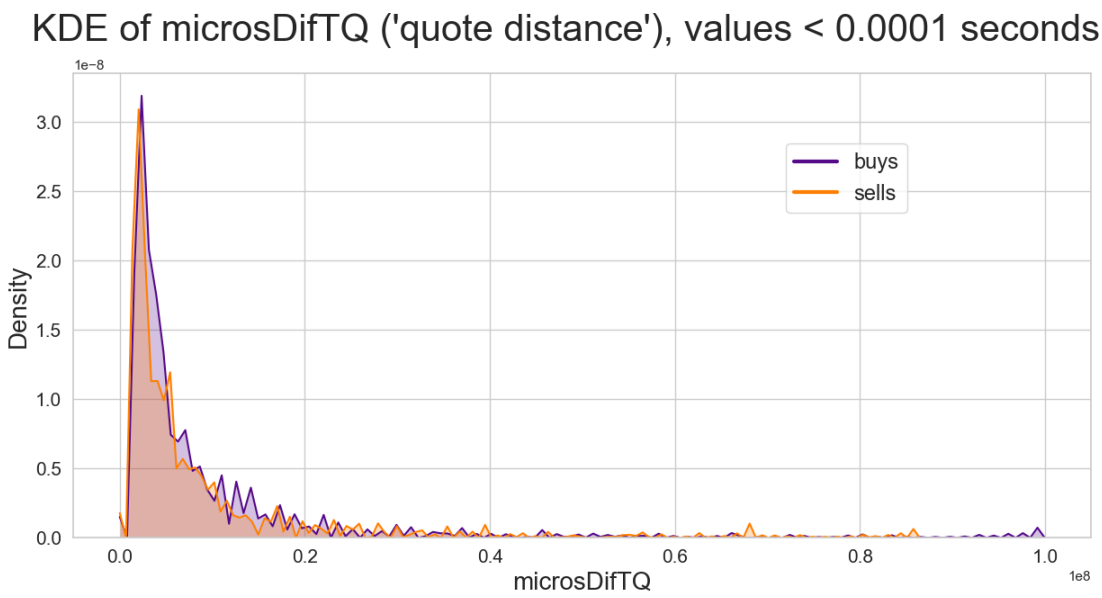

Fig. 4.10: Overlaid KDE plots of `microsDifTQ` (quote distance) values for buys and sells. Kernel density estimates for each distribution were smoothed with bandwidth parameters of 0.004. Similar to `std_SIZE`, we see no discernible difference in the density estimates of buys and sells.

analysis (PCA) [40], t-SNE [41], and PHATE [42]. Thanks to the improvements which PHATE has made upon PCA and t-SNE, including preservation of global and local structure as well as data de-noising, we have used it to generate Figures 4.11 and 4.12. The first two dimensions of PHATE are plotted to capture the most significant variation and structure within the data. From Figure 4.11 we see no discernible groupings of any of the tickers. If this were the case, it would be an indication of trades for a ticker (and their corresponding input vectors) being fundamentally different from those of other trades, making them harder to classify and our models less generalizable. In Figure 4.12 we have the same plot but colored by trade directions. The fact that buys and sells show no apparent difference is also somewhat of a good thing; if they were easily separable in just a two-dimensional representation then the use of sophisticated modeling and feature engineering would be unnecessary, and the problem would be trivially easy in some sense. Since the data show intricate global and local structure, a model more sophisticated than reductive decision rules will very likely be needed to characterize these intricacies and achieve reliable discriminatory

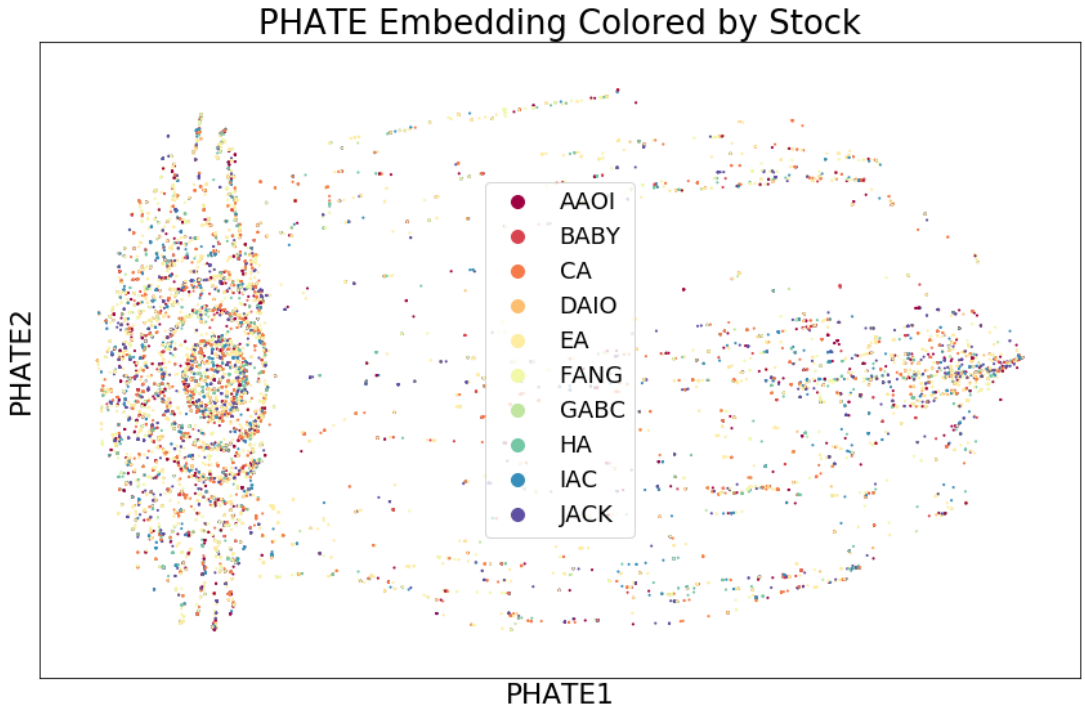

Fig. 4.11: The first two PHATE dimensions of March 12^{th} data, colored by ticker. The fact that trades from individual tickers do not group together is encouraging, indicating that trades represented in a low-dimensional space are generally similar to each other regardless of ticker. This aids the argument that our models will generalize well to unseen trades.

power between the two classes.

4.3.3 Insights Gained from EDA

We have found compelling evidence that our data is fairly representative of broader markets, comprised of stocks from a variety of sectors with diverse market caps and trading volumes. Additionally we've shown where our trades lie relative to the quotes; if the findings of past studies hold here, we now know we should concentrate efforts on correctly classifying the 28% of trades which lie inside of the quotes. We also observed that some stocks exhibit an imbalance between the number of buys and sells executed, suggesting that sampling training data to create class balance representative of the broader study data may help improve test accuracy. Finally, we recognize that most features considered alone do a poor job of discriminating between buys and sells. We will need to invest significantly in

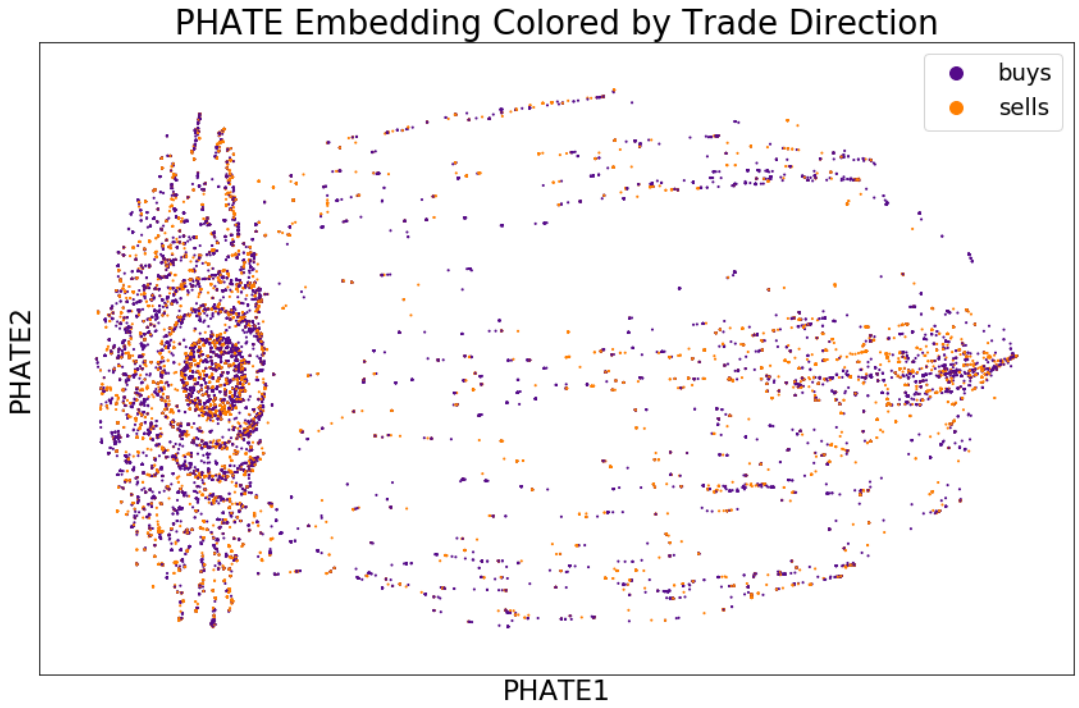

Fig. 4.12: The first two PHATE dimensions of March 12^{th} data, colored by trade sign. The homogeneous mixing of buys and sells illustrates that we are not able to predict sign based on a low-dimensional, continuous representation of the data.

feature engineering to capitalize on the value of our data to create representations which will work best with different models. Armed with these insights we will now proceed to create predictive models.

CHAPTER 5

PREDICTION METHODS

5.1 Modeling Methods Used

The end product of model development will be a highly-accurate, out-of-sample classifier of trade signs created with the specific goal of outperforming the results of LR, EMO, and CLNV. To this end, as well as for thoroughness and comparison's sake, we will employ eight different binary classifiers to see which approach works best: logistic regression [43] , quadratic discriminant analysis (QDA hereafter) [44, p. 110], the k-Nearest Neighbors classifier (k-NN hereafter) [45], decision trees [46], random forests (RF hereafter) [47], gradient-boosted trees [48], support vector machines (SVM hereafter) [49], and a feedforward neural network [50].

Since the theory, mathematics, and computation underlying these methods are well-documented, we will simply address relevant considerations from the standpoint of a practitioner. All models were developed using Python's `scikit-learn` library [51]. The one exception to this was the neural network model which was developed in Python using the `PyTorch` library [52].

Logistic regression and QDA computed quickly and were fantastically easy to implement. Neither method required any tuning since we used basic logistic regression without any regularization. The k-Nearest Neighbors (k-NN) classifier was not far behind in ease-of-use, only requiring tuning of k via cross-validation on training data.

Decision trees were the easiest to tune of the tree-based methods, only requiring us to adjust the `max_depth` argument as all other hyperparameters had negligible effects on accuracy. RF were not much harder, requiring adjustment of `min_samples_split`, `max_features`, and `n_estimators` (see Section 5.4.4 for more in-depth discussion of tuning for RF). We employed the `XGBoost` [53] library to create an ensemble classifier of gradient-

boosted trees. Consequential tuning parameters included `max_depth`, `learning_rate`, and `n_estimators`. By setting the `n_jobs` argument for RF and XGBoost to -1 computation time was dramatically reduced.

While RF and XGBoost were trained in a reasonable amount of time, creating SVM models took much longer due to their computation being on the order of $O(N^3)$ (where N is the number of training examples) [54]. Tuning included trial of four different kernels: linear, polynomial, radial basis function, and sigmoid. Hyperparameters C, γ, and degree of the polynomial kernel were also adjusted where applicable.

Although SVM models took quite some time to tune, the neural network models were the most labor-intensive by far. Architectural design included exploration of the number of hidden layers, number of neurons in each hidden layer, and activation functions used. Additionally, regularization of the loss function, dropout, and batch normalization were tested. Other trials included the use of different optimizers and tuning of their respective hyperparameters.

5.2 Model Development Overview

As mentioned in Section 4.2.3, we have split our data into a training and test set. The first two days and last two days of data (March 1, 2, 27, and 28) are withheld for use as test data while the middle seventeen days of data (March 5, 6, ..., 26, 27) are used for training and validation. These middle seventeen days will be referred to as just "training data" with the implicit understanding that portions of this data will be used for validation within this chapter.

Two stages of model development lead to the selection of final models for prediction onto test data. In the first stage, we create a cursory baseline model and then assess which of the eight methods listed above show promise and deserve further exploration. March 12 and March 27 data were randomly selected as training and validation sets for this initial stage, containing 30,753 and 33,911 trades respectively. The second, and final, stage of model development involves additional feature engineering, feature selection, examination of the effect of class imbalance, and hyperparameter tuning. Training data for stage two is

comprised of the 109,655 trades from March 9, 13, 15, and 20, and the 63,498 trades from March 8 and 22 constitute the validation data.

A variety of metrics exist for quantifying the performance of binary classifiers, including recall (sensitivity), precision, specificity, and F1 score. However, the chief concern of this work is to accurately sign trades, making percent correctly classified (PCC) the metric which ultimately matters. Therefore, PCC (or just "accuracy") will be the measure used to appraise performance and make decisions regarding model development.

5.2.1 Assessment Schemes

To facilitate a valid comparison of models, two schemes of assessment will be used. These schemes are depicted in Figure 5.1 below. The blue highlighting illustrates what

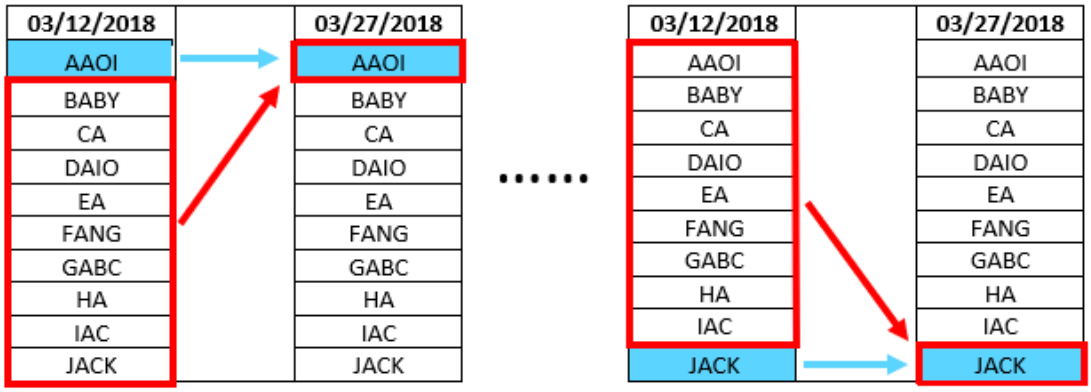

Fig. 5.1: Depiction of same-ticker validation and all-but validation using 03/12/2018 data for training and 03/27/2018 data for validation. The all-but validation scheme is of primary importance since it represents out-of-sample prediction by using training and validation data which are disjoint both temporally and in terms of tickers.

we will call "same-ticker" validation, while the red highlighting illustrates what we will call "all-but" validation. Same-ticker validation simply means that we build the model using training data from a single ticker and predict onto the validation data of that same ticker. All-but validation occurs when we train on the data from nine tickers and predict onto the validation data from the omitted tenth ticker. Both schemes result in ten different validation accuracies, one for each ticker.

We intuitively believe that data within a trading day contains similarities, irrespective of ticker. For example, trades from a day during the thick of the Great Recession, regardless of ticker, likely contained far more sells than buys with prices declining as markets dropped precipitously. We also suspect that data within a ticker contains similarities, irrespective of trading day. A simple example is a very stable security, such as a utility stock, versus a more volatile security like a tech startup. We would expect a much lower volume of trading and less price movement in the utility stock than we would in the tech stock. Due to these dynamics, the all-but validation accuracies are the most important since they represent out-of-sample prediction in terms of both securities and trading days. To generate an aggregate validation PCC, we must weight the all-but accuracies according to how many observations are in the validation set for each ticker. We will define weighted-all-but PCC (wab-PCC), or all-but accuracy, as:

$$\text{wab-PCC} := \left[\frac{\sum_{i=AAOI}^{JACK} (length_i)(\text{all-but PCC}_i)}{\sum_{i=AAOI}^{JACK} (length_i)} \right]$$

where $length_i$ is the number of observations for ticker i in the validation data set and all-but PCC_i is the all-but validation accuracy for ticker i.

We will determine which methods to further pursue based on wab-PCC. We will also compare overall validation accuracies of LR, EMO, and CLNV to wab-PCC values to determine if models are improving upon these older decision rules' results. Analysis of the ten individual all-but PCC values and the ten respective accuracies of LR, EMO, and CLNV will indicate stocks whose trades are difficult to classify.

5.3 Stage 1: Initial Model Prototyping

5.3.1 Baseline Model

Prior to model development we should determine some rough baselines to inform our expectations and findings later on. To do this we will perform the minimal amount of feature

manipulation needed to generate predictions. Variables such as `PRICE` and `SIZE` differ in scale across [day + ticker] combinations so we must standardize them within each [day + ticker] combination before training and validation. This leaves us with ten standardized variables as well as the trade time feature, quote time, quote distance, `direction2`, and the predictions of the LR, EMO, and CLNV methods as predictors. Due to the reputation of RF working well right out of the box [55, p. 167], we construct a baseline RF model with 100 trees and all other parameters set to `scikit-learn` defaults. Figure 5.2 illustrates the performance of the baseline RF model against the results of older methods. We can see that

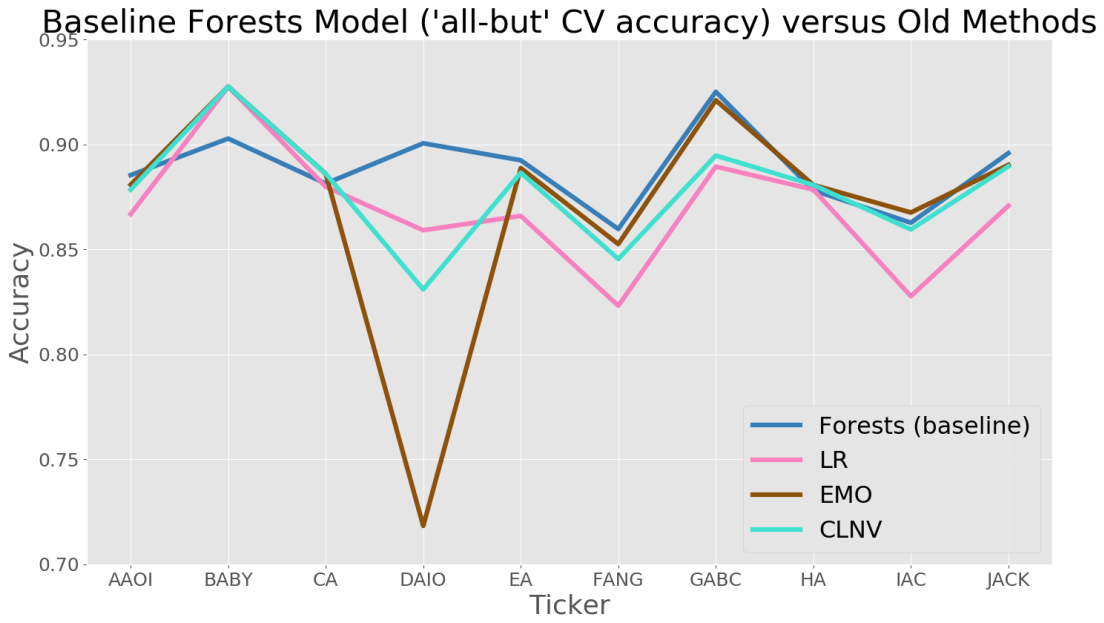

Fig. 5.2: All-but validation accuracy of baseline RF model (only original features and scaled features) plotted against the accuracies of the LR, EMO, and CLNV methods. The performance of RF is nearly indiscernible from the results of the older decision rule methods. RF does somewhat better on DAIO trades, but does the same or worse than LR, EMO, and CLNV for the other nine tickers.

RF yields negligible, if any, improvement over the older methods. The wab-PCC of RF is 88.2%, while LR achieves 86.0% accuracy, EMO reaches 88.0% accuracy, and CLNV comes in at 87.7% accuracy. Based on these results we know that generic application of existing models will not be sufficient to achieve our desired results of substantially outperforming

the LR, EMO, and CLNV methods. For this reason, we turn to a variety of supplementary techniques for enhancing the predictive accuracy of our models.

5.3.2 Initial Feature Engineering

In addition to using sophisticated machine learning methods, our approach differs substantially from others in how we exploit data. Specifically, extensive use and exploration of feature engineering is a key contribution of this work and its strong results. For any who may be unfamiliar with the term feature engineering, it simply means that we use values of existing features to create new features. A simple example would be the midpoint of the quotes. We originally have A_k and B_k for each trade and can engineer a new M_k (`midpoint`) feature where $\left[M_k = \frac{A_k + B_k}{2}\right]$, using the newly-calculated midpoint as another predictor.

The primary reason we engineer new features is to increase the accuracy of our models. Rather than gathering additional data, we can greatly enhance models' predictions simply by representing the data which we already have in a different manner [56]. This is a crucial part of generating high-quality machine learning models, but can often be specific to a discipline or problem being addressed. Therefore, we will use a few generally accepted approaches but will particularly look to prior works and our own deep understanding of the data-generating process to engineer new features.

Two examples of feature engineering from previous works were particularly influential in our approach. As described in Section 3.6, Rosenthal used a proximity function to encode prices and quotes into information metric predictors. Additionally, he created once-lagged versions of these same predictors [4]. Another example of feature engineering is the creation of the `bid30` and `ask30` features from the CLNV method (see Section 3.5). These two features respectively represent the boundaries of the bottom 30% and top 30% of the spread and are simply calculated by using A_k and B_k for each trade [3].

While the specifics of these two examples are noteworthy, their primary utility was inspiring the general idea of contemporaneous features and lagged features. We can delineate these two types of features by simply thinking in terms of indices. If data are organized in a table with each row representing a single trade and each column representing a field like

`PRICE` or `NBO`, then contemporaneous features use only values with the same index in their calculation. The `bid30` and `ofr30` features are examples of this. Both of them use only A_k and B_k with index k in calculation of their values. On the other hand, if the calculation of a new feature involves any values not having the same index, then we can consider it lagged.

The inspiration for contemporaneous features is clear: order book conditions at execution convey substantial information about a trade being a buy or a sell. The quote test is a clear example of this. Depending on where P_k is relative to A_k and B_k we can somewhat reliably classify whether the trade at time k is a buy or sell. Alternatively, lagged features help us to characterize predictive relations across time. The tick test is a clear example of this. Knowing the price of the previous differing trade offers substantial information, evidenced by the tick test predating LR as a trade signing method. To summarize, contemporaneous features capture information within trades and lagged features capture information across trades.

Armed with the idea of contemporaneous and lagged features, we began by creating the following engineered feature types:

- **scaleOneVar(x)**: the new column is $\left[\frac{x_i - median_x}{IQR_x}\right]$.
  This name is shorthand for the robust-scaled values of the variable x. We use robust scaling since some of our features contain outliers. In performing traditional standardization within a variable we subtract the mean from each observation and divide by the standard deviation. Here, we subtract the median of the variable and divide by the IQR (the difference between the 75^{th} and 25^{th} percentiles of the variable). Since the median and IQR are based on percentiles, they are more robust to outliers.

- **nominalDiff(x, y, n)**: the new column is $\left[x_i - y_{i-n}\right]$.
  This name is shorthand for the nominal difference between the values of variables x and y where n is the number of indices by which the y variable is lagged. If x and y are the same variable then we must have $n > 0$, creating a lagged feature. If x and y are different and $n = 0$ then we have a new contemporaneous feature.

- **scaledDiff(x, y, n)**: the new column is $\left[\frac{x_i - y_{i-n}}{y_{i-n}}\right]$.
  This name is shorthand for the scaled difference between the values of two variables. A scaledDiff feature is much the same as a nominalDiff feature, with the only difference being that the nominalDiff numerator is now scaled by the y variable below.

- **colsAgree(x, y)**: the new column is $\left[x_i == y_i\right]$.
  This name is shorthand for the agreement (or lack thereof) of the values of x and y. We represent true values as 1 and false values as 0.

- **lagVar(x, n)**: the new column is $\left[x_{i-n}\right]$.
  This name is shorthand for lagging the values of variable x by n indices to create a new column.

Features which incorporate a non-zero n argument will have n missing values at the start of the column for the new feature. We simply drop these observations since they represent less than 0.01% of our data. We can also use new features to create additional new features. An example of this is applying lagVar(`buysellLR`, 1), creating a new column of once-lagged LR predictions. We can then apply colsAgree(`buysell_LR`, `buysell_LR_lag1`) to create a column indicating if the current trade has the same LR prediction as the previous trade. Another point to recognize is that it only makes sense to engineer new variables using existing variables which are analogues of each other. As pictured in Figure 4.7, these groups include `PRICE`, `SIZE`, `buysell`, and `microsTIME` analogues. It would make no sense to apply nominalDiff(`PRICE`, `SIZE`, 0) since the price and size of a trade are on completely different scales and are uncorrelated. Sixty eight features were created and ubiquitously used for prototyping new models. Nearly all combinations within each of the four analogue groups were attempted, including lags one and two for many variables. Due to its length, the complete list of variables has been omitted. However, during discussion of feature selection we will mention specific variables and provide a final listing in Tables A.6 and A.7.

5.3.3 Initial Results

Using data from 03/12/2018 for training and 03/27/2018 for validation, models for each of the eight methods outlined in Section 5.1 were created for each of the all-but and same-ticker validation schemes. The logistic regression, QDA, RF, and k-NN models all required no tuning or were auto-tuned using resubstitution or out-of-bag (OOB) error, with other values left at defaults. The `max_depth` parameter for the decision tree model was tuned using all-but validation, the optimal value being seven (see Figure 5.3). The XGBoost, neural network, and SVM models all required extensive tuning; optimal hyperparameters were found using grid searches. The architecture of the neural network, including number of layers, neurons in each layer, number of training epochs (see Figure 5.4), and activation functions used were adjusted manually.

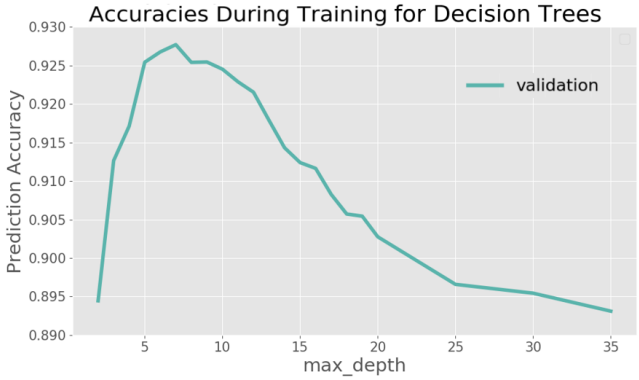

Fig. 5.3: Tuning `max_depth` parameter for the decision tree model. The highest validation accuracy occurs at a `max_depth` value of seven. We use a `max_depth` value of seven for all subsequent decision tree models.

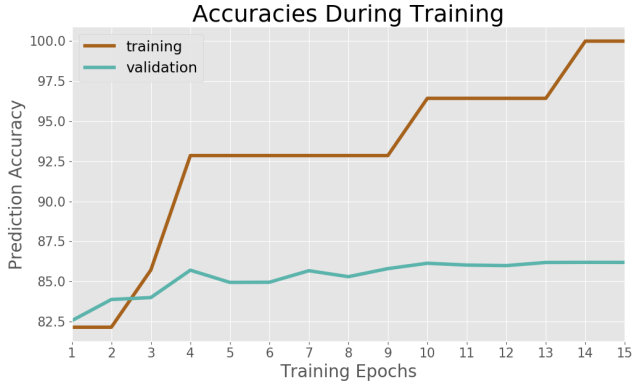

Fig. 5.4: Determining the minimum number of training epochs for the neural network model. Although training accuracy rises to 100%, validation accuracy stabilizes around 86.5% after 10 epochs. We use 15 epochs for all subsequent network models.

The optimal XGBoost model had a `learning_rate` of 0.05, a `max_depth` of 3, and `n_estimators` = 100 (100 decision tree base learners). Tuning of other parameters for XGBoost yielded no appreciable increase in validation accuracy. Trained on normalized inputs, the optimal SVM model used the simple linear kernel and a `C` value of 0.01. The optimal network model, pictured in Figure 5.5, is a fully-connected feedforward network

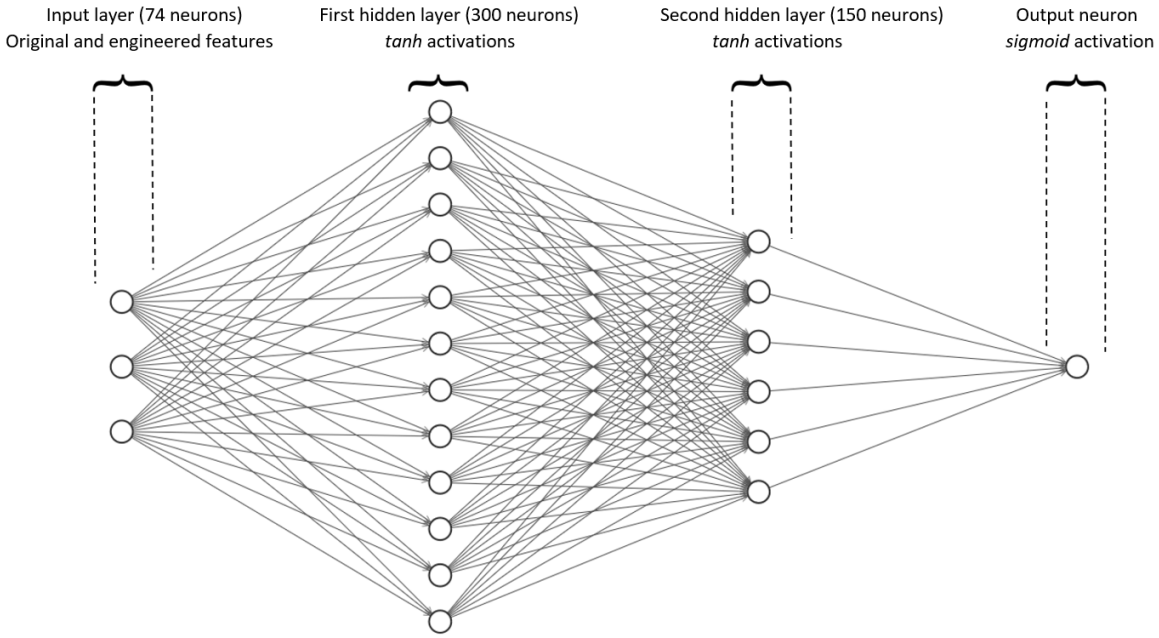

Fig. 5.5: Diagram of the neural network model. The number of neurons in the input and hidden layers have been reduced by a factor of 25 to allow for visualization. The architecture employed is a fully-connected, feedforward network which uses backpropogation to learn weights and biases.

having one hidden layer with 300 neurons, a second hidden layer with 150 neurons, and a single output neuron. A binary cross-entropy loss function is used, the output neuron uses a sigmoid activation function, and all neurons in the hidden layers use the *tanh* activation function. The Adam optimizer slightly outperformed stochastic gradient descent with a final `learning_rate` of 0.001. Dropout and regularization only degraded validation accuracy, while batch normalization yielded a jump in accuracy from about 65% up to 82% before fine-tuning of other parameters. Validation performance in Figure 5.4 illustrates that very few epochs are needed to achieve the asymptotic accuracy of this network. Based on this result, all iterations of the neural network model were only trained for 10 epochs.

The all-but validation results for tuned prototype models are seen below in Figure 5.6 and same-ticker validation results are depicted in Figure 5.7. We note that there are four models which do particularly well in both schemes: XGBoost, RF, decision trees, and the neural network. Most of them have an average validation accuracy in the neighborhood of 90%, with same-ticker accuracies unsurprisingly exceeding all-but accuracies. Since RF

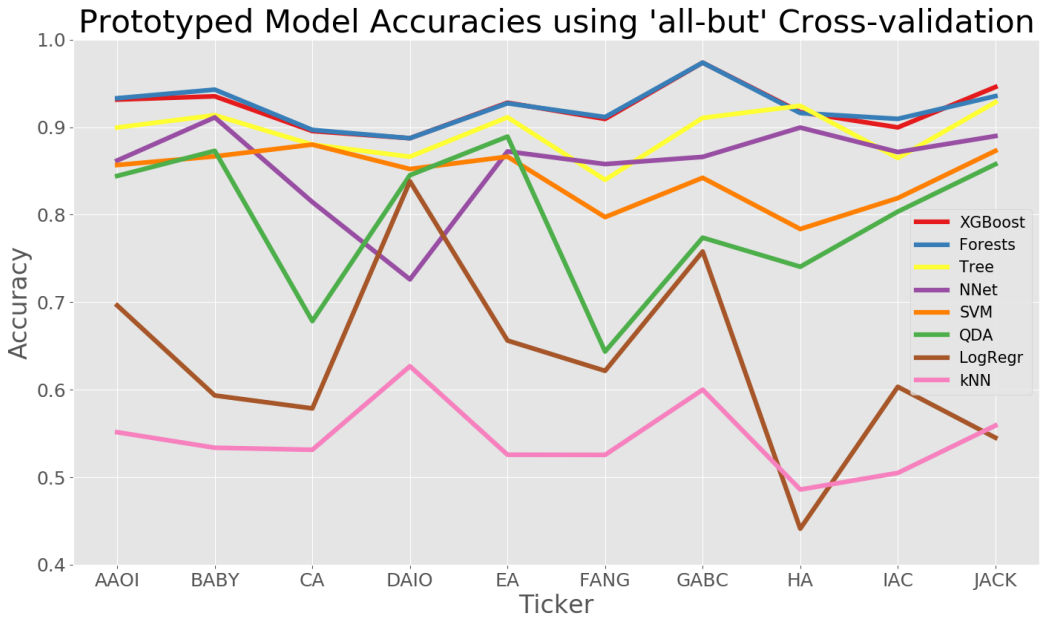

Fig. 5.6: All-but validation accuracies training on 03/12 data and validating on 03/27 data. Simpler methods fared poorly while tree-based methods achieved the best results. RF and gradient-boosted trees performed the best, obtaining nearly identical results.

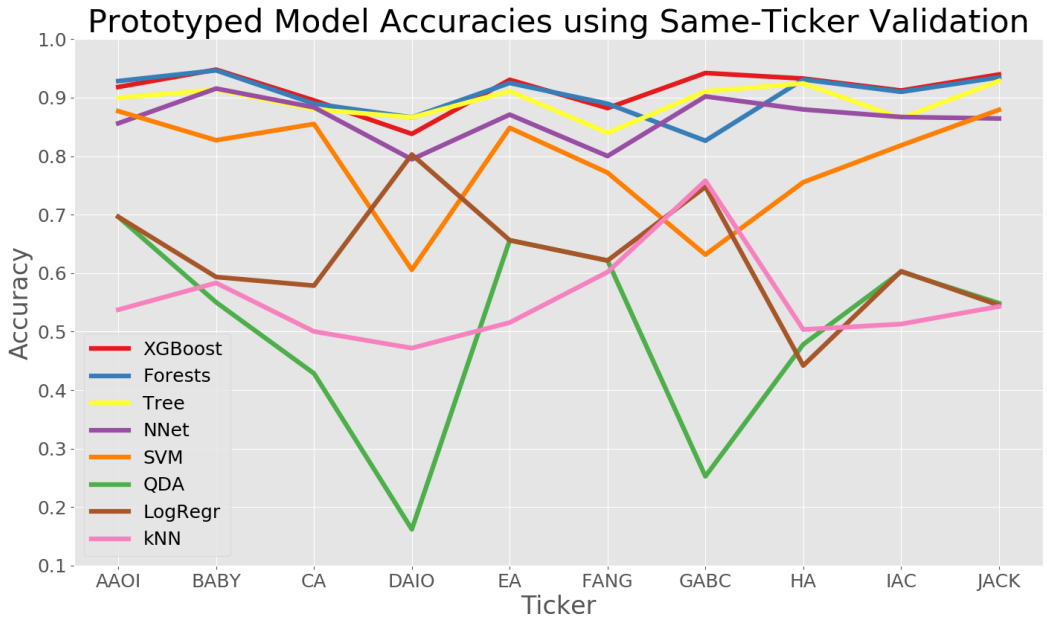

Fig. 5.7: Same-ticker validation accuracies training on 03/12 data and validating on 03/27 data. The results here are similar to those of all-but cross-validation. Simple methods did poorly and tree-based methods achieved the highest accuracies. The neural network model also showed improvement for this validation scheme.

requires little tuning and either outperforms or equals XGBoost and decision trees, we will only examine RF for further development of tree-based approaches. QDA actually fared decently for being a simple method, even outperforming the neural network for the DAIO and EA trades in the all-but scheme. Therefore, we will continue developing the RF, QDA, and neural network models. Comparing same-ticker and all-but results shows that the neural network improved substantially on DAIO prediction when training on other DAIO trades, jumping from 72.6% up to 79.6% accuracy. We will see why this is the case when examining the results of the older methods in Figure 5.9.

Although we won't be specifically developing the decision tree model any further, a look at its topmost splits in Figure 5.8 is still informative. This is the decision tree for

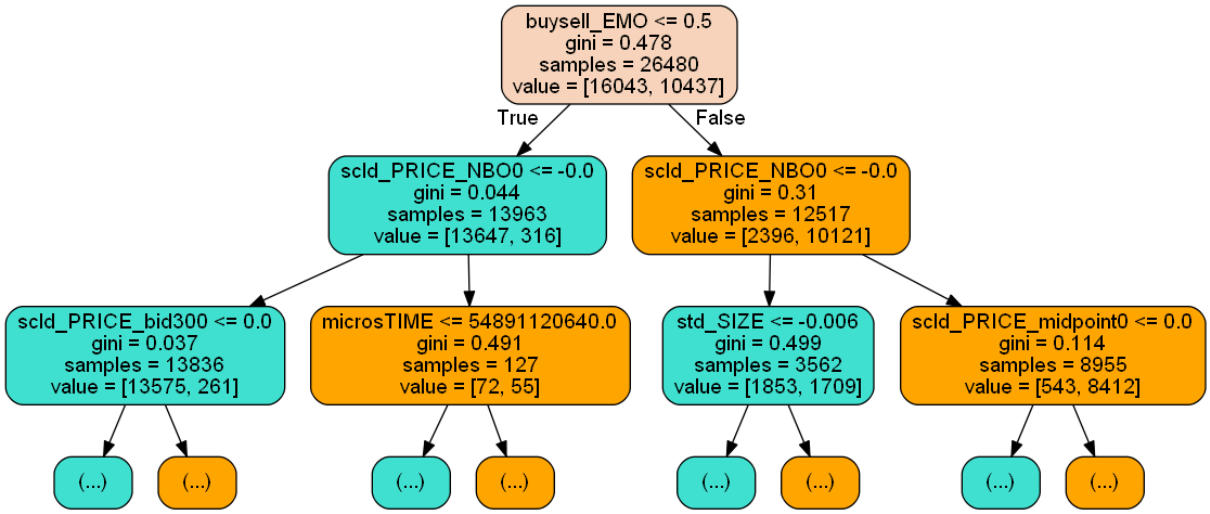

Fig. 5.8: The first three levels of splits for the all-but BABY decision tree model. The predictions of the EMO method prove to be the most valuable split, achieving 89% accuracy. Subsequent splits tend to be upon features which are scaled differences involving `PRICE` and one of its analogues, such as `NBO`, `midpoint`, or `bid30`.

the all-but model where BABY serves as validation data. The respective decision trees for all ten tickers were examined and this one was the most representative. It shows that the predictions of the EMO method do quite a good job, achieving $(13647 + 10121)/(13647 + 316 + 2396 + 10121) = 89.8\%$ accuracy in the first split. The all-but BABY accuracy of the decision tree model in Figure 5.6 is 91.4%, meaning that further splits yield an incremental

improvement of 1.6 percentage points. Of the remaining splits, engineered features which contemporaneously relate `PRICE` to other price analogues dominate. This is an indication that the position of a trade's price relative to the rest of the spread is valuable information for enhancing prediction.

Comparing the selected QDA, RF, and neural network models to older methods in Figure 5.9 it is clear to see that RF substantially outperform all older methods. This is

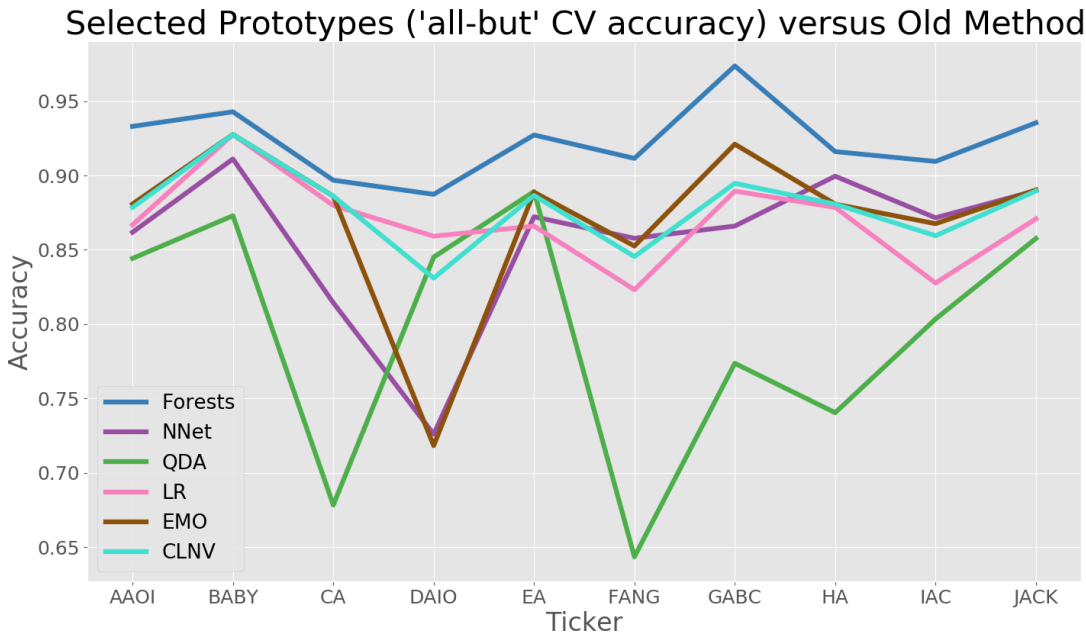

Fig. 5.9: Prototyped all-but accuracies of the QDA, RF, and neural network models versus the LR, EMO, and CLNV decision rule methods. QDA performs quite poorly and the network model achieves similar results to the LR, EMO, and CLNV methods. The RF model outperforms all other models shown here, indicating that feature engineering has helped to substantially improve performance.

a very promising result relative to the outcome of our initial baseline model that had no engineered features. Our RF model weighted all-but accuracy has improved from 88.2% up to 91.7%. We can also see that the EMO method did very poorly on the DAIO trades, with the neural network yielding a nearly equal result. We know from the decision tree model that the `buysell_EMO` feature is very important; it would appear that the neural network also learned this, but relies on this feature more heavily than RF since it mirrored the poor

performance of the EMO method on DAIO trades. Based on these findings, we will continue with model development geared specifically towards RF since it shows the most promise for improving accuracy in trade signing.

5.4 Stage 2: Final Model Development

We now expand our training and validation sets to more closely parallel our final testing setup. Final testing occurs on the first and last two days of data (March 5, 6, 28, and 29) after training on the middle seventeen days (March 5, 6, ..., 26, 27). To imitate this we will use March 8 and 22 data as our new validation data and March 9, 13, 15, and 20 as our new training data during this final stage of model development. Efforts made to improve validation accuracy include additional feature engineering, feature selection, addressing class imbalance via sampling, and final hyperparameter tuning.

5.4.1 Additional Feature Engineering

Before creating more features it is important to develop a technical understanding of what engineered features RF can and cannot learn on their own. If an important feature cannot be learned from original inputs but is valuable for prediction, it is crucial that we understand what these features might look like so that we can manually create them. In a 2016 IEEE conference paper, Heaton demonstrated the types of features which neural networks, SVM, RF, and gradient-boosted trees can and cannot learn from original inputs. Of particular interest for our model, RF do a poor job of learning new ratio variables of the form $\left[x_{ratio} = \frac{x_1}{x_2}\right]$, rational difference variables of the form $\left[x_{ratDiff} = \frac{x_1 - x_2}{x_3 - x_4}\right]$, and count variables of the form $\left[x_{count} = \sum_{i=1}^{n} 1 \text{ if } x_i > t, \text{ else } 0\right]$ [5]. This bolsters the importance of creating the scaledDiff variables described in Section 5.3.2, as they essentially follow the rational difference form Heaton describes. However, we can draw from these suggestions, as well as our knowledge of the problem of trade signing, to create a few new features which may be valuable.

A new feature we create, `priceRelQuotes`, is of the four-variable rational difference form which Heaton suggests. The work of both EMO [2] and CLNV [3] found that the proximity of trade price to quotes contains valuable predictive information. This was so much the case that CLNV created the `bid30` and `ofr30` features to better characterize this relationship. Drawing from this insight, we create the new feature `priceRelQuotes` $= \left[\frac{P_k - B_k}{A_k - B_k}\right]$. See Figure 5.10 below for a numerical illustration of its implementation. Regardless of the

Quotes	Trade Price ($\mathbf{P_k}$)	priceRelQuotes
	78.81	1.2
$\mathbf{A_k}$	**78.80**	**1.0**
	78.79	0.8
	78.78	0.6
$\mathbf{M_k}$	**78.775**	**0.5**
	78.77	0.4
	78.76	0.2
$\mathbf{B_k}$	**78.75**	**0.0**
	78.74	-0.2
	78.73	-0.4

Fig. 5.10: Numerical illustration of the `priceRelQuotes` feature. This feature quantifies the position of a trade's price relative to quotes in a single number. Trades with a `priceRelQuotes` value of 1.0 occur at the `NBO`, trades with a `priceRelQuotes` value of 0.0 occur at the `NBB`, and all other trades are similarly quantified based on the possition of the trade's price relative to the quotes. Values of `priceRelQuotes` are comparable across all trades, regardless of that trade's specific execution price, `NBO` value, or `NBB` value.

quote values A_k and B_k, trades occurring at the NBO will have a `priceRelQuotes` value of 1.0, trades at the midpoint will have a `priceRelQuotes` value of 0.5, and trades at the NBB will have a `priceRelQuotes` value of 0.0. The `bid30` and `ofr30` variables serve to discretize the spread but stop short of directly relating a trade's price to these values. The `priceRelQuotes` feature directly relates a trade's price with corresponding quotes, numerically capturing where in the spread a trade's price falls. We will see further on that this is a very valuable feature in generating predictions.

Another feature added to the model is `ticks`. The `direction2` feature from Holden and Jacobsen's code [32] contains the difference between the current trade price P_k and the first

preceding different price P_d, $\left[\texttt{direction2} = P_k - P_d\right]$. The engineered feature `dif1_PRICE` captures the nominal difference between each successive trade, $P_k - P_{k-1}$. Considered together these two features create the tick test as described in Section 3.2. Rather than thresholding tick test results to be only buys and sells we will define four values for the `ticks` feature, hopefully lending models additional predictive information:

$$\begin{cases} \text{uptick} & \texttt{ticks} = 2 \quad (P_k > P_j) \\ \text{zero-uptick} & \texttt{ticks} = 1 \quad (P_k = P_j) \text{ and } (P_k > P_d) \\ \text{zero-downtick} & \texttt{ticks} = \text{-}1 \quad (P_k = P_j) \text{ and } (P_k < P_d) \\ \text{downtick} & \texttt{ticks} = \text{-}2 \quad (P_k < P_j) \end{cases}$$

The last, and most important, feature which we will engineer is `sumOfBuysellCols`, simply defined as $\texttt{sumOfBuysellCols} = \left[\ \texttt{buysell\_LR} + \texttt{buysell\_EMO} + \texttt{buysell\_CLNV}\ \right]$. The inspiration for this feature is twofold. First, we know from Heaton's work that RF struggle with creation of count variables [5], which this variable undoubtedly is. Second, we can roughly think of this feature as a very simplistic ensemble learner. The LR, EMO, and CLNV methods are the weak learners, and by summing them we hope to obtain more accurate predictions by aggregating their results. We can also think of this variable as communicating to our models the relative confidence of the predictions of older methods: a value of zero is confidently classified as a buy, a value of three is confidently classified as a sell, and values of one and two are tentatively classified as buys and sells respectively.

5.4.2 Feature Selection

A distinct advantage of RF over other machine learning approaches is the easy selection of variables using feature importance. The default variable importance (VIMP) calculation and tree splitting criterion implemented in `scikit-learn` is based on the decrease in Gini impurity. While it has been shown that Gini-based importance may tend to favor high-cardinality numerical features when compared to permutation importance [57], the standard use of impurity-based importance is sufficient for our purposes. We will also see that it yields

sensible results when verified against our understanding of the different features.

We now add the `priceRelQuotes`, `ticks`, and `sumOfBuysellCols` features from Section 5.4.1 to our model. Additionally we have supplemented our original 68 features with several other scaledDiff and nominalDiff variables, bringing our total number of features to 85. When training with the expanded training and validation sets, the accuracy of LR is 88.2%, EMO is 89.6%, and CLNV is 89.3%. Our RF model achieves wab-PCC of 93.6%, improving from 91.7% at the end of our first stage of model development.

Having created so many new features we ought to see which, if any, of them are valuable predictors. Analogous to our treatment of overall validation accuracy using wab-PCC, we assess variable importance for the remainder of this work by creating the ten all-but models and taking the weighted average of the importance of each variable before making the final ranking. A graphical representation of variable importance for our 85-feature model is given in Figure 5.11. The newly-added `sumOfBuysellCols` and `priceRelQuotes` features are the first and third most important variables respectively. Unsurprisingly, the predictions of older methods are important as well, with `buysell_EMO`, `buysell_CLNV`, and `buysell_LR` ranked second, fourth, and eighth. Finally, scaledDiff features which incorporate `PRICE` as one input and a price analogue such as `midpoint`, `NBO`, or `ofr30` dominate this list of top 20 variables. In particular, features which incorporate a price analogue nearer the top of the spread (`NBO` and `ofr30`, for example) show up more often. This may be an indication that trades at the best ask are more difficult to classify than trades at the best offer.

As such, we remove the 45 least important variables, leaving a model with just the top 40. Many of the variables removed are of the boolean colsAgree variety (e.g. colsAgree(`buysell_EMO`, `buysell_CLNV`)), nominal differences within a variable (e.g. nominalDiff(`PRICE`, `PRICE`, 1)), and scaled differences within a variable (e.g. scaledDiff(`NBBqty`, `NBBqty`, 1)). Also, many variables having to do with size and timestamp were unimportant, with the notable exceptions of `microsDifTQ` and `std_SIZE` (seen above in the top 20 features in Figure 5.11).

The results of this 40-feature model are very similar to the model with all 85 features,

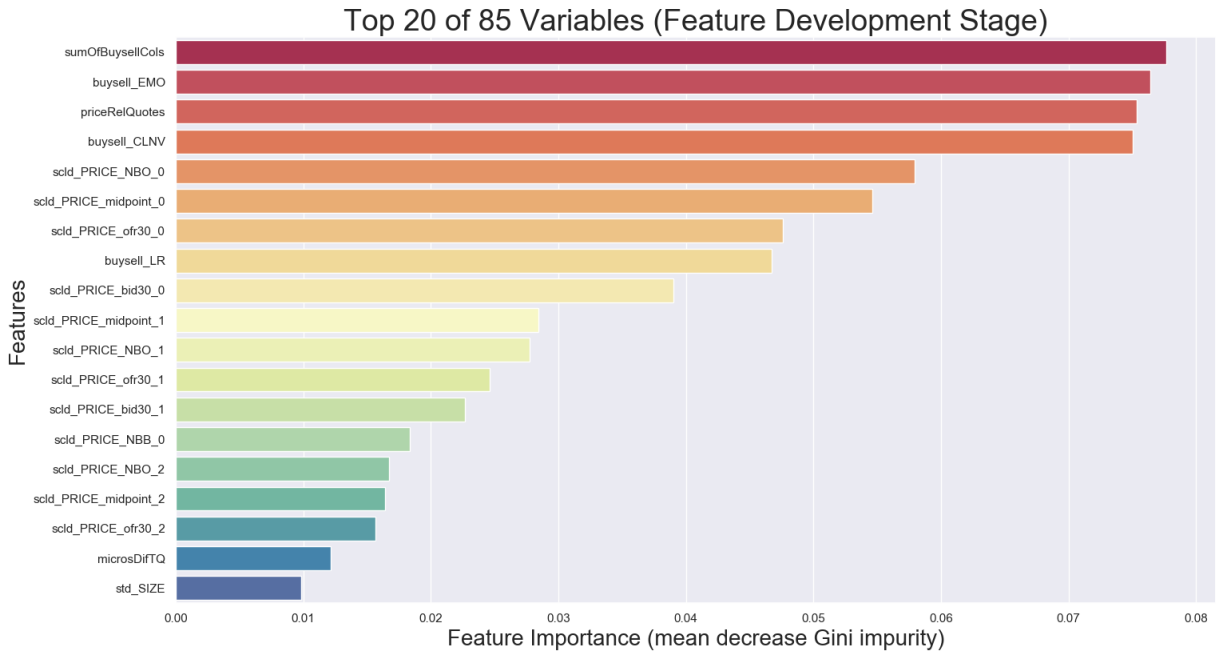

Fig. 5.11: Variable importance of top 20 features for the prototyped RF model containing 85 features. Engineered features `sumOfBuysellCols`, `priceRelQuotes`, and scaled difference features involving `PRICE` and its analogues dominate the list. Features containing the predictions of old methods are also highly important, with EMO predictions coming in as the second most important feature.

edging up from 93.6% up to 93.7% wab-PCC. Since computation is aided by having a smaller set of predictors while accuracy is not compromised, we will use this set of features for the time being. Before declaring feature selection and development complete, we take a look at which trades are being misclassified to see if there are any additional features which we can engineer to improve prediction.

Two highly important features with intuitive meaning are `sumOfBuysellCols` and `priceRelQuotes`. Figures 5.13 and 5.14 illustrate the differences in distribution of these two features for all training trades (in blue) and misclassified test trades (in red). The proportion of sells in the training data is about 45.2% while the proportion of sells in the misclassified test data is about 41.0%. Although somewhat different, this slight disparity is little cause for concern. Figure 5.13 shows that nearly all incorrect predictions result from trades with values of 0 and 2 for `sumOfBuysellCols`. The trades with a `sumOfBuysellCols` value of 0 are "confident buys" according to the three older methods, but often turn out to

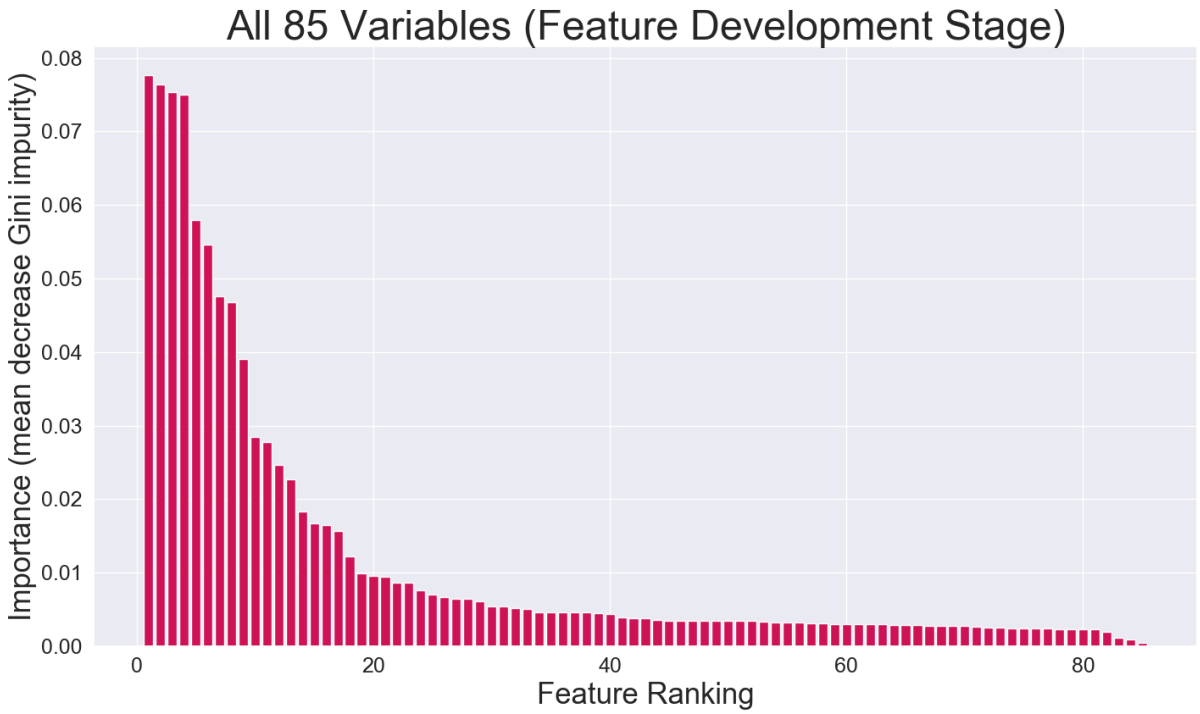

Fig. 5.12: Distribution of variable importance for the prototyped RF model containing 85 features. Notice that feature importance drops off somewhat after the top 20 most important variables and drops again after the 40 most important variables.

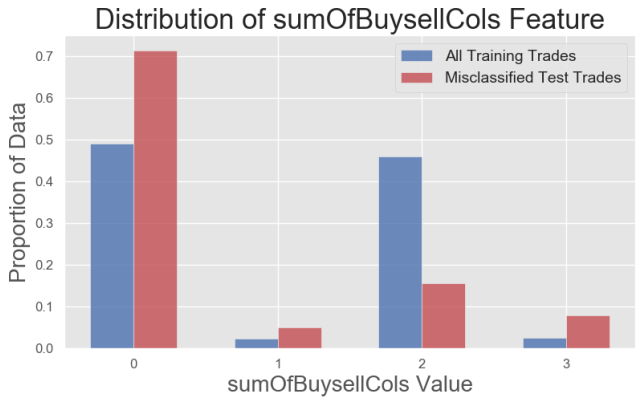

Fig. 5.13: Proportional distribution of `sumOfBuysellCols` for misclassified and training trades. Notice that trades confidently classified as buys (value of 0) by older methods are the most misclassified.

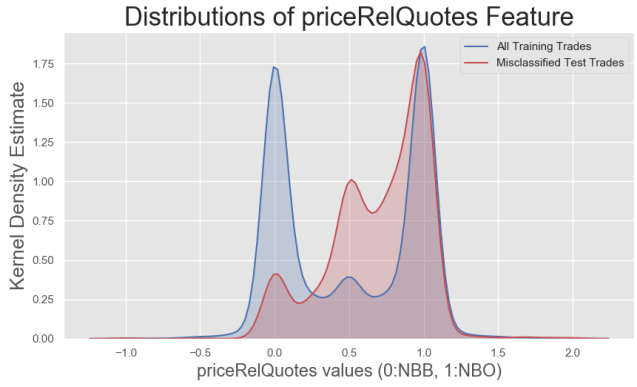

Fig. 5.14: Density esimates of `priceRelQuotes` for misclassified and training trades. Notice that most misclassified trades have a `priceRelQuotes` value near 1.0.

be sells according to our figure. Similarly, trades with a `sumOfBuysellCols` value of 2 are "tentative sells" according to LR, EMO, and CLNV, but may often turn out to be buys.

These ideas are further substantiated by Figure 5.14. Trades near the `NBO` (having a `priceRelQuotes` value around 1.0) comprise the majority of misclassified trades. According to the quote test, LR, EMO, and CLNV methods, we classify trades at the `NBO` as buys. However, this approach often fails when predicting onto validation data as evidenced by the disproportionate concentration of misclassified trades having a `priceRelQuotes` value near 1.0. This is a new finding that is not discussed in any of the works of LR, EMO, and CLNV. Both EMO and CLNV discuss trades near the midpoint being difficult to classify, but neither observed the issue of sells at the `NBO` being difficult to classify. This may be a newly-discovered general phenomena of our model or it may simply be bias from having a small selection of securities and trading days in our data.

Since trades inside of the quotes are more difficult to classify, we extend the logic of the `priceRelQuotes` feature to create the `priceRelNBB` and `priceRelNBO` features. They are calculated as $\texttt{priceRelNBB} = \left[\frac{P_k - B_k}{M_k - B_k}\right]$ and $\texttt{priceRelNBO} = \left[\frac{P_k - A_k}{A_k - M_k}\right]$. These two variables more granularly quantify a price's relation to the NBB (B_k) and NBO (A_k) individually, as opposed to `priceRelQuotes` which does this simultaneously. We also add some scaled differences across indices within `priceRelQuotes`, `priceRelNBB`, and `priceRelNBO`. These new features will hopefully aid our model in signing difficult-to-classify trades which occur at the `NBO`.

5.4.3 Addressing Class Imbalance via Sampling

An imbalance in the proportion of classes is often a concern in binary classification problems, especially when correct prediction of one class is of chief importance. One way of dealing with this is by modifying the cost function being optimized to weight misclassifications of the two classes differently. This will more strongly penalize incorrect predictions for the class we're more concerned with characterizing accurately. Another method, which we will explore here, is the use of sampling techniques to adjust the balance of the classes within training data.

Two commonly used sampling techniques are random oversampling and random undersampling. To even the proportion of the two classes in the training data, oversampling

augments the original data with repeated draws from the minority class observations until parity is realized. This necessitates that some minority class observations be included multiple times in the new training data while others might only appear once. On the other hand, undersampling keeps all minority class observations and randomly selects majority class observations without replacement until equal proportions are achieved, omitting some original majority class observations [37]. While more sophisticated methods for sampling exist, we will first explore these two to determine if there is any promise in pursuing this technique as a means of improving test accuracy.

We found during EDA that some tickers contain a proportion of sells which differs greatly from the overall ratio as well as differing from the ratio of other tickers (see Figure 4.6). To see if treating these tickers differently improves performance we will conduct training data aggregation in two formats. For the first format we aggregate across the data for the nine training tickers and then remedy class imbalance via sampling. For the second format we individually remedy class imbalance within each of the nine tickers and then aggregate them into a single all-but training set. In both cases the validation data is left untouched. As in other experiments, we will continue using all-but validation to gauge results.

In addition to using two methods of sampling and two methods of training data aggregation, we will sample to create two different ratios of buys to sells. There are 42% sells and 58% buys in the original training data ($42/58 \approx 0.72$), so we will sample to create a sells-to-buys ratio of 0.72 to see if obtaining data representative of the broader sample helps. We also sample to create a balanced ratio of 1.0 to see if this improves performance.

Two sampling methods, two ratios of sells to buys, and two methods of training data aggregation results in eight different approaches. The results of these approaches are detailed in Table 5.1. Unfortunately, none of these different sampling schemes yield any consequential improvement on the 93.7% wab-PCC which we have already achieved without use of any sampling. Therefore, we will dispense with all further use of sampling techniques as a means of improving out-of-sample accuracy.

RESULTS OF SAMPLING APPROACHES			
Sampling Method	**Ratio of sells to buys**	**Aggregation Scheme**	**wab-PCC**
Oversampling	0.72	within ticker	93.68
Oversampling	1.00	within ticker	93.75
Oversampling	0.72	across ticker	93.67
Oversampling	1.00	across ticker	93.67
Undersampling	0.72	within ticker	93.70
Undersampling	1.00	within ticker	93.62
Undersampling	0.72	across ticker	93.74
Undersampling	1.00	across ticker	93.66

Table 5.1: All-but validation results of eight different sampling approaches. None of the sampling schemes for rectifying the slight class imbalance in study data resulted in an appreciable improvement in accuracy. Therefore, no sampling method was used for the final model.

5.4.4 Hyperparameter Tuning

Having explored feature engineering, feature selection, and sampling methods as means for increasing wab-PCC, we will tune the hyperparameters of our RF model to see if any final improvements in accuracy can be made. To this point we have simply used 100 trees (`n_estimators` = 100) and all other parameters set to defaults when generating RF models. Relevant defaults include `max_depth` = None (no restrictions on the depth to which trees can grow), `min_samples_split` = 2 (the number of internal samples required to further split a node), and `max_features` = $\sqrt{\text{number of predictors}} = \sqrt{53} \approx 7$ (the number of randomly-chosen features considered for splitting at each node).

Cutler, Cutler, and Stevens explain that there are three parameters which can be tuned to influence RF's accuracy: the number of randomly chosen features considered for splitting at each node (`max_features`), the number of trees grown (`n_estimators`), and the size of each tree [55]. We will specify tree size using the minimum number of observations required to split a node (`min_samples_split`).

We first tune `max_features` and `min_samples_split` simultaneously by using a grid search and selecting optimal values based on wab-PCC. Values of `max_features` = 14 and `min_samples_split` = 19 resulted in the highest wab-PCC of 93.83% accuracy, a

slight improvement on 93.7%. Relative to the default value of seven, the higher value of `max_features` increases computation needed since we have to explore twice as many variables for splitting at each node of each of the 100 trees. On the other hand, the increase of `min_samples_split` from two up to 19 helps to reduce computation since we stop well before reaching fully-grown trees.

Finally, we will tune the number of trees grown for our model. Breiman's original paper on RF showed that as the number of trees increases the generalization error of the model almost surely converges to a limit [47]. As recommended by Cutler et al. [55], we increase the number of trees in our model and plot OOB accuracy versus validation accuracy as seen in Figure 5.15. We observe no overfitting (OOB accuracy continuing to improve while

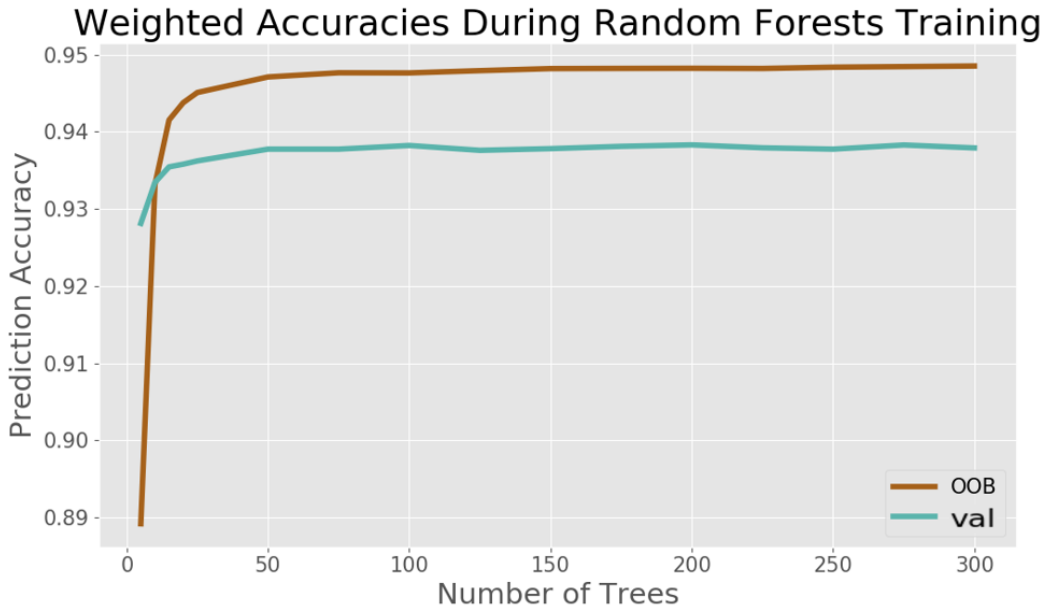

Fig. 5.15: OOB and all-but validation accuracy as a function of the number of trees. Since validation accuracy levels off after 100 trees, we will use `n_estimators` = 100 in our final RF model.

validation accuracy flatlines) and can see that we reach an asymptotic accuracy of 93.8% somewhere around the 50-tree mark. Since the risk of overfitting seems to be negligible we will stick with 100 trees in our final model to be sure that we're obtaining every bit of predictive improvement possible.

5.4.5 Specifications of Final Model

The model which we have developed for final use on test data is a RF classifier with `scikit-learn` argument values of `n_estimators` = 100, `min_samples_split` = 19, and `max_features` = 14. Our training and test data contain 53 features; 46 of these are engineered from original features and seven of them are remaining original features. We have dropped all original features, such as `PRICE` or `SIZE`, whose scales are not shared across [day + ticker] combinations. No sampling is performed on training data to try and ameliorate class imbalance. A complete list of features used, and their respective formulae for calculation, can be found in Tables A.6 and A.7.

CHAPTER 6

FINAL RESULTS

6.1 Design of Performance Assessment

In much the same manner as we assessed models during development in Chapter 5, we will calculate weighted all-but PCC (wab-PCC) to quantify out-of-sample performance. Of all metrics which we might use to evaluate model performance, this one most closely simulates the prediction of trade signs for data from different days and securities than those used in our study. Models were trained on March 5, 2018 through March 27, 2018 data from nine tickers and the collective March 1, 2, 28, and 29 data from the omitted tenth ticker is predicted onto. The fact that training and test data are both temporally and ticker-wise disjoint emphasizes out-of-sample prediction and model generalizability.

Since there are ten tickers, this generates ten different test accuracies. Rather than simply average these ten accuracies we weight each accuracy by the number of trades in the test data for that ticker to ensure they're represented proportionally. One of these ten train-test splits is illustrated in Figure 6.1 where we train on all-but AAOI trades from March 5-27 and then predict onto AAOI trades from March 1, 2, 28, and 29. In total, there are 498,528 trades in the training data and 146,395 trades in the test data.

6.2 Final Model Results

After conducting model training and testing as described above, we obtain the results seen in Figure 6.2 for the random RF, neural network, and QDA models. Although QDA achieved passable results in model prototyping with 78.7% wab-PCC, it performs poorly during final testing coming in at just 69.1%. The neural network model moves in the other direction, improving from 86.6% wab-PCC during prototyping to 91.2% during final testing. Some of this improvement may be due to the fact that the poor performance of

03/05/2018	.....	03/27/2018
AAOI	.....	AAOI
BABY	.....	BABY
CA	.....	CA
DAIO	.....	DAIO
EA	.....	EA
FANG	.....	FANG
GABC	.....	GABC
HA	.....	HA
IAC	.....	IAC
JACK	.....	JACK

03/01/2028	03/02/2018	03/28/2018	03/29/2018
AAOI	AAOI	AAOI	AAOI
BABY	BABY	BABY	BABY
CA	CA	CA	CA
DAIO	DAIO	DAIO	DAIO
EA	EA	EA	EA
FANG	FANG	FANG	FANG
GABC	GABC	GABC	GABC
HA	HA	HA	HA
IAC	IAC	IAC	IAC
JACK	JACK	JACK	JACK

Fig. 6.1: Depiction of all-but testing using March 5-27 data for training and March 1, 2, 28, and 29 data for testing. This scheme emphasizes out-of-sample prediction to measure final model performance, furthering the argument of the model generalizing well onto unseen data.

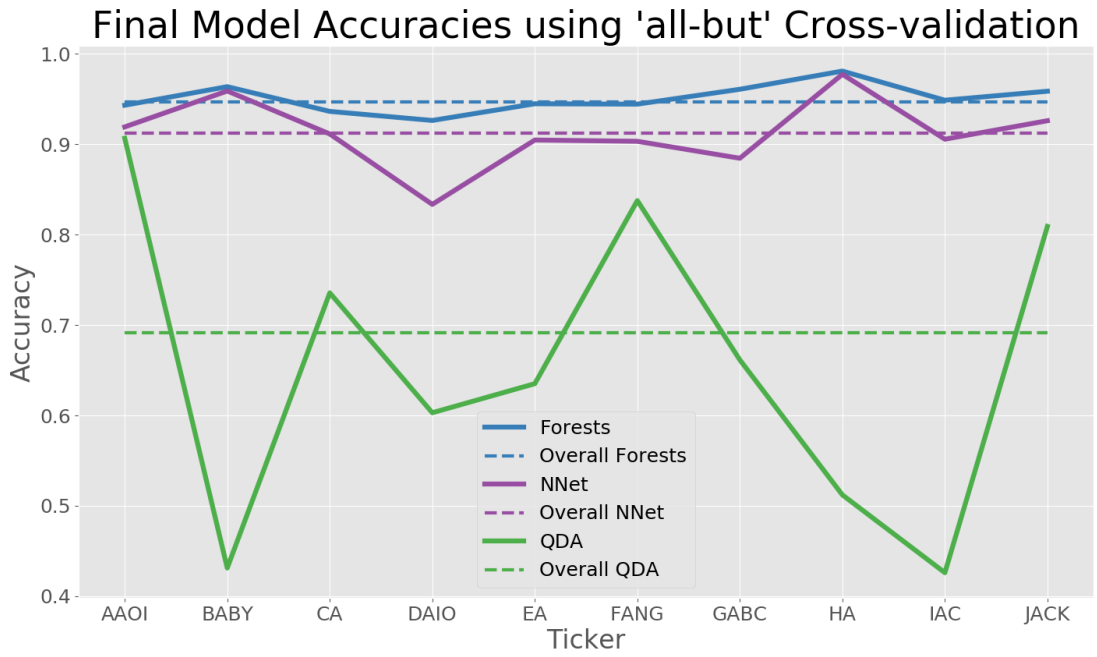

Fig. 6.2: All-but test accuracies of the final RF, neural network, and QDA models. QDA performs poorly, achieving roughly 70% accuracy. RF and the neural network do quite well; the RF model performs better overall, as well as for each individual out-of-sample ticker.

the EMO method on DAIO trades in our small prototyping sample bled over into the poor performance of the prototyped neural network. This bias has been somewhat mitigated by training on many more samples. The final network model still struggles on DAIO trades at just 83.4% accuracy but substantially improves on the 72.6% accuracy of the prototyped network. The RF model also realizes an increase over its prototyped counterpart. The RF

model in stage two of development had wab-PCC of 93.7% while the results depicted here represent 94.7% wab-PCC on final test data.

The single most salient result of this work is pictured below in Figure 6.3. The final RF model clearly and substantially outperforms the predictions of the LR, EMO, and CLNV methods. RF's wab-PCC of 94.7% handily beats accuracies of 89.3%, 90.2%, and 90.1%

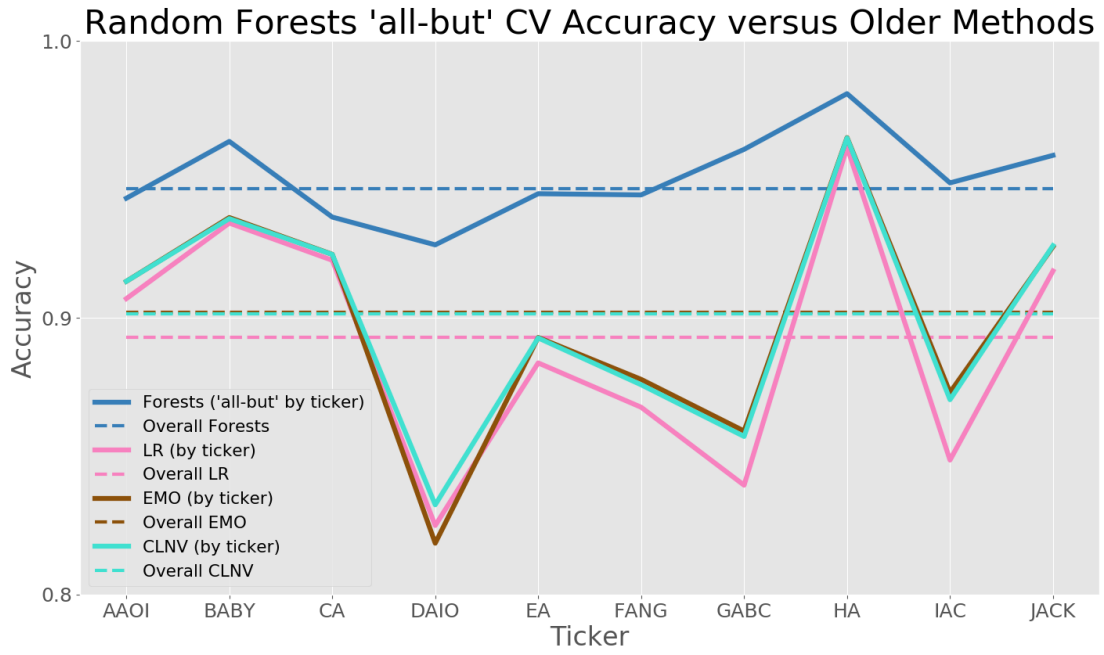

Fig. 6.3: All-but test accuracies of the final RF model, LR, EMO, and CLNV methods. The final RF model substantially outperforms the older methods overall (dotted lines), as well as for each individual ticker (solid lines). The RF model improves upon the predictions of older methods for the DAIO and GABC tickers by more than 10 percentage points.

from LR, EMO, and CLNV respectively, outpacing all of them by at least 4.5 percentage points and delivering state-of-the-art results. Not only is the overall performance of RF much better than old methods, but the prediction for individual tickers is also better for all ten securities. In particular, RF's out-of-sample prediction onto DAIO trades at 92.6% is 10.1 percentage points better than the 82.5% accuracy of the old methods. Predictions of the LR method for GABC trades are only 84.0% accurate while RF achieves 96.1% out-of-sample accuracy for a sizable improvement of 12.1 percentage points.

These accuracies are substantially higher than those reported in the works of LR, EMO, CLNV, and Rosenthal. An examination of why this is the case is not undertaken here, but a couple of thoughts on why this may be happening are given. One obvious advance is the removal of human clerical error and delay thanks to the use of computerized order books and trading systems. Lee and Ready point to this as a concern in their 1991 publication [1], but it is no longer an issue when using 2018 data. Another potential reason for these higher accuracies is the enhanced quality and granularity of data. Timestamps of trades and quotes have become more precise over time, with our data having microsecond timestamps. This allows for more accurate matching of trades with corresponding quotes. In turn, decision rule methods which rely heavily on the best bid and best ask to determine trades' signs produce more accurate predictions as seen in our study.

6.3 Insights into the Final RF Model

Although RF offer less opportunity for interpretation than some methods, we can still deduce valuable insights from our final model. The feature importance plot in Figure 6.4 is fairly similar to that of our prototyped model. One noticeable difference is how much more important the `sumOfBuysellCols` and `buysell_EMO` features are here in the final model than they were in the prototyped model (Figure 5.11). We also see that `priceRelNBO` and `priceRelNBB` were valuable additions to the set of features used. As we also observed in the prototyped model, features which relate `PRICE` to `NBO` comprise much of the 20 most important features, suggesting trades which occur at the `NBO` may be harder to classify.

Another means we can employ to understand our RF model is the use of partial dependence plots [48]. These plots characterize the marginal effect which a given variable has on the predicted response. The partial dependence plot for the `priceRelQuotes` feature is shown in Figure 6.5. The quote test classifies trades with a price greater than the midpoint as a buy and trades with a price less than the midpoint as a sell. The plot for `priceRelQuotes` is suggesting similar behavior: trades with a `priceRelQuotes` value of 0.5 or less (which are trades whose prices are less than the `midpoint`) have a predicted probability of about 0.58, increasing up to about 0.62 at a value of 0.0 (trades near the `NBB`). In

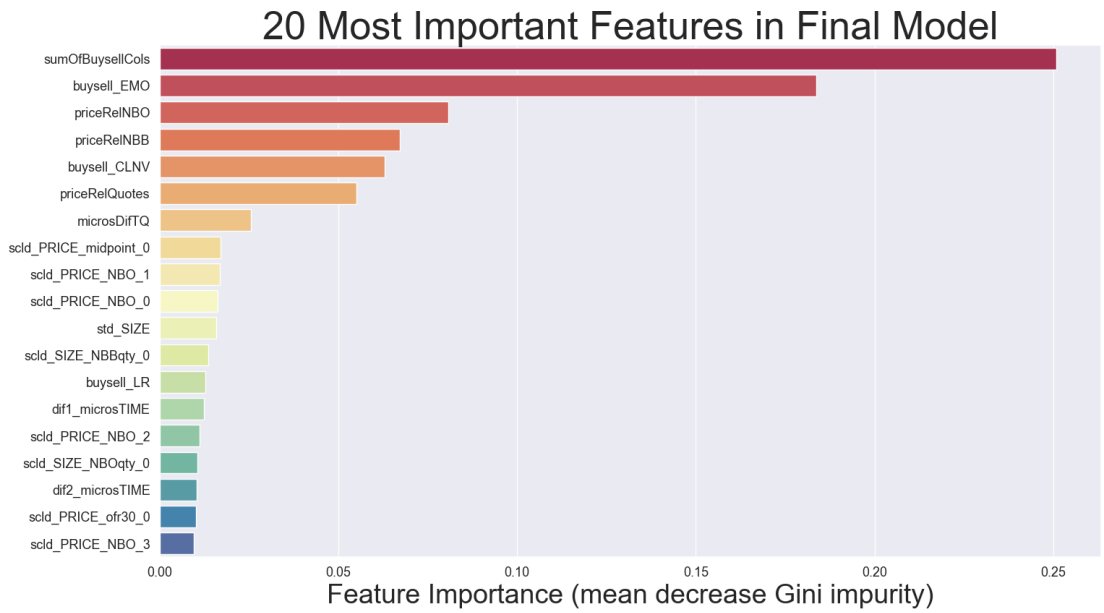

Fig. 6.4: Variable importance of the 20 most importance features in the final RF model. Aside from `buysell_EMO` and `buysell_CLNV`, the ten most important features are all engineered. The `sumOfBuysellCols` feature is by far the most important, with `priceRelNBO`, `priceRelNBB`, and `priceRelQuotes` rounding out the top five.

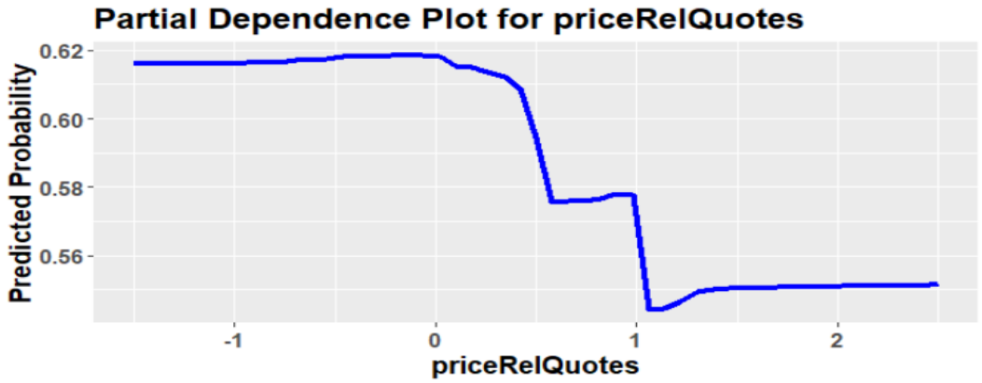

Fig. 6.5: Partial dependence plot of the `priceRelQuotes` features from the final RF model. The marginal effect of the `priceRelQuotes` feature on the predicted response is closely aligned with the intuition of the quote test. Trades with a `priceRelQuotes` value of 0.0 are predicted to be sells and trades with a priceRelQuotes value of 1.0 are predicted to be buys. The shelf in the middle of the graphic is at 0.58 (rather than 0.5) since this is the proportion of buys in the data. A predicted probability above (below) 0.58 corresponds to a sell (buy) prediction.

other words, these trades are more likely to be classified as sells. This cleanly squares with the logic of the quote test, which would also classify these trades as sells. Still considering

the effect of `priceRelQuotes` on prediction, trades above the `midpoint` are more likely to be buys, with trades above the `NBO` (`priceRelQuotes` value of at least 1.0) most likely to be classified as buys.

We should note that assuming a cutoff of 0.5 for predicted probability when classifying trades is not used for this plot. The clear shelf near 0.58 for predicted probability aligns nearly perfectly with the proportion of buys in the training data. The plot makes much more sense when using 0.58 as the predicted probability threshold, suggesting that trades at or below the `NBB` are influenced by `priceRelQuotes` to be sells, trades at or above the `NBO` are influenced by `priceRelQuotes` to be buys, and trade inside of the quotes are more difficult to classify with a clear difficulty in classifying trades between the `midpoint` and the `NBO` as indicated by the shelf residing between values of 0.5 and 1.0.

Examining the distribution of misclassified test trades in Figures 6.6 and 6.7 sheds additional light on the difficulty of classifying trades between the `midpoint` and `NBO`. Figure

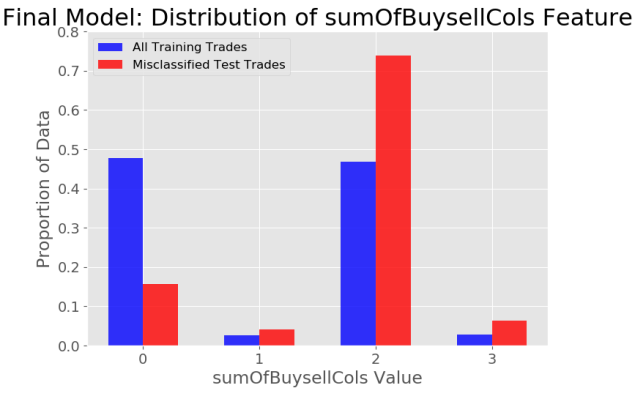

Fig. 6.6: Proportional distributions of `sumOfBuysellCols` for training trades and misclassified test trades. Most misclassified trades are tentatively predicted to be sells by this feature (have a `sumOfBuysellCols` value of 2).

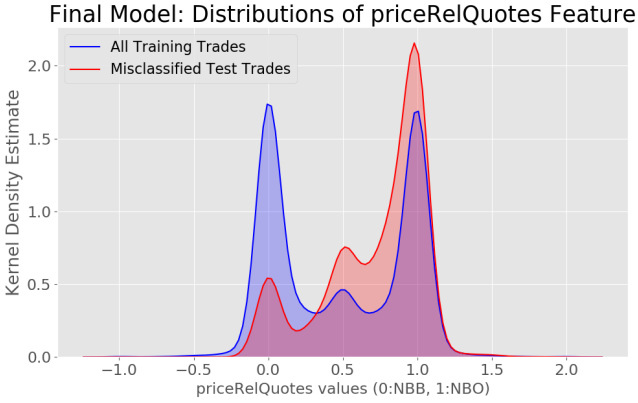

Fig. 6.7: Density esimates of `priceRelQuotes` for training trades and misclassified test trades. Most misclassified trades occur at the `NBO` (a `priceRelQuotes` value near 1.0). We would expect these trades to be buys, but this graphic suggests that some of them are sells.

6.7 shows that most misclassified test trades have a price near the `NBO`. This is substantiated by Figure 6.6. The majority of misclassified trades have a `sumOfBuysellCols` value of two,

meaning that they are tentatively classified as a sell by LR, EMO, and CLNV, and end up being predicted incorrectly as sells by the RF model. In the context of splits made within trees, it may be that `sumOfBuysellCols` is classifying difficult trades at the `NBO` while the also-important `buysell_EMO` is classifying the more well-behaved trades. Even with the discriminatory aid of `priceRelQuotes` and `sumOfBuysellCols`, some trades near the `NBO` still prove difficult to classify.

A potential cause of this might be the effect of the "uptick rule." The most recent iteration of this SEC regulation stipulates that if a security's price dropped by at least 10% during the previous day, short selling can only occur on an uptick or at a price above that of the most recent execution [58]. Sells typically occur at the `NBB` (a `priceRelQuotes` value of 0.0), but the uptick rule forces some sells to occur at higher prices than usual. This causes some trades which occur at the `NBO` (typically buys) to be sells and likely results in their misclassification due to the importance of a trade's position in the spread for determining direction. However, the uptick rule was only triggered once during our sample period; on March 28, 2018, the uptick rule was triggered at 12:44 PM for the DAIO ticker. Only 0.6% of misclassified test trades are from the DAIO ticker, while DAIO trades make up 0.8% of all test trades. This suggests that DAIO trades are not disproportionately likely to be misclassified. Based on this evidence, we suspect that something other than the uptick rule is biasing misclassification toward trades occurring near the `NBO`.

CHAPTER 7

FUTURE WORK AND CONCLUSION

The application of machine learning methods as a novel approach to the classification of trade direction has yielded highly promising results in this study. To generate predictions for the LR, EMO, and CLNV methods of trade signing, NBBO data was used to augment trade data with best offer and best bid values for each executed trade. However, a large portion of the quotes in NBBO files were never used to aid prediction since many of them were not matched with a trade. Movements in the NBB and NBO prior to a trade's execution would likely contain very valuable information for determining direction. This information could potentially be leveraged through a two-tiered neural network. The first tier would take in a variable-length vector containing unused, chronological pairs of best bid and best ask values preceding a trade. This vector would be passed through the first network, outputting a fixed-length context vector. This context vector and the original input vector used in our models could then be fed into the second network to create a final prediction.

While a single neural network did improve upon the results of older methods, a random forests model performed best and achieved state-of-the-art results in correctly predicting trade signs for March 2018 NASDAQ data. Engineering of novel, tailor-made features from existing data substantially improved performance of our models. Our final model outperformed the LR, EMO, and CLNV methods of signing trades when predicting onto out-of-sample test data, surpassing their predictions by 4.5 percentage points overall and improving accuracy by as much as 12.1 percentage points for individual securities. Implementation of this model by researchers will remove the need to process order-level data, promote more reliable results in studies relying on accurate trade signs, and can save academic departments thousands of dollars in data subscription fees.

REFERENCES

[1] C. M. Lee and M. J. Ready, "Inferring trade direction from intraday data," *The Journal of Finance*, vol. 46, no. 2, pp. 733–746, 1991.

[2] K. Ellis, R. Michaely, and M. O'Hara, "The accuracy of trade classification rules: Evidence from NASDAQ," *Journal of Financial and Quantitative Analysis*, vol. 35, no. 4, pp. 529–551, 2000.

[3] B. Chakrabarty, B. Li, V. Nguyen, and R. A. Van Ness, "Trade classification algorithms for electronic communications network trades," *Journal of Banking & Finance*, vol. 31, no. 12, pp. 3806–3821, 2007.

[4] D. W. Rosenthal, "Modeling trade direction," *Journal of Financial Econometrics*, vol. 10, no. 2, pp. 390–415, 2012.

[5] J. Heaton, "An empirical analysis of feature engineering for predictive modeling," in *SoutheastCon 2016*. IEEE, 2016, pp. 1–6.

[6] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. Springer, 2013, vol. 112.

[7] J. Hasbrouck, "Trades, quotes, inventories, and information," *Journal of Financial Economics*, vol. 22, no. 2, pp. 229–252, 1988.

[8] L. R. Glosten and L. E. Harris, "Estimating the components of the bid/ask spread," *Journal of Financial Economics*, vol. 21, no. 1, pp. 123–142, 1988.

[9] R. W. Holthausen, R. W. Leftwich, and D. Mayers, "The effect of large block transactions on security prices: A cross-sectional analysis," *Journal of Financial Economics*, vol. 19, no. 2, pp. 237–267, 1987.

[10] M. E. Blume, A. C. MacKinlay, and B. Terker, "Order imbalances and stock price movements on October 19 and 20, 1987," *The Journal of Finance*, vol. 44, no. 4, pp. 827–848, 1989.

[11] L. Harris, "A day-end transaction price anomaly," *Journal of Financial and Quantitative Analysis*, vol. 24, no. 1, pp. 29–45, 1989.

[12] C. M. C. Lee, "Information dissemination and the small trader: an intraday analysis of the small trader response to announcements of corporate earnings and changes in dividend policy." Ph.D. dissertation, Cornell University, Ithaca, NY, 1991.

[13] C. M. Lee, "Earnings news and small traders: An intraday analysis," *Journal of Accounting and Economics*, vol. 15, no. 2-3, pp. 265–302, 1992.

[14] G. Bernile, J. Hu, and Y. Tang, "Can information be locked up? Informed trading ahead of macro-news announcements," *Journal of Financial Economics*, vol. 121, no. 3, pp. 496–520, 2016.

[15] C. W. Holden and S. Jacobsen, "Liquidity measurement problems in fast, competitive markets: Expensive and cheap solutions," *The Journal of Finance*, vol. 69, no. 4, pp. 1747–1785, 2014.

[16] N. Hautsch, *Econometrics of Financial High-Frequency Data*. Springer Science & Business Media, 2011.

[17] A. Madhavan, M. Richardson, and M. Roomans, "Why do security prices change? A transaction-level analysis of NYSE stocks," *The Review of Financial Studies*, vol. 10, no. 4, pp. 1035–1064, 1997.

[18] J. Hasbrouck, *Empirical Market Microstructure: The Institutions, Economics, and Econometrics of Securities Trading*. Oxford University Press, 2007.

[19] WRDS. About wharton research data services (WRDS). [Online]. Available: https://wrds-web.wharton.upenn.edu/wrds/about/index.cfm

[20] M. K. Pratt. (2005, Aug. 15) Sidebar: Some ideas just aren't worth it. ComputerWorld. [Online]. Available: https://www.computerworld.com/article/2558143/sidebar--some-ideas-just-aren-t-worth-it.html

[21] NASDAQ. Price list - U.S. equities. [Online]. Available: http://www.nasdaqtrader.com/Trader.aspx?id=DPUSdata

[22] Lobster. Access options. [Online]. Available: https://lobsterdata.com/info/AccessOptions.php

[23] Wall Street Prep. Bloomberg vs. Capital IQ vs. FactSet vs. Thomson Reuters Eikon. [Online]. Available: https://www.wallstreetprep.com/knowledge/bloomberg-vs-capital-iq-vs-factset-vs-thomson-reuters-eikon/

[24] W. Kenton. (2019, Mar. 29) Broker-dealer. Investopedia. [Online]. Available: https://www.investopedia.com/terms/b/broker-dealer.asp

[25] J. Hasbrouck, "Securities trading: Principles and procedures," *Manuscript, version*, vol. 12, 2017.

[26] S. Shobhit. (2019, Jun. 25) The world of high-frequency algorithmic trading. Investopedia. [Online]. Available: https://www.investopedia.com/articles/investing/091615/world-high-frequency-algorithmic-trading.asp

[27] E. Terrell, "History of the American and NASDAQ stock exchanges," in *Library of Congress–Business Reference Services*, 2010.

[28] W. F. of Exchanges, *Annual Report and Statistics*. The Federation, 2014.

[29] NASDAQ. (2018) NASDAQ TotalView-ITCH 5.0. [Online]. Available: https://www.nasdaqtrader.com/content/technicalsupport/specifications/dataproducts/NQTVITCHspecification.pdf

[30] E. R. Odders-White, "On the occurrence and consequences of inaccurate trade classification," *Journal of Financial Markets*, vol. 3, no. 3, pp. 259–286, 2000.

[31] WRDS. Overview of TAQ data. [Online]. Available: https://wrds-www.wharton.upenn.edu/pages/support/data-overview/wrds-overview-taq

[32] Holden, Craig W and Jacobsen, Stacey, "HOLDEN AND JACOBSEN DAILY TAQ CODE 2018-03-16," http://www.excelmodeling.com/Holden-and-Jacobsen-Daily-TAQ-and-Monthly-TAQ-Codes-2018-03-16.zip, 2018.

[33] G. Grolemund and H. Wickham, "R for data science," Feb 2018. [Online]. Available: https://vidia.ccr.buffalo.edu/resources/698

[34] S. Alizadeh, M. W. Brandt, and F. X. Diebold, "Range-based estimation of stochastic volatility models," *The Journal of Finance*, vol. 57, no. 3, pp. 1047–1091, 2002.

[35] Financial Engines. (2018, Jan) Market capitalization: Large-cap, mid-cap, and small-cap stocks. Financial Engines. [Online]. Available: https://financialengines.com/education-center/small-large-mid-caps-market-capitalization/

[36] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[37] A. Estabrooks, T. Jo, and N. Japkowicz, "A multiple resampling method for learning from imbalanced data sets," *Computational Intelligence*, vol. 20, no. 1, pp. 18–36, 2004.

[38] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning," in *International Conference on Intelligent Computing*. Springer, 2005, pp. 878–887.

[39] B. W. Yap, K. A. Rani, H. A. A. Rahman, S. Fong, Z. Khairudin, and N. N. Abdullah, "An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets," in *Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013)*. Springer, 2014, pp. 13–22.

[40] C. J. ter Braak, "Principal components biplots and alpha and beta diversity," *Ecology*, vol. 64, no. 3, pp. 454–462, 1983.

[41] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[42] K. R. Moon, D. van Dijk, Z. Wang, S. Gigante, D. B. Burkhardt, W. S. Chen, K. Yim, A. van den Elzen, M. J. Hirn, R. R. Coifman *et al.*, "Visualizing structure and transitions in high-dimensional biological data," *Nature Biotechnology*, vol. 37, no. 12, pp. 1482–1492, 2019.

[43] J. Berkson, "Application of the logistic function to bio-assay," *Journal of the American Statistical Association*, vol. 39, no. 227, pp. 357–365, 1944.

[44] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning*. Springer Series in Statistics New York, 2001, vol. 1, no. 10.

[45] E. Fix and J. Hodges, "Discriminatory analysis, nonparametric discrimination: consistency properties," *Technical Report 4, USAF School of Aviation Medicine*, 1951.

[46] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Wadsworth & Brooks/Cole Advanced Books & Software, 1984.

[47] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[48] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of Statistics*, pp. 1189–1232, 2001.

[49] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[50] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics and Intelligent Laboratory Systems*, vol. 39, no. 1, pp. 43–62, 1997.

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python ," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[52] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," *NIPS 2017 Autodiff Workshop*, 2017.

[53] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[54] A. Abdiansah and R. Wardoyo, "Time complexity analysis of support vector machines (svm) in libsvm," *International Journal Computer and Application*, vol. 128, no. 3, pp. 28–34, 2015.

[55] A. Cutler, D. R. Cutler, and J. R. Stevens, "Random forests," in *Ensemble Machine Learning*. Springer, 2012, pp. 157–175.

[56] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[57] C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn, "Bias in random forest variable importance measures: Illustrations, sources and a solution," *BMC Bioinformatics*, vol. 8, no. 1, p. 25, 2007.

[58] U.S. Securities and Exchange Commission, "Amendments to exchange act rule 10a-1 and rules 201 and 200 (g) of regulation SHO," https://www.sec.gov/divisions/marketreg/tmcompliance/rules10a-200g-201-secg.htm, 2008.

APPENDICES

APPENDIX A

Additional Tables

CLEANED STUDY DATA FEATURES		
Feature	**Data type**	**Description**
DATE	String	Day of the trade's execution, MM/DD/YYYY.
SYM_ROOT	String	Four character ticker symbol for the stock.
TIME_M	Integer	Microseconds since midnight when trade executes.
QTime	Integer	Microseconds since midnight for the prevailing quote.
EX	Char	Exchange. All values are Q, indicating NASDAQ.
SIZE	Integer	Number of shares exchanged in the trade.
PRICE	Float	Trade's per-share price in dollars.
TR_SEQNUM	Integer	Unique, TAQ-specified value. Monotonically increasing.
NBO	Float	Prevailing best offer (ask) in dollars.
NBB	Float	Prevailing best bid in dollars.
NBOqty	Integer	Number of shares of the NBO limit order.
NBBqty	Integer	Number of shares of the NBB limit order.
midpoint	Float	Average of the NBO and NBB in dollars.
direction2	Float	$P_k - P_d$ in dollars.
ofr30	Float	$0.7A_k + 0.3B_k$ in dollars.
bid30	Float	$0.3A_k + 0.7B_k$ in dollars.
buysell_S	Integer	True responses (from ITCH): 1 = sell, 0 = buy.
buysell_LR	Integer	Predicted direction via LR method.
buysell_EMO	Integer	Predicted direction via EMO method.
buysell_CLNV	Integer	Predicted direction via CLNV method.

Table A.1: A complete listing of features included in final cleaned study data prior to any feature engineering or standardization.

Distribution of Trade Prices Relative to Quotes			
Position of Price Relative to Quotes		**Training Data**	**Test Data**
$P_k > A_k$	Above the ask	1.88%	2.13%
$P_k = A_k$	At the ask	32.84%	35.45%
$(P_k < A_k)$ & $(P_k > M_k)$	Between ask and midpoint	12.18%	10.46%
$P_k = M_k$	At the midpoint	6.30%	4.27%
$(P_k < M_k)$ & $(P_k > B_k)$	Between midpoint and bid	12.15%	10.52%
$P_k = B_k$	At the bid	32.73%	35.11%
$P_k < B_k$	Below the bid	1.92%	2.06%
	Total	**100.0%**	**100.0%**

Table A.2: Proportion of trades relatives to quotes for training and test data. We observe that the training and test data share a very similar distribution, and most trades occur at the quotes. The percentages in this table correspond to the visual representation of the data in Figure 4.5.

CORRELATIONS OF `microsTIME` ANALOGUES			
	`TR_SEQNUM`	`microsTIME`	`microsQTime`
`TR_SEQNUM`	1.0	0.995376	0.995374
`microsTIME`	0.995376	1.0	0.999999
`microsQTime`	0.995374	0.999999	1.0

Table A.3: Correlation matrix for `microsTIME` analogues. All pairwise correlations for this group of variables are at least 0.995. See Figure 4.7 for a visual representation.

CORRELATIONS OF buysell_S ANALOGUES					
	direction2	buysell_S	buysell_LR	buysell_EMO	buysell_CLNV
direction2	1.0	0.391199	0.460282	0.517889	0.495802
buysell_S	0.391199	1.0	0.817980	0.827657	0.831066
buysell_LR	0.460282	0.817980	1.0	0.912820	0.950601
buysell_EMO	0.517889	0.827657	0.912820	1.0	0.962181
buysell_CLNV	0.495802	0.831066	0.950601	0.962181	1.0

Table A.4: Correlation matrix for buysell_S analogues. Pairwise correlations for the four variables with prefix buysell are quite high. The fact that these correlations are high indicate valuable predictive information, while also indicating that there is room for improvement in predictive accuracy. See Figure 4.7 for a visual representation.

CORRELATIONS OF STANDARDIZED PRICE ANALOGUES						
	std_bid30	std_NBO	std_midpoint	std_PRICE	std_NBB	std_ofr30
std_bid30	1.0	0.998648	0.999883	0.996886	0.999727	0.999541
std_NBO	0.998648	1.0	0.999325	0.996211	0.997173	0.999763
std_midpoint	0.999883	0.999325	1.0	0.996960	0.999253	0.999888
std_PRICE	0.996886	0.996211	0.996960	1.0	0.996339	0.996813
std_NBB	0.999727	0.997173	0.999253	0.996339	1.0	0.998564
std_ofr30	0.999541	0.999763	0.999888	0.996813	0.998564	1.0

Table A.5: Correlation matrix for standardized PRICE analogues. All pairwise correlations for this group of variables are at least 0.996. See Figure 4.7 for a visual representation.

FIRST 27 OF FINAL 53 FEATURES INCLUDED IN FORESTS MODEL			
Rank	**VIMP**	**Feature Name**	**Calculation**
1	0.250855	$\texttt{sumOfBuysellCols}_i$	$\left(\texttt{buysell\_LR}_i + \texttt{buysell\_EMO}_i + \texttt{buysell\_CLNV}_i\right)$
2	0.183704	$\texttt{buysell\_EMO}_i$	$\texttt{buysell\_EMO}_i$ (original)
3	0.080638	$\texttt{priceRelNBO}_i$	$\left(\frac{\texttt{NBO}_i - \texttt{PRICE}_i}{\texttt{NBO}_i - \texttt{midpoint}_i}\right)$
4	0.067092	priceRelNBB	$\left(\frac{\texttt{PRICE}_i - \texttt{NBB}_i}{\texttt{midpoint}_i - \texttt{NBB}_i}\right)$
5	0.062819	$\texttt{buysell\_CLNV}_i$	$\texttt{buysell\_CLNV}_i$ (original)
6	0.054997	$\texttt{priceRelQuotes}_i$	$\left(\frac{\texttt{PRICE}_i - \texttt{NBB}_i}{\texttt{NBO}_i - \texttt{NBB}_i}\right)$
7	0.025455	$\texttt{microsDifTQ}_i$	$\texttt{microsDifTQ}_i$ (original)
8	0.017037	scld_PRICE_midpoint_0	$\left(\frac{\texttt{PRICE}_i - \texttt{midpoint}_i}{\texttt{midpoint}_i}\right)$
9	0.016866	scld_PRICE_NBO_1	$\left(\frac{\texttt{PRICE}_i - \texttt{NBO}_{i-1}}{\texttt{NBO}_{i-1}}\right)$
10	0.016163	scld_PRICE_NBO_1	$\left(\frac{\texttt{PRICE}_i - \texttt{NBO}_i}{\texttt{NBO}_i}\right)$
11	0.015682	std_SIZE	$\left(\frac{\texttt{std\_SIZE}_i - median_{\texttt{std\_SIZE}}}{IQR_{\texttt{std\_SIZE}}}\right)$
12	0.013538	scld_SIZE_NBBqty_0	$\left(\frac{\texttt{SIZE}_i - \texttt{NBBqty}_i}{\texttt{NBBqty}_i}\right)$
13	0.012690	$\texttt{buysell\_LR}_i$	$\texttt{buysell\_LR}_i$ (original)
14	0.012321	$\texttt{dif1\_microsTIME}_i$	$\texttt{microsTIME}_i - \texttt{microsTIME}_{i-1}$
15	0.011083	scld_PRICE_NBO_2	$\left(\frac{\texttt{PRICE}_i - \texttt{NBO}_{i-2}}{\texttt{NBO}_{i-2}}\right)$
16	0.010448	scld_SIZE_NBOqty_0	$\left(\frac{\texttt{SIZE}_i - \texttt{NBOqty}_i}{\texttt{NBOqty}_i}\right)$
17	0.010235	dif2_microsTIME	$\texttt{microsTIME}_i - \texttt{microsTIME}_{i-2}$
18	0.010107	scld_PRICE_ofr30_0	$\left(\frac{\texttt{PRICE}_i - \texttt{ofr30}_i}{\texttt{ofr30}_i}\right)$
19	0.009521	scld_PRICE_NBO_3	$\left(\frac{\texttt{PRICE}_i - \texttt{NBO}_{i-3}}{\texttt{NBO}_{i-3}}\right)$
20	0.006292	dif1_microsQTime	$\texttt{microsQTime}_i - \texttt{microsQTime}_{i-1}$
21	0.006198	std_TR_SEQNUM	$\left(\frac{\texttt{std\_TR\_SEQNUM}_i - median_{\texttt{std\_TR\_SEQNUM}}}{IQR_{\texttt{std\_TR\_SEQNUM}}}\right)$
22	0.005646	scld_PRICE_NBO_4	$\left(\frac{\texttt{PRICE}_i - \texttt{NBO}_{i-4}}{\texttt{NBO}_{i-4}}\right)$
23	0.005496	$\texttt{microsQTime}_i$	$\texttt{microsQTime}_i$ (original)
24	0.005442	$\texttt{microsTIME}_i$	$\texttt{microsTIME}_i$ (original)
25	0.005205	$\texttt{direction2}_i$	$\texttt{direction2}_i$ (original)
26	0.004570	scld_PRICE_bid30_1	$\left(\frac{\texttt{PRICE}_i - \texttt{bid30}_{i-1}}{\texttt{bid30}_{i-1}}\right)$
27	0.004422	scld_PRICE_midpoint_1	$\left(\frac{\texttt{PRICE}_i - \texttt{midpoint}_{i-1}}{\texttt{midpoint}_{i-1}}\right)$

Table A.6: First 27 features in the final random forests model containing 53 features.

LAST 26 OF FINAL 53 FEATURES INCLUDED IN FORESTS MODEL			
Rank	**VIMP**	**Feature Name**	**Calculation**
28	0.004403	scld_PRICE_ofr30_2	$\left(\frac{\texttt{PRICE}_i - \texttt{ofr30}_{i-2}}{\texttt{ofr30}_{i-2}}\right)$
29	0.004344	scld_PRICE_ofr30_3	$\left(\frac{\texttt{PRICE}_i - \texttt{ofr30}_{i-3}}{\texttt{ofr30}_{i-3}}\right)$
30	0.004205	scld_PRICE_ofr30_1	$\left(\frac{\texttt{PRICE}_i - \texttt{ofr30}_{i-1}}{\texttt{ofr30}_{i-1}}\right)$
31	0.004006	scld_PRICE_ofr30_4	$\left(\frac{\texttt{PRICE}_i - \texttt{ofr30}_{i-4}}{\texttt{ofr30}_{i-4}}\right)$
32	0.003876	scld_PRICE_NBB_1	$\left(\frac{\texttt{PRICE}_i - \texttt{NBB}_{i-1}}{\texttt{NBB}_{i-1}}\right)$
33	0.003683	scld_PRICE_NBB_0	$\left(\frac{\texttt{PRICE}_i - \texttt{NBB}_i}{\texttt{NBB}_i}\right)$
34	0.003659	dif1_PRICE	$\texttt{PRICE}_i - \texttt{PRICE}_{i-1}$
35	0.003261	scld_PRICE_midpoint_4	$\left(\frac{\texttt{PRICE}_i - \texttt{midpoint}_{i-4}}{\texttt{midpoint}_{i-4}}\right)$
36	0.003246	scld_PRICE_bid30_0	$\left(\frac{\texttt{PRICE}_i - \texttt{bid30}_i}{\texttt{bid30}_i}\right)$
37	0.003073	scld_PRICE_midpoint_2	$\left(\frac{\texttt{PRICE}_i - \texttt{midpoint}_{i-2}}{\texttt{midpoint}_{i-2}}\right)$
38	0.003057	dif4_priceRelQuotes	$\texttt{priceRelQuotes}_i - \texttt{priceRelQuotes}_{i-4}$
39	0.002818	dif3_priceRelQuotes	$\texttt{priceRelQuotes}_i - \texttt{priceRelQuotes}_{i-3}$
40	0.002752	scld_PRICE_bid30_4	$\left(\frac{\texttt{PRICE}_i - \texttt{bid30}_{i-4}}{\texttt{bid30}_{i-4}}\right)$
41	0.002696	scld_PRICE_bid30_2	$\left(\frac{\texttt{PRICE}_i - \texttt{bid30}_{i-2}}{\texttt{bid30}_{i-2}}\right)$
42	0.002671	scld_PRICE_NBB_2	$\left(\frac{\texttt{PRICE}_i - \texttt{NBB}_{i-2}}{\texttt{NBB}_{i-2}}\right)$
43	0.002630	scld_PRICE_NBB_4	$\left(\frac{\texttt{PRICE}_i - \texttt{NBB}_{i-4}}{\texttt{NBB}_{i-4}}\right)$
44	0.002619	scld_PRICE_midpoint_3	$\left(\frac{\texttt{PRICE}_i - \texttt{midpoint}_{i-3}}{\texttt{midpoint}_{i-3}}\right)$
45	0.002370	scld_PRICE_bid30_3	$\left(\frac{\texttt{PRICE}_i - \texttt{bid30}_{i-3}}{\texttt{bid30}_{i-3}}\right)$
46	0.002352	scld_PRICE_NBB_3	$\left(\frac{\texttt{PRICE}_i - \texttt{NBB}_{i-3}}{\texttt{NBB}_{i-3}}\right)$
47	0.002202	dif1_priceRelQuotes	$\texttt{priceRelQuotes}_i - \texttt{priceRelQuotes}_{i-1}$
48	0.002201	dif1_priceRelNBO	$\texttt{priceRelNBO}_i - \texttt{priceRelNBO}_{i-1}$
49	0.002060	dif1_priceRelNBB	$\texttt{priceRelNBB}_i - \texttt{priceRelNBB}_{i-1}$
50	0.001794	dif2_priceRelQuotes	$\texttt{priceRelQuotes}_i - \texttt{priceRelQuotes}_{i-2}$
51	0.001715	dif2_priceRelNBB	$\texttt{priceRelNBB}_i - \texttt{priceRelNBB}_{i-2}$
52	0.001689	dif2_priceRelNBO	$\texttt{priceRelNBO}_i - \texttt{priceRelNBO}_{i-1}$
53	0.001249	ticks	See Section 5.4.1

Table A.7: Last 26 features in the final random forests model containing 53 features.