A Surprising Thing: The Application of Machine Learning Ensembles and Signal Theory to Predict Earnings Surprises

Derek Snow

Department of Accounting and Finance

University of Auckland

Owen G. Glenn Building

12 Grafton Road, Auckland, 1010

+6422 3922 056

d.snow@auckland.ac.nz

Contents

Tables

Figures

Abstract

Nonlinear classification models can predict future earnings surprises with a high accuracy by using pricing and earnings input data. Surprises of 15% or more can be predicted with 71% accuracy. These predictions can be used to form profitable trading strategies. Additional variables have been created using signal-processing and handcrafted feature-engineering methods. Some of these variables have in the past been known to be related to analyst bias. The machine learning model in effect corrects for analyst mistakes and biases by incorporating these variables into a nonlinear prediction model to predict future earnings surprises.

I. Introduction and Motivation

Zacks Research and Thomson research both have proprietary tools to predict earnings surprises that deconstruct individual analysts past performance. In this chapter I predict earnings surprises without deconstructing individual analysts' performance and instead focus on improving on the aggregated analyst consensus which is then used as one of the inputs in the surprise model. I allow the additional inputs and a machine learning model to identify and correct the mistakes and biases identified in consensus analyst forecasts. The earnings surprise classification model[1] shows an increase in accuracy compared to a random choice benchmark. The predictions are made with minimal effort by utilising a narrow set of data and incorporating carefully engineered variables as inputs to a gradient boosting machine (GBM) for a wide cross-section of earnings quarters.

Following from this, this chapter seeks to answer three research questions. Can machine learning be used to predict earnings surprises better than a random choice benchmark? If so, is this evidence of analyst bias and analyst mistakes? And, is it possible to form a simple trading strategy based on earnings surprise predictions? To address these questions, I use a classification task to predict the occurrence of earnings surprises and a trading strategy to identify whether the results produced by the classification model are economically significant. I show that past earnings, current analyst forecasts, and differences between the two are the most important variables for predicting future earnings surprises.

The specific task is to predict future earnings surprises for listed US firms using non-linear machine learning models and extensive variable engineering. I perform validation exercises for two model building stages. In the first stage, the model and variables are selected based on a validation set containing 15% of the data. This model and variables are subsequently used in the second modelling stage to dynamically adjust the model hyperparameters for four different periods preceding the test sets[2] to avoid information leakage. The methodology in the second stage consists of chronologically evaluating the performance of the model by keeping the size of the test set fixed while walking-forward and increasing the size of the training data for each successive split. In short, it is a walk-forward expanding-window validation process. Additional information on this methodology can be found on page 19.

[1] Machine learning terminology for predicting a discrete outcome variable.
[2] 1998-2003, 2004-2008, 2009-2012, and 2013-2016

I label three classification surprise buckets (negative, neutral and positive) for three different classification models of earnings that deviate more than 5%, 10%, and 15% from the expected value. Some notable results show that a classification model can predict a 15%+ positive earnings surprise, 53% of the time, while random guessing yields only 24%. Negative surprises of 15%+ can correctly be predicted 40% of the time, while random guessing achieves a meagre 11%. And the neutral class can be predicted 76% of the time compared to the 65% of random guessing. Further robustness checks include the use of different surprise thresholds to design a trading strategy in order to test the economic significance of predicted surprises.

Each day, I form a long (short) portfolio on stocks with a predicted positive (negative) surprise that deviates between -50% and +50% from analyst forecasted earnings. In the process, I identify the optimal surprise deviation parameter for the best trading strategy as tested on a validation set. The results show that the best trading strategies exist between 5% and 20%, with 15% surprise deviation being optimal.[3] I, show that an event-driven trading strategy that takes long positions in stocks with 15% positive surprise predictions, while earning the market rate of return over non-earnings surprise days, produces an alpha of 8% relative to a five-factor asset-pricing model on an out-of-sample test set (Fama & French, 2015). There is no good reason to form a long-short portfolio with negative and positive earnings surprises, as the constituent firms in the portfolio are not comparable and positive and negative surprise predictions do not necessarily fall on the same dates. For further tests see the trading strategy section on page 47.

The models used in this study incorporate readily available inputs in the form of historical earnings, analyst forecasts, and pricing data. I subsequently show that a handful of variables account for the majority of the prediction success. In predicting earnings surprises, I always incorporate analyst forecasts as an input. In chess, at least currently, human-machine collaborators are the most difficult opponents to beat (Kasparov, 2010). Similarly, the inclusion of analysts' estimates in a machine learning model leads to better performance than just relying on analyst estimates or a machine learning model without analyst estimates. Research by Kogan, Levin, Routledge, Sagi, and Smith (2009) showed that using historical volatility was better at predicting future volatility than using textual analysis scans over financial reports, but that when they combined both predictions in a single model, it significantly outperformed a model purely based on historical volatility. An important reason

[3] See the trading strategy section on page 56 for an elaborate explanation of the methodology used.

why human-machine collaborators tend to outperform has to do with the uncorrelated or dissimilar approaches they take to solve the same problem.

Accuracy and out-of-sample performance are two important concepts in building prediction models. The model can be made more generalisable by following certain procedural steps. Accuracy, especially "base" accuracy, is achieved through domain knowledge in choosing and obtaining relevant variables. Accuracy is enhanced by model selection, parameter adjustments, and data improvements. For this study, I have followed standard approaches for promoting generalisability, such as the creation of train-validate-test data splits in time-series to ensure that the model is tested only against unseen holdout-sets. In this chapter, base accuracy is achieved by selecting variables from three data sources, namely prices, analyst forecasts and earnings data. The set of variables is then expanded using feature engineering, which consists of interacting and transforming existing variables to create new variables.

The use of algorithmic models to improve prediction accuracy is only useful if researchers or practitioners can identify and understand domain specific biases. Machine learning techniques allow users to swiftly correct for analyst biases by engineering variables and feeding them into a high dimensional model in the attempt to 'reveal' these systematic biases to the model. Mullainathan and Spiess (2017) refer to three use-cases for machine learning in economics and finance: (1) To utilise new kinds of data; (2) To find new ways to analyse data; (3) To ask new questions. In this chapter I consider new ways to analyse old problems.

This chapter investigates two types of biases. I first identify earnings forecast biases from the literature, i.e., the systematic deviations of actual realizations from forecasts as reflected by past research (*Table A15*). Subsequently, I speculate on possible unobservable biases through the identification of the most relevant variables and variable groupings for beating analyst forecasts. I incorporate variables that appear to relate to forecast bias as additional inputs into the machine learning models. These variables are contained in the earnings, technical and signal processed variable sets, and are identified in the predictor analysis section. Similar to linear coefficients, the majority of the models identify the contributing factor of all variables. This allows us to theorise about relationships and associations in a multi-dimensional domain.

Researchers have yet to successfully incorporate contemporary machine learning methods in earnings prediction research. This study innovates in the field of finance and machine learning by mapping signal processing algorithms over existing time series variables

to carve out deeper patterns for 'machines' to analyse and learn to predict event outcomes. Furthermore, this paper finds a unique use for technical trading indicators to predict changes in earnings, rather than changes in future returns. It is the first machine learning research model to improve analysts' forecasts and predict earnings surprises by developing a model aware of potential biases. It is also the first paper to show that one can earn profits by predicting future earnings surprises. Lastly, this model is dynamic, in that the most important variables tend to change over time, possibly related to the process of analyst learning.

II. Literature

A. Earnings Surprise Prediction

Earnings surprises have been shown to influence stock prices (Graham & Dodd, 1934). The response of the stock price has been demonstrated to be statistically related to earnings announcements (Bartov, Givoly, & Hayn, 2002; Kasznik & McNichols, 2002). Managers also believe that they have to meet or exceed the market's earnings expectation to increase or maintain the share price (Graham, Harvey, & Rajgopal, 2005). From this, we can infer that the successful predictions of earnings surprises can be used to develop profitable trading strategies.

Ball and Brown (1968) provide compelling evidence that there is information content in earnings announcements. Strategies have been developed to take advantage of the difference between forecasts and surprises (Latane & Jones, 1977). The economic effects of surprises are not necessarily immediately noticeable as the market may take some time to reflect the perceived economic impact of the surprise (Bernard & Thomas, 1990). In contrast to the studies mentioned above, the purpose of this paper is not to prove the profitability of a strategy by exploiting the post-earnings announcement drift (PEAD); rather, the focus is on the short-term, same-day price reaction of the stock[4].

Numerous trading strategies can be fathomed that relies on having information on the likelihood of future earnings surprises. Brown, Han, Keon, and Quinn (1996) have developed earnings surprise prediction strategies using multi-factor regressions, focusing on variables such as stock returns, price-to-earnings ratios, book-to-market ratios, and firm size to predict future earnings, finding that past earnings surprises are an important variable driving future

[4] The next trading day is used for after-hours announcements.

surprises. My study is different in that I do not include non-earnings-related fundamental variables, and that I make use of nonlinear prediction methods. A study by Dhar and Chou (2001) makes use of genetic algorithms with moving averages and a comprehensive list of fundamental information as inputs for 12,164 observations from 1986 to 1997. In contrast, I show that better results can be achieved with a larger sample and a narrow set of easily accessible earnings and pricing data while using modern machine learning techniques.

The first part of the study focuses on prediction rather than hypothesis testing. This paper does, however, attempt to explain variable importance and their respective directions. Due to the nonlinear nature of the prediction methods, this analysis is more involved than simply running linear models and identifying their respective coefficients. In traditional finance, we mostly add interactions and combinations of variables manually when using conventional linear models; here the dimensions can be measured by kn. In ML (Machine Learning) the dimension is a result of the function one chooses, such as the number of nodes (ex. n) and the number of alternative choices to each node (ex. k) in a decision tree. For these tree models, k^n is a more accurate representation of the order of dimension. Further transformations of these trees into ensembles/meta-models lead to an even higher dimensional space (Joret et al., 2016). Using an out-of-sample data set allows a researcher to minimise over-fitting and improve generalisability by adjusting the complexity of a model; for econometric regression, this can be done using L1 or L2 regularisation; for a decision tree, this can be done by adjusting the depth of the trees.

There are a few notable points to consider when constructing an earnings surprise trading strategy, such as the possibility of stale earnings per share (EPS) forecasts (Lys & Sohn, 1990), the likelihood of low analyst coverage (Kinney, Burgstahler, & Martin, 2002), analyst forecast dispersion (Freeman & Tse, 1992), I/B/E/S exclusions (Abarbanell & Lehavy, 2002) and earnings preannouncement (Anilowski, Feng, & Skinner, 2007), all of which can have a big effect as to whether the market actually experiences an earnings surprise. If these factors excessively warp the analyst forecasts, then the defined surprise may merely be a surprise on paper and not be perceived as one by the market. Johnson and Zhao (2012) also remark that a large portion of returns are in the opposite direction of the earnings surprise. They do, however, say that the majority of such occurrences are observed in interior deciles. For that reason, this study incorporates various surprise thresholds to identify this effect on profitability.

B. Feature Engineering and Biases

Feature engineering is a machine learning term for the creation of variables from a raw set of data. For example, rolling average price features can be created from a series of historical prices. The first set of engineered features are generated from price data; in this study, I use the opening, closing, high, low, and volume data for each stock. With this data, I create a further 57 variables using a wide range of technical indicators. Many studies have attempted, and have to some extent succeeded, in making use of technical indicators in combination with machine learning models to predict future stock price movement (Kim, 2003; Patel, Shah, Thakkar, & Kotecha, 2015). In the 1960s, trading rules, based on technical indicators, were said not to be profitable (Fama & Blume, 1966). However, these indicators were never used in combination with learning algorithms and, more specifically, never applied for event prediction. In prior literature, no study makes use of technical indicators to predict financial event outcomes as opposed to changes in the stock price.

In this chapter, I apply signal process mappings over all price and volume containing variables to calculate, among others, Langevin fixed points and Fast Fourier transform coefficients. Signal processing algorithms have been used in finance before. Ramsey and Zhang (1997) used waveform dictionaries to decompose signals contained within the foreign exchange market, and a Langevin approach was used to describe stock market fluctuations and crashes (Bouchaud & Cont, 1998). To my knowledge, there is also no academic study that creates variables from such signal processing decompositions on pricing data to predict financial events like earnings announcements. Making use of signal processing techniques allows us to characterise price and return patterns before announcements by mapping hundreds of functions over a time-series of price and technical variables and testing the relevance of each variable on a holdout set. This gives the machine learning model an added advantage in uncovering patterns and associations for an enhanced understanding of the stock price before abnormal events.

The variable engineering part of the paper takes inspiration from control system engineering to try to model and describe the shape of pre-reaction curves before earnings announcements. This study shows that second-order system parameters such as magnitude and peak time, for historical and technical price series, provide unique insights into data that is not necessarily revealed by traditional pricing and technical indicators alone, as technical indicators are constrained by design to facilitate a specific financial use case. To calculate the parameters, a range of algorithms is applied to price and technical time-series for the 30

days leading up to the announcement date. For a list of these algorithms, please see Appendix *Table A14*.

After consulting past research, I include additional variables deemed most useful for counteracting analyst biases. The first is a running forecast error to identify systematic over- and under-prediction (Butler & Lang, 1991; Fried & Givoly, 1982). Another measure is the rolling average percentage difference in forecasts to track the 'stickiness' in the forecast of earnings (Givoly & Lakonishok, 1984). Barron, Harris, and Stanford (2005) provide empirical evidence that private information inferred at the time of an earnings announcement is correlated with greater trading volume; I, therefore, include volume as a model input. Furthermore, the variance of volume before an announcement may also be a promising input (Beaver, 1968). Two additional types of measures include a skewness measure of past earnings from research by Butler and Lang (1991) and a range of dummies to identify whether or not the last earnings were above surprise thresholds of 5, 10 or 15 percent, due to the tendency of earnings surprises to repeat (Brown et al., 1996). Other variables include the number of analysts covering the firm (Lys & Soo, 1995); this measure also serves as a means to quantify the level of public information available for each stock (Das, Levine, & Sivaramakrishnan, 1998). Das et al. (1998) argue that when earnings are less predictable, analysts have a stronger incentive to issue optimistic forecasts; I will include a standard deviation of earnings measure to proxy for this uncertainty. For a full list of these biases and whether they form part of the top ten most important features in subsequent results, see *Table A15* in the Appendix.

C. Machine Learning

Ensemble learning refers to the weighted voting of multiple models (Dietterich, 2000). In this study, the models are constructed from decision trees. There are two traditional ways to execute an ensemble strategy, namely bagging or boosting. This study will use a versatile boosting model very similar to Gradient Boosting Decision Trees defined by Friedman (2001) known as XGBoost (Chen & Guestrin, 2016). Boosting is the process of fitting an initial model to predict a target value after which new models are subsequently fitted on the errors of the previous step to improve the final prediction model. Gradient boosting for classification models takes the additional step to fit the iterative models on the gradient of a log-loss (cross-entropy) function in order to minimise a differentiable function.

In the past, more research has been conducted in the univariate category than in the multivariate category, but that has slowly changed over the years. The use of machine learning in finance is also becoming more common as researchers slowly uncover the nonlinearity of financial data, as has shown to be the case for quarterly earnings per share data (Callen, Kwan, Yip, & Yuan, 1996). Callen et al. (1996) showed that machine learning models have for a long time been able to beat time-series models in forecasting. Xiao, Xiao, Lu, and Wang (2013) demonstrate the power of ensembles in financial market forecasting; they show that the flexibility of the ensemble approach is key to their ability to capture complex nonlinear relationships to predict future stock prices. And finally, Gu, Kelly, Xu (2018) shows how machine learning methods can be used in empirical asset pricing.

As outlined by Kuhn and Johnson (2013), I select the best model by starting with the most flexible and best-performing models as disclosed in past research, and I analyse their performance on a subsection of data to establish a performance ceiling, after which I select the best model. I further compare the performance of different types of Neural Networks, Support Vector Machines, Random Forest, Naïve Bayes, Adaptive Boosting, and Extreme Gradient Boosting Decision Trees models. In traditional finance research the acceptability of empirical results generally hangs on the requirement for interpretable causality, in this study, accuracy, associations, profitable trading strategies, and measures of variable importance trump the interpretability of the model. The study does not attempt to demonstrate causal relationships but rather the relevance of previously unstudied variables and the performance of machine learning to predict earnings surprise outcomes.

Applied machine learning has gradually made its way into finance. One of the first quality papers came from Teixeira and de Oliveira (2010) who used economic and financial theory in combination with fundamental, technical, and time-series analysis to predict price behaviour using artificial neural networks. Other researchers like Bagheri, Peyhani, and Akbari (2014) used an adaptive networked-based fuzzy inference system to forecast financial time-series for currencies while Hu, Feng, Zhang, Ngai, and Liu (2015) used a hybrid evolutionary trend following algorithms to introduce a trading algorithm that selected stocks based on different indicators. The majority of research in this field has focused on the price movements of stocks, indices, and currencies. These studies are limited in sample size, most opting to analyse a small number of stocks, making it difficult to rule out random results. Very few studies look at financial outcomes; additionally, no study has used modern machine learning techniques to investigate the probability of financial event outcomes in order to apply it in a trading strategy. The study as presented in this paper serves as the foundation for

a novel system that aids investors and market makers in managing their stock ownership before earnings announcements, not just for profit maximisation but also for risk management purposes.

III. Data

All available quarterly earnings per share measures were obtained from the Thomson Reuters' I/B/E/S Detailed File for publicly traded firms in the US, starting from the first available date in 1983 to 2016. I make use of the detailed file due to known rounding errors in the I/B/E/S summary file (Payne & Thomas, 2003). As per Claus and Thomas (2001), I dropped the quarters before 1984 from the I/B/E/S database; before this date the database provided too few firms with complete data to represent the overall market. The starting number of observations are 455,142 firm quarters. Daily stock information was sourced from CRSP's Daily stock file. I exclude observations flagged as "Excluded" or "Stopped Coverage" by IBES. Analyst ratings had to be published two months before the actual announcement, following Behn, Choi, and Kang (2008). This establishes a fair method to compare the machine learning model's performance against timely analyst forecasts. This resulted in 313,416 firm quarters.

I also include only the most recent forecast of each analyst for every brokerage house for each earnings period; this further decreased the number of firm quarters used in the preceding calculation to 175,176. Other data cleaning operations include removing all analyst forecasts that appear after the announcement date and entries where the ticker or forecasted value is null, removing about 3% of the observations. There is no explanation other than the entries being in error. A further requirement is for the firm to have 6 years of prior financial information to create rolling earnings-related variables, leaving 158,224 quarters for the main prediction algorithm. The amount of trailing data is similar to O'Brien (1998) and Kross, Ro, and Schroeder (1990), who have used around seven years of trailing data for their time-series models. Cutting away 6 years leaves us with 26 years of data, a period from 1990-2016, to perform the machine learning operations on.

In this study, 15% of the data is used for model validation, hyper-parameter tuning, and feature selection process after which I disregard the data to uphold prediction integrity, leaving 134,490 observations.[5] This validation process is used to select the model, reveal the

[5] A smaller size of 15% of the data is used for validation because with this particular validation method the data ultimately gets dropped and decreases the size of future training, validation and test data.

starting hyperparameters, and select the features. It is unrelated to the validation process used to develop the models that report the final metrics using a chronological, walk-forward, and expanding-window validation method to dynamically adjust the hyperparameters and test the performance of the model throughout time. Additional details are provided in the next section.

IV. Methods

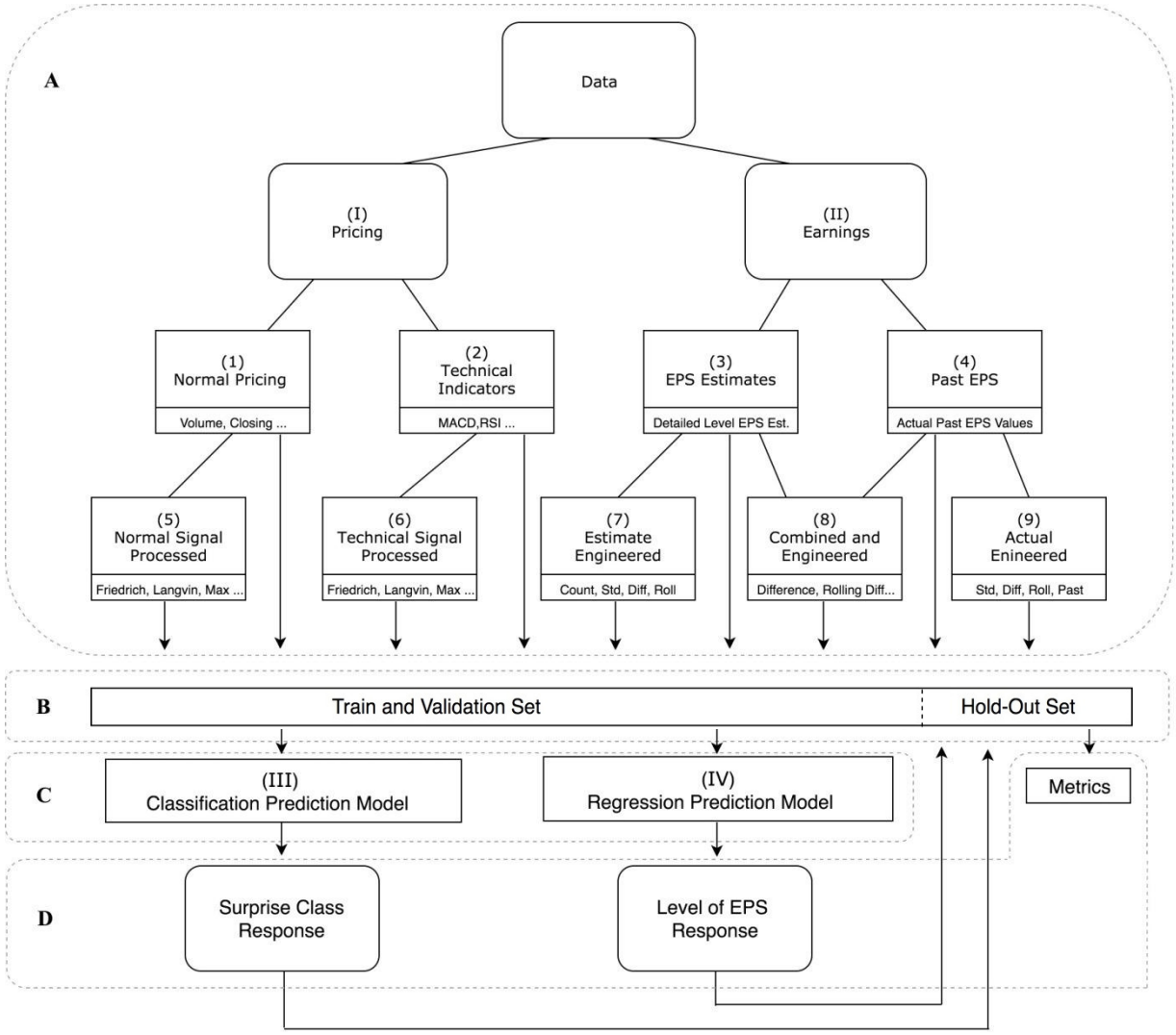
This section develops the methodology of the machine learning classification task. Readers who are interested in the results of this study rather than the methods used should skip to Classification *Section V* or straight to the table on page 27. The empirical part of the study consists of a variable creation stage after which a training dataset is used to train the prediction algorithm. Each quarterly surprise classification is described according to a set of variables and a response value. The trained model is used to make predictions against a test set to, among others, assess its accuracy. This study goes beyond simple machine learning; it does not just report on how well the prediction fits the test set but also sheds light on what the predictions tell us about the quality of analysts' predictions, and whether the algorithmic predictions can help to uncover analyst biases. The first stage will follow the standard machine learning process; the second part will focus on what these predictions tell us about analysts, and how these predictions can be used in practice, such as in establishing trading strategies and improving forecasts. *Figure 1* is a critical diagram revealing the process of obtaining data, creating variables, and training and testing a model. The headings and associated labelling in this section correspond to that of the Figure.

A. Variables

In this study, the black-box prediction model incorporates information leading up to the day before a firm's official earnings announcement date to make a prediction of the forthcoming earnings event. To make these predictions, variables are constructed from a wide range of datasets by applying multiple transformations over time-series data so as to convert them to a series of cross-sectional values. The majority of these transformations are derived from signal processing literature. The final set of single value inputs can then be fed into the machine learning model. *Figure 1* shows the type of variables used in the model. There are

multiple input variables and the relations between these variables play an integral part in the models' prediction success.

Figure 1: Process Tree



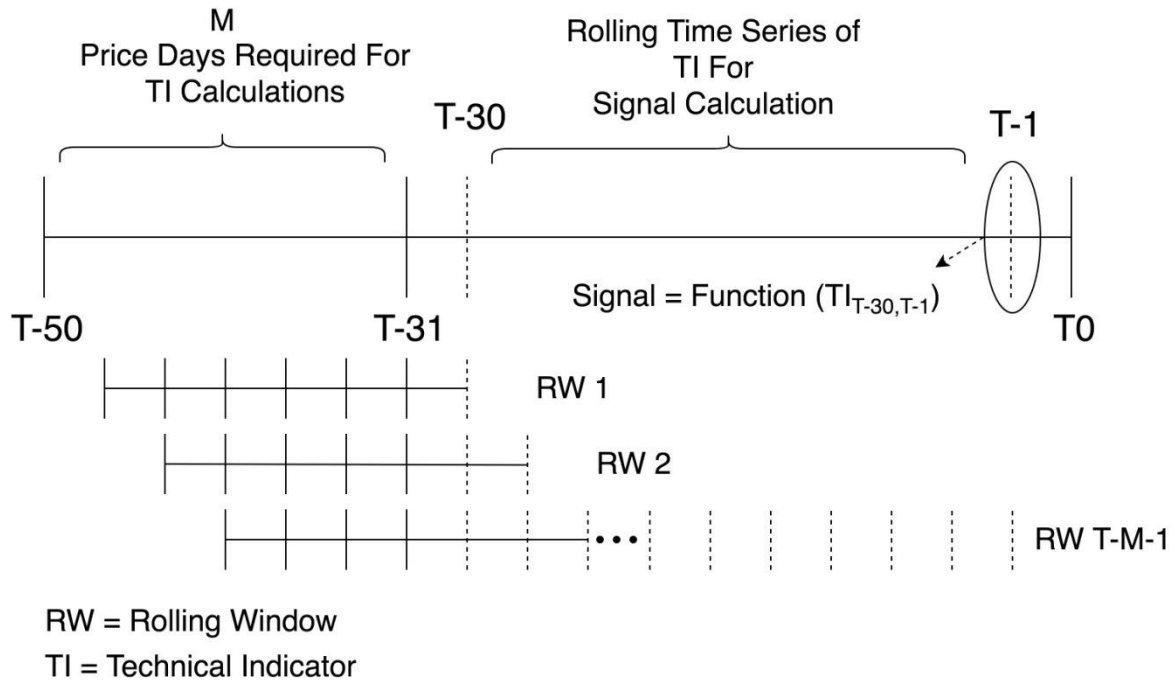
(A) Items (1) - (9) constitute the final groups of variables that would be used in this study. All of these variables are derived from pricing and earnings-related datasets. The earnings-related data can further be deconstructed into (3) EPS forecasts and (4) Past EPS values and hand engineered variables (7) - (9). (1) In total, there are no more than 5 normal pricing variables, (2) 57 technical indicators, (3) 1 EPS forecast, (4) 1 EPS value, (5) 20-50 normal signal processed measures, depending on the task, (6) more than 500 technical signal processed measures, (7) 22 analyst-only variables, (8) 22 combination variables, (9) and 23 actual-only variables. (B) identifies a generic train-validation-test split. (C) identifies the black-box prediction models. In this chapter we will only look at classification models. (D) is the response variables and calculation of performance metrics on a hold-out set classification metrics.

1. Pricing Variables

Of the average 62 trading days between announcements, (1) 50 trailing days of pricing data are needed to create a standard set of technical variables. (2) Of these 50 days, 20

trailing days are "consumed" by a technical indicator transformations process. In (5) - (6), the last 30 days of the bundle of pricing and technical variables transform down to a single number by undergoing a range of signal processing transformations. Such a deep analysis will likely pick up on insider trading as well as other pre-patterns associated with earnings surprises. *Figure 2* illustrates the process from (1) to (6). The appendix item *Method A 1* describes this process in greater depth and includes an explanation of the variable selection process (the process of removing variables which seem irrelevant for modelling). Also, for a comprehensive list of the signal processes mappers, see *Table A14*.

Figure 2: Transforming Price to Technical and Technical to Signal



In this study, I use 50 pricing days to calculate the final signal. The first step is to calculate a time-series of the technical indicator. Depending on the indicator, the function can incorporate between 3 to 20 past pricing days. The resulting time-series covers 30 days for all technical indicators in the study. The calculations do not include the price at *T* as this information is not available to us at the time of prediction, *T-1*. The next trading day is time *T*, for days where the earnings announcement occurs after-hours. The time-series of technical indicators gets fed into a signal processing function, which calculates a singular value from the time-series. Following is an example to better understand what variables get included in the learning and prediction algorithm. The model will include as inputs the closing price at *T-1*, the last rolling value of the technical indicator at *T-1*, and the singular signal processed value as calculated at *T-1*. The above figure only describes the process for technical indicators to signal values, but normal pricing data also gets signal processed as shown in *Figure 1* (5).

2. Earnings Variables

The earnings variables are constructed from past earnings per share (EPS) forecasts and actual EPS values. The machine learning phrase for the process is "feature engineering." For the variables (7) - (9), the calculation involves multiple timeframes of rolling averages, weighted averages, lagged values, past differences, the standard deviation of forecasts and a count of the number of analysts per forecast. A lot of the inspiration for the above variables' calculation selection was drawn from analyst biases and mistakes as noted in past literature. A table identifying some of these variables can be found in the Appendix, *Table A15.*

3. Response Variable

For the classification task[6] (*Figure 1* (C)) the response variable for the machine learning model is the occurrence of an earnings surprise. An earnings surprise is simply defined as a percentage deviation from the analyst's EPS expectation, as described in the data section, and the actual EPS as reported by the firm. In this study, we include percentage thresholds, *S,* as a means of expressing the magnitude of a surprise so as to construct various tests.

$$X = \frac{EPSAC_{it} - EPSAN_{it}}{EPSAN_{it}} - 1$$

$$SURP_{itsx} = 0, \ where \quad X < -S, \ \text{Negative}$$

$$SURP_{itsx} = 1, \ where - S \leq X \leq S, \ \text{Neutral}$$

$$SURP_{itsx} = 2, \ where \quad X > S, \ \text{Positive}$$

i = Firms in the sample

t = Time of the quarterly earnings announcement

S = A constant surprise threshold, 5%, 10%, or 15%

x = A constant percentage of samples sorted by date of earnings announcement

[6] The task predicting a binary dependent variable.

$EPSAN$ = Mean analyst EPS forecast

$EPSAC$ = The actual EPS measure as reported by the firm

This surprise measure is simply the difference between the actual and expected EPS scaled by the expected EPS. This measure is similar to Foster, Olsen, & Shevlin (1984), the difference being that they used the absolute actual earnings as the denominator instead of expected actual earnings. I have tested three other variants where the actual EPS is the denominator and is absolute in value; this led to a small but non-significant improvement (Foster, Olsen, & Shevlin, 1984). I also looked at the level of earnings using the same formula, which produced the same results as EPS, and finally a standardized unexpected earnings (SUE) measure that also led to a small but non-significant improvement in returns (Brown et al., 1996). I found that all these measures are highly correlated from 85% upwards so I only report the most obvious solution. What is important is whether the prediction of surprises as defined leads to positive abnormal returns. If it does, then one is said to have a definition of earnings surprise similar to that of the market. It is also worth noting that I set three thresholds and that these thresholds mostly eliminate issues regarding highly positive or highly negative values when it is close to zero.

B. Train, Validation and Test Sets

The model building in this chapter occurs in two stages. The first stage lays the initial groundwork by choosing the model type and variables that would be used in all future model iterations. In this first stage, various types of models with multiple sets of hyperparameters are compared to each other and the best model is used to perform feature selection. The second stage uses the first model to dynamically adjust the hyperparameters over time. In the second stage, no additional model selection and variable selection procedures are performed; the models in this second iteration are used to report the prediction results on the test data. The methodology of the second stage are presented in *Figure 3;* it consists of chronologically evaluating the performance of the model by gradually increasing the window size while keeping the size of the test data fixed.

To clarify quantitatively, in the first stage, the data is sorted by time, and the first 20% is used for training while a random selection of 15% of the remaining data is used for validation. The first modelling stage uses a fixed but chronological train-validation split to perform model, hyperparameter, and variable selection, after which the validation data are

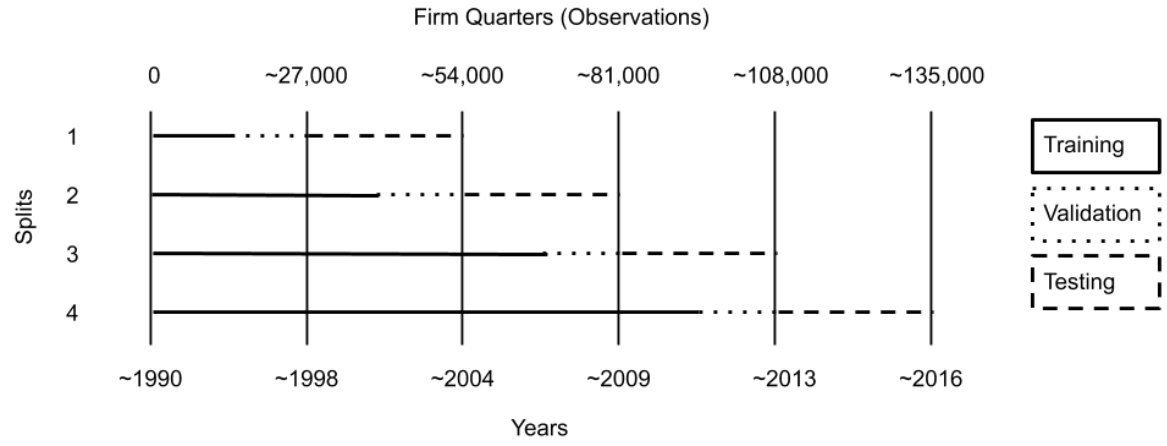
dropped indefinitely. This percentage of data used for validation is kept small because this particular data cannot be used for future training, validation, and testing; the data ultimately gets dropped after use.

This is the safest possible approach to build an applied machine learning model in time-series. The model type and variables selected in stage one remains constant in stage two while only the internal model parameters and hyperparameters are allowed to adjust freely in each of the five successive time-splits. As more data becomes available, the training set increases allowing for improved machine learning prediction. Although the test set stays constant in size, it shifts forward to test distinct non-overlapping periods. The testing data never contains data that is younger than the training data, which is a sensible step for preserving prediction integrity; the validation data never gets recycled as testing data. This particular walk-forward model is also helpful in that the model can easily be tested for robustness over time without a large gap in the data that would normally result from alternative validation procedures in time-series.

The first model in the second development stage starts out with the same 20% of the original time-ordered data, but this time it is used for both training and validation purpose. The data is chronologically split into five chunks each containing 20% of the data (~27,000 observations). The size of the validation set for each iteration is 10% of the original size of the data. For each iteration, the test data is fixed at ~27,000 observations and the validation data is fixed at ~13,500 while the training data increases with each step forward. The size of the validation data has been set based on the objective to have the most up to date hyperparameters for the test data. The validation data is not dropped in this model development stage but ingested by a second model that is trained on both the initial training and validation data. This is done to ensure that the timeliest training data is included in the model to more accurately adjust internal model parameters. This final model is then applied against a test set.

Once the model has been selected and specifications made, it is necessary to measure the out-of-sample performance of the model. This method is well suited for time-series evaluation where there is an expectation of structural changes in pattern change over time. This technique can be used for both classification and regression tasks (Bergmeir & Bentez, 2012). Although the test set stays constant in size, it shifts forward to evaluate distinct non-overlapping periods. To calculate the final result, I compute an average value across all four equally sized test sets and calculate a confidence interval.

Figure 3: Expanding Window Train-Validation-Test Splits



To ensure consistent performance measurements on the test splits, they should be the same size. In this study, the data is split into five equal-sized sections. And the model is trained on four of the five sections and tested on four of the five sections. For each additional section, the model trains on an increasing number of samples ordered by date. This study reports both the overall accuracy and breaks the accuracy down for each period and surprise threshold in question. This table does not show a separate process used to do feature and hyperparameter selection (the process of removing variables that seem irrelevant for modelling), which appears in the first model building stage. The feature selection is done on a small validation set constituting 15% of the data to ensure that during the development stage there is no 'double dipping' into the data; therefore, the model always gets tested on a fresh out-of-sample dataset.

C. Machine Learning

1. Black Box Understanding

The prediction algorithm used in this study can be viewed as a 'black box.' Below is high-level pseudo code to provide a better understanding of some of the core concepts of the black-box model such as its relationship with the training set, test set, validation set, predicted values, and metrics.

(1) $TrainedModel = ModelChoice(TrainInputs, TrainTarget, Valid, Param)$

(2) $Predictions = TrainedModel(TestInputs)$

(3) $Metrics = Functions(TestTarget, Predictions)$

$ModelChoice$ is the model used to approximate a function that closely resembles the target (response) function. I use an XGBoost model (Extreme Gradient Boosting Decision Tree) developed by Chen & Guestrin (2016). $TrainInput$s are the inputs of the training data, meaning all the data except the target variable. $TrainTarget$ are the target values we want to

train the model on, also known as the response variable in social sciences. $Valid$ is the validation data that will be used to adjust the hyperparameters. $Param$ is a list of 23 hyperparameters that can be tweaked to improve the model's performance. The parameters mostly relate to adjustments in the complexity of the model, these values are not learned by the model, but enough techniques exist so that external models can approximate the best parameter values for the prediction problem at hand. $Prediction$ is the predicted target value obtained by running the inputs of the test set through a trained machine learning model. $TestTarget$ is the actual target variable that the model tries to predict. $Functions$ are the numerous metrics that can be applied to evaluate the success of matching the target variables with the predictors on the test set. For the classification task, the main metrics used are accuracy and ROC (AUC) scores. What follows is an example of the pseudocode for the classification task.

(1) $Classifier = XGBoostTreeClassifier\big(Train_X, SURP_{its(0\,:\,x)}, Valid, Param\big)$

(2) $PredSURP_{its} = Classifier(Test_X)$

(3) $Metrics = Functions\big(SURP_{its(x\,:\,1)}, PredSURP_{its}\big)$

The prediction values, $PredSURP_{its}$, of the classifier are a categorical variable that falls within the values $\{0,1,2\} \rightarrow \{\text{Negative Surprise}, \text{No Surprise}, \text{Positive Surprise}\}$, for different surprise thresholds s $\{5\%, 10\%, 15\%\}$ of firms *i* at time *t*. If we assume that the training set is 60% of the original data set, then the training set's target value is $SURP_{its\ (0\%\,:\,60\%)}$, being the first 60% of the dataset ordered by date.[7] A section of 15 percentage points in the 60% of training data are used as validation data ($Valid$) to select hyperparameters ($Param$). The test set's target values are $SURP_{its\ (60\%\,:\,100\%)}$, i.e., the last 40% of the dataset. The metrics for a classification task comprises of accuracy (proportion of correctly classified observations), precision (positive predictive value), recall (true positive rate), as well as confusion matrices/contingency tables.

As a result of defining the surprise thresholds and discretising it into three buckets, there is a possibility that some information is lost. Instead of developing a multi-classification

[7] In this example the validation data is included in $Train_X$ because after the validation is performed and parameters selected on a section of the training data, a new model is formed to include the validation data as training data to get up to data information for future predictions; similar to the second stage of model development in this chapter.

model, one can predict the level of earnings using a machine learning regression model[8] and then convert the predicted dollar earnings into a predicted surprise percentage after which you allocate it to the respective surprise buckets. It could be that a method that predicts the level of earnings and discretise post-model is more accurate than a method that discretises before the model is trained, but this has to be empirically tested. Another alternative is to perform ordinal classification as opposed to multiclass classification as the surprise categories are ordinal in nature. All of these methods have been tested on validation data, and the multiclass classification model performed the best. I suspect it is due to the minimisation of noisy predictions as a result of using a small number of categories and also due to distinct nature and distribution of the defined earnings surprise outcomes.

2. Model of Choice

Machine learning is defined as the study of inductive algorithms that 'learn' (Provost & Kohavi, 1998). For the purpose of this study, it is valuable to have an intuitive grasp of the XGBoost machine learning model. XGBoost is short for Extreme Gradient Boosting. It is a nonlinear inductive algorithm used to approximate the function between inputs and outputs. The idea behind Gradient Boosting is to 'boost' many weak learners or predictive models so as to create a stronger overall model. The training process iteratively adds additional trees to reduce the errors of prior trees that are then combined with previous trees to produce the final prediction.

To create the overall ensemble model, such as the *Classifier* model described in the pseudocode above, we have to establish a loss function, L to minimise in order to optimise the structure and performance of the model. This function has to be differentiable as we want to perform a process of steepest descent, which is an iterative process of attempting to reach the global minimum of a loss function by going down the slope until there is no more room to move closer to the minimum. We, therefore, solve the optimisation by minimizing a loss function, $f(x)$, numerically via the process of steepest descent. For our classification task, we use logistic regression to obtain probabilistic outputs of the target (response) classes. In normal gradient descent one updates model parameters like the coefficients in a linear regression or the weights in a neural net, however, gradient descent in XGBoost essentially "updates" the model by adding new trees instead of updating coefficients or weights.

[8] Machine learning terminology for a model predicting a continuous value.

Furthermore, it is necessary to minimise the loss over all the points in the sample, (x_i, y_i). For a more detailed description of this process and other more involved formulae see the appendix *Method A 2.* The minimisation is done in a few phases. The first process starts with adding the first and then successive trees. Adding a tree emulates adding a gradient-based correction. Making use of trees ensures that the generation of the gradient expression is successful, as we need the gradient for an unseen test point at each iteration, as part of the calculation $f(x)$. Finally, this process will return $f(x)$ with weighted parameters. The detailed design of the predictor, $f(x)$, is outside the purpose of the study, but again for more extensive computational workings see the appendix *Method A 3.* At this point of the study, all the steps in *Figure 1* from (A) - (C) have been dealt with; the next step (D) evolves the evaluation of performance metrics, detailed prediction analyses, and the description of prospective trading strategies.

V. Classification

A classification task involves a classifier that assigns an instance (observation) to every class (category) based on the learned patterns of a training set. The training consists of past observations in which the classes are known. The model, therefore, learns class associations from the past patterns of explanatory variables, commonly called features, and maps this input data into a class according to a newly learned, weighted, and approximated function. The XGBoost classifier used in this study is a probabilistic classifier which simply outputs a probability of an instance belonging to one of the specified classes.

A probabilistic classifier is especially useful because the magnitude of the probability can itself be seen as a confidence value associated with the class choice. For example, if the probability of both a positive and negative earnings surprise is high for a single instance (earnings quarter), you may not want to follow through with a trade on that particular stock, but if the difference in probability is positive, you can be more at ease. The class probabilities can be used as parameters in a trading strategy.

A positive and negative surprise is defined as an occurrence in which actual observed earnings deviate 5%, 10% or 15% more than analysts' consensus forecasts. In this study, the focus is on a multiclass classification problem, which includes the above-mentioned surprise classes as well as a neutral class that lives between the positive and negative surprise thresholds. Past researchers such as Johnson and Zhao (2012) have noted that reaction in the share price can be in the opposite direction, especially in the inner deciles of surprises,

making it necessary for us to investigate surprises that are not just positive or negative, but positive and negative with more than, for example, a 5% deviation. The surprise threshold can also be viewed as a parameter that can be adjusted to suit the prediction task at hand. For a certain type of trading strategy, you might prefer a 10% or 15% surprise as it minimises false positives.

I present the classification results in four main ways. I first present an accuracy measure and make use of an inductive technique to identify the importance of groups of variables that explain the model success. Thereafter, I move into alternative metrics such as precision and recall by means of illustrative confusion-matrices/contingency tables. For the last-mentioned steps, I produce benchmark scores based on random choice to easily compare against the accuracy and precision metrics of the model. After this first form of analysis, under the "prediction analysis" section, I present the important predictors of the model in tabled and in graph form so that the reader can appreciate the vast range of predictors and the nonlinearity of the model. The last method of analysis involves testing different periods of the sample period and different thresholds of surprise. I finally end this section with a trading strategy to show the application value of the model and to confirm that an 'earnings surprise,' as defined in this model, is not just a surprise on paper, but also a surprise to the overall market.

A. Evaluation

The accuracy can be defined as the percentage of correctly classified instances (observations) by the model. It is the number of correctly predicted surprises (true positives) and correctly predicted non-surprises (true negatives) in proportion to all predicted values. It incorporates all the classes into its measure $(TP + TN)/(TP + TN + FP + FN)$, where *TP*, *FN*, *FP* and *TN* are the respective true positives, false negatives, false positives and true negatives values for all classes. The measure can otherwise be represented as follows:

$$acc(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(\hat{y}_i = y_i) \qquad (1)$$

Table 1 represents the accuracy measures for three surprise thresholds. The percentage surprise represents the threshold of the surprise. The overall accuracy of selecting

the correct class out of three classes is 63%; this number is reported in the last row of the 4th column of the table. By always selecting the highest populated class, a benchmark accuracy of just below 50% can be achieved. This can only be done if the benchmark is made 'knowledgeable' about the underlying distribution. The model leads to a 15-percentage point improvement over this benchmark. This benchmark is a reported figure in *Table 1* (5). In past literature, this benchmark normally 'peeks' at the distribution of the *training* data to make its predictions accordingly; however, for added robustness and to create a higher-level watermark for the forthcoming trading strategy, I allow the benchmark to peek at the class distribution of the *testing* data. It would use this information to select for the percentage of the most occurring class to calculate an accuracy benchmark.

After *Table 4,* I further deconstruct benchmarks, and show, by way of example, the difference between a benchmark focused on accuracy and a benchmark focused on precision. The above-reported aggregate scores can be broken down to calculate the accuracy of predicting surprises of certain predefined magnitudes, for example, 5% or more. This number incorporates both sides of the deviations - i.e., a 5% or more upwards deviation, being a positive surprise, and a 5% or more downwards deviation, a negative surprise. The confidence interval reports the 95% confidence interval derived from four different train-validate-test splits calculated for each surprise threshold.

In *Table 1,* I have included the accuracy results as more groups of variables are introduced to the model, (1) - (4) in the table. These variable groups are presented in *Figure 1*. This process is done to identify the additional accuracy each group of variables deliver to the final outcome. From left to right, the model includes actual earnings variables, analyst variables, basic pricing and technical variables, and lastly, all signal-processed variables whilst calculating accuracy metrics at each step. This method is similar to an inductive theory testing method promoted by Mullainathan and Spiess (2017). The difference is they started with a completed model and worked their way backwards.

The approach is noteworthy as it goes beyond the task of simple prediction by also decomposing the importance of the type of variables in the study, providing added value to the literature. The results show that price and price-derived variables (3 and 4 in table) contribute to 42%[9] of the improvement over the benchmark; past realised (1) earnings variables contribute to about 19% of the improvement; (2) and the analysts and interaction variables with analysts and actual reported earnings variables provide for the further 39% of

[9] $(0.032 + 0.031)/0.150 \approx 42\%$

the improvement. *Table 1* shows, empirically, the importance of the different groups of variables. The table also demonstrates that the accuracy increases as ever higher surprise threshold are tested. This is likely due to a bigger 'neutral' surprise class becoming a more obvious prediction but could also be due to the identification of more distinct characteristics and patterns associated with firm earnings quarters at larger levels of surprises.

Table 1: Accumulative Accuracy Comparison Table - Surprise & Non-surprise

Surprise	(1) Act.	(2) Frc.	(3) Price.	(4) Signal		(5) Bench		(6) Improved	(7) Confidence
5%	0.433	0.508	0.531	0.550	-	0.398	=	0.152	+/- 0.018
10%	0.490	0.566	0.601	0.633	-	0.472	=	0.162	+/- 0.024
15%	0.603	0.630	0.667	0.711	-	0.573	=	0.137	+/- 0.033
Average	0.509	0.568	0.600	*0.631*	-	*0.481*	=	*0.150*	*+/- 0.025*
(8) Accu.	0.028	+ 0.059	+ 0.032	+ 0.031			=	0.150	

This table compares the various surprise thresholds' accumulative accuracy for different variable sets. (1) are the contribution of actual earnings-related variables. (2) the analyst forecast variables and analyst interaction variables with historical earnings. (3) the original price and technical indicator variables. (4) includes the signal processed variables over price variables and is also the final model. (5) is the benchmark as the most frequently occurring class. (6) is the average percentage point improvement of the full model over the benchmark for all surprise thresholds. (7) is the confidence at 95% level. (8) The accumulative improvement starts by deducting the benchmark average (0.481) from the model earnings related variables (0.509), i.e., 0.028; and each successive improvement is the additional accuracy contribution from adding more data (0.568-0.509; 0.600-0.568; 0.631-0.600). This is only one of many methods to measure variable importance, other methods simply permute the features of interest and retrain the model. The problem with permutation methods is that it does not give accurate importance scores with collinear data, whereas the problem with this method is that the order of introduction matters because of interaction effects.

The accuracy score is not always very informative in an imbalanced classification study, as it does not look at class breakdown precision, nor does it provide evidence of true positive or true negative values. Accuracy measures work better for balanced than imbalanced datasets; the problem is that the 'neutral' class comprises the bulk of the data — hence the relatively high benchmark value obtained above by simply predicting the same category. The issue is that the accuracy for the neutral class is not useful for a trading strategy. In finance, especially when constructing trading strategies, you would generally prefer improvements in precision, True Positives / (True Positive + False Positives), as opposed to accuracy. Similarly, in most strategies you prefer precision above recall, True Positive / (True Positive + False Negative). A low recall means lost opportunities, but as long as there is a diversified portfolio or enough observations, a low recall is not a problem. A

trader would rather be concerned with the precision metric, which is the category accuracy of the predictions made.

The next section explores various confusion matrices/contingency tables. Each category has its own recall and precision fraction. All confusion matrix tables are formed by running models over four serial timeframes for 3 different surprise thresholds, being 5%, 10%, and 15%. For each model, 26,895 class predictions get tested against the true classes. This equates to 322,740 predictions in total. This should not be confused with the original sample size, as these are aggregated across three different surprise thresholds.

Table 2: Aggregated Surprise vs Non-Surprise Confusion Matrix

Confusion Matrix		**Predicted**		Sample Distribution
		Non-surprise	Surprise	
Actual	Non-surprise	**144,417**	54,534	0.62
	Surprise	64,428	**59,361**	0.38
Precision		0.69	0.52	
Improvement		0.08	0.14	322,740

This confusion matrix creates a summary by aggregating all the surprises, positive or negative, together, and separately all the non-surprises. Non-surprises consist of neutral or wrongfully predicted positive or negative surprises. The purpose of this matrix is to gain an understanding of the model's overall performance without having to discriminate in the direction of the surprise (+/-) or the threshold of the surprise (5%, 10%, 15%) or the period over which the test was done (4 splits). The *sample distribution* is equal to all the true observations of a certain classification divided by all the observations; an example along the first row: (144+54)/(144+54+64+59) ≈ 62%. The *precision* is calculated by dividing the true positives (Surprises) with the sum of itself and the false negatives (Not non-surprises). An example along the first column: $144/(144+64) = 69\%$

Table 2 shows a breakdown of predicted and actual classes for observations of surprises and non-surprises. Surprises incorporate both negative and positive surprises. This table, by means of the precision measure, digs deeper than the overall accuracy measure that has been reported in *Table 1*. "Improvement" in the above results show preliminary evidence that the model is better at predicting surprises than non-surprises; improvement is calculated by deducting the underlying sample distribution from the precision scores. This again highlights the potential of a trading opportunity. This improvement can also be expressed by drawing up a confusion matrix from random guessing. *Table 3* shows that for both of the classification groups, the model performs better than random guessing based on the underlying distribution. The purpose of *Table 3* is to highlight the difference in performance compared to model's in *Table 2*. It clearly shows the true positives decreasing and the false positives increasing. True positives went from $144{,}417 + 59{,}361\,(203{,}778)$ to

122,642 + 47,480 (170,122). False positives went from 54,534 + 64,428 (118,962) to 76,309 + 76,309 (152,618).

Table 3: Random Guessing Aggregate Confusion Matrix

Random Confusion Matrix		**Random Guess**		Marginal Sum of Actual Values
		Non-surprise	Surprise	
Actual	Non-surprise	**122,642**	76,309	198951
	Surprise	76,309	**47,480**	123789
Marginal Sum of Predictions		198951	123789	322740

This table is formed by 'randomly choosing the observations' by the allocation of observations according to the underlying test distribution, as presented by Sample Proportion in *Table 2*. A random choice benchmark is the most appropriate in this scenario, as the general theory is that models are not able to beat analysts in the estimation process (Brown, 1987).

The original confusion matrix can further be expanded so that we zoom into the type of surprise. In *Table 4*, I will deconstruct the results of *Table 2* into categories and directions of surprises. The next three tables look at, neutral, i.e., non-surprises, negative and positive surprises. I again produce a breakdown of random guessing. By comparing the first two tables, it can be seen that even without controlling for the recall, as identified by the marginal sum of predictions, the model outperforms random guessing in each category. In a later table, I show more definitely how the model outperforms the benchmark.

By using *Table 4* and *Table 5*, we can get a good indication of what is meant by naïve accuracy and precision benchmarks. A rational, uninformed person, trying to establish the best accuracy measure, would consistently predict the most frequent occurring class. In this scenario, they would consistently predict the neutral class and achieve an overall accuracy of 52% (Marginal Sum Neutral/All Observations, i.e., 169/322), close to the average "Bench" accuracy (5) of 48% in *Table 1*[10]. The problem with the benchmark model is that the precision would be very low at 52% and recall high at 100%. This shows the importance of deconstructing the accuracy measure to identify the precision across classes. If a person was to create a benchmark to enhance precision for all classes, the best strategy would be to use the distribution of the *training* data when allocating observations. In this study, I have created a more robust precision benchmark, in that I allow the benchmark to 'peek' at the underlying *test* distribution. Therefore, a person with this knowledge will randomly allocate 169500

[10] The "Bench" accuracy is slightly different because it was equal weighted across three groups and not value weighted as is the case for the confusion matrices.

observations to Neutral, 46821 observations to Negative and 106419 to Positive. This is indeed what *Table 5* illustrates; this can be appreciated by appreciating that the actual marginal sum of values matches the marginal sum of predictions. The reported accuracy of this multi-class prediction is the sum of the bolded figures divided by the total sum of observations. For *Table 4* this is 63% and for *Table 5* it is 41%.

Table 4: Surprise Breakdown Confusion Matrix

Confusion Matrix		**Predicted**			Marginal Sum of Actual Values
		Neutral	Negative	Positive	
Actual	Neutral	**144417**	4134	20949	169500
	Negative	16860	**7878**	22083	46821
	Positive	47568	7368	**51483**	106419
Marginal Sum of Predictions		208845	19380	94515	322740

This table expands on *Table 2* by splitting the prediction in three distinct groups. It is clear that the model produces many more positive than negative surprise predictions. To see whether the low number of predictions is warranted, we can look at the recall score in *Table 6*.

Table 5: Surprise Breakdown Random Guessing Confusion Matrix

Random Confusion Matrix		**Random Guessing**			Marginal Sum of Actual Values
		Neutral	Negative	Positive	
Actual	Neutral	**89020**	24590	55890	169500
	Negative	24590	**6792**	15439	46821
	Positive	55890	15439	**35090**	106419
Marginal Sum of Predictions		169500	46821	106419	322740

This table is formed by "randomly choosing the observations" by allocating the observations according to the underlying distribution. A random choice benchmark is the most appropriate in this scenario, as the general theory is that models are not able to beat analysts in the estimation process (Brown, 1987).

By looking at the proportions and precision in *Table 6* below, we can gain a better understanding of the model's performance results. All classes provide far superior precision scores than the sample proportions, i.e., random selection. The Negative surprise class' precision experienced the greatest improvement over and above random selection; an improvement of 26 percentage points, followed in order by the Positive and Negative class, with 21 and 12 percentage points respectively. Further, by looking at *Table 4* to *Table 6* together, we gain a better overall presentation of the model's performance. Of an aggregated total of 322,740 tested observations, 153,240 (48%) are surprise instances (observations). The model correctly recalled 59,361 (39%) of all the surprise instances. The model predicted

94,515 instances of positive surprises, 51,483 of which were correct, 43,032 of which were wrongly classified as either neutral or negative surprise, giving a precision score of 54%. This is far better than a random choice of 33%. The model predicted 19,380 instances of negative surprises, 7,878 were correctly predicted, and 11,502 were wrongly classified; this gives a precision score of 41%, which is once more much better than the random choice of 15%.

Table 6: Surprise Breakdown Percentage Composition, Proportions, Recall and Precision Measures

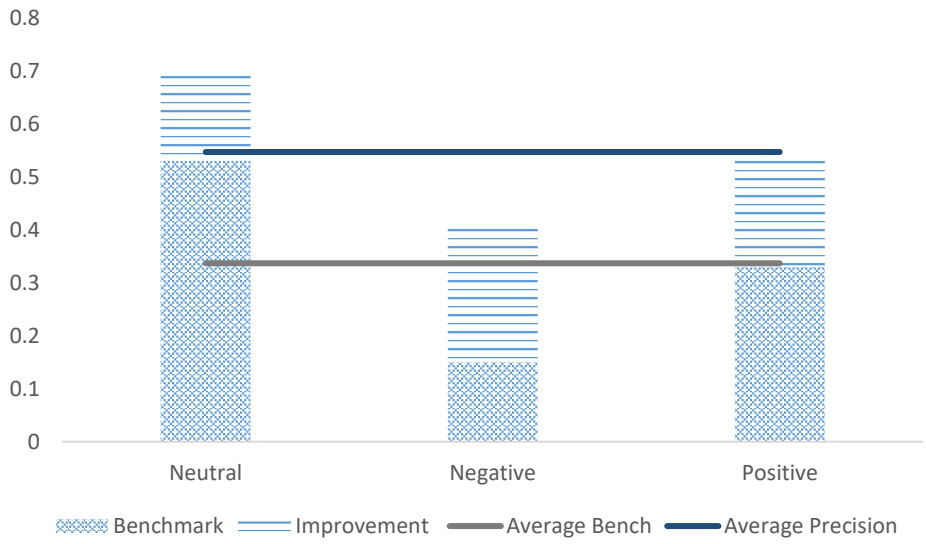
Percentage Composition Confusion Matrix		**Predicted**			Sample Distribution	Recall
		Neutral	Negative	Positive		
Actual	Neutral	**0.45**	0.01	0.06	0.53	0.85
	Negative	0.05	**0.02**	0.07	0.15	0.17
	Positive	0.15	0.02	**0.16**	0.33	0.48
Predicted Proportions		0.65	0.06	0.29		
Precision		0.69	0.41	0.54		
+ive to -ive Outcome Ratio		N/A	1.07	2.33		
Improvement		0.17	0.26	0.21		
Average Accuracy			0.63			

This table reports the same information as is reported in *Table 4* just in another way and with some new calculations over the results. In this table, we report the proportions instead of the number of observations. If you look at the correctly predicted negative observations, they are 2% of the overall proportion, but 15% of the population, so it is clear the model sacrifices recall for precision. This recall score can be improved by changing the decision threshold at the expense of precision. The Recall reports the proportion of the predicted category observations to the underlying sample of that category. The Precision reports the proportion of the predictions that are correct. To use an example above for the Negative Prediction, Recall = Correctly Predicted/Sample Proportion = 0.02/0.15 = 0.17. Precision = Correctly Predicted/Predicted Proportions = 0.02/0/07 = 0.41. More formally, Recall: TP/(TP+FN), Precision: TP/(TP+FP). The average is calculated by multiplying the precision scores with the underlying sample distribution. The average equal weighted percentage point improvement over all classes is 0.20 as can be seen in Figure 5.

This information can be further broken down by investigating the effect of differently sized surprise thresholds. *Table 7*, near the end of this section, shows, that on average, the higher the surprise thresholds are, the greater the improvement over random selection. For surprises 15%+, the average percentage point improvement is about 30%. This improvement is much more consistent for positive than negative surprises as showcased by a 0.03 as opposed to 0.12 confidence interval. The average improvement across the different surprise

thresholds (5%, 10%, 15%) is not that dissimilar from each other and falls between 0.14 - 0.16 with similar sized confidence intervals.

Figure 4: Precision Score Figure Accompanying *Table 6*



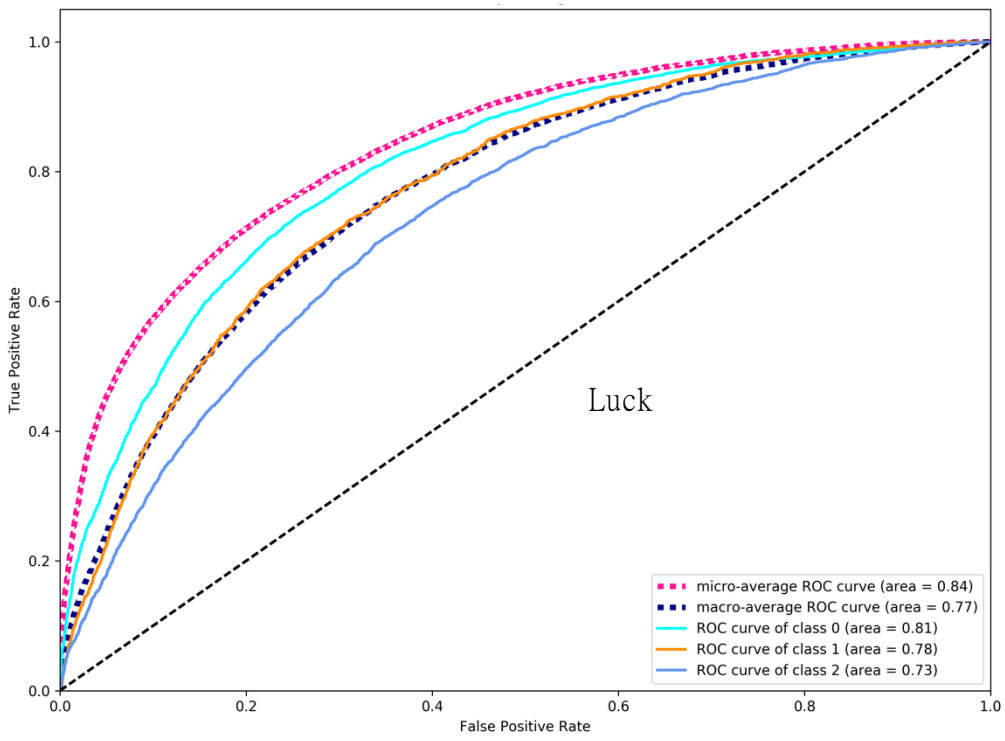
This figure presents the precision scores, i.e., the accuracy of correctly predicting neutral as neutral, negative as negative and, positive as positive, in an easy format. The lines above the benchmark are the improvement the model makes over and above the benchmark precision. Finally, the two average precision scores represent the average precision score vs. the average benchmark score. It is useful to note that the average precision score is different compared to the average. The simple average weights the different classes according to their underlying sample distribution, whereas the average precision presented here equal weights the classes.

The best model for predicting surprises is the 15%+ threshold, likely due to the patterns leading up to the announcement being more pronounced than that of lower thresholds. The higher surprise thresholds performance is slightly being offset by a decrease in the improvement for predicting the ever-increasing neutral class. If we ignore the neutral class, i.e., isolate surprises, we can clearly see an increase in the overall surprise prediction performance. This is an interesting finding as it shows that the model is picking up distinct and definitive patterns. The growing improvement also makes it much easier to create trading strategies for bigger deviating surprises.

The multiclass ROC is a universal way to identify the performance of a classification model. The AUC (area under curve) score provides an integral-based performance measure of the quality of the classifier. It is arguably the best single-number metric that machine learning researchers have to measure the performance of a classifier. The middle line is a line of random ordering. Therefore, the tighter the ROC-curves fit to the left corner of the plot, the better the performance. Two other measures included in the graph are a macro-average

measure that gives equal weight to each class (category) and a micro-average measure that looks at each observation. AUC values of 0.70+ are generally expected to show strong predictive effects. The ROC test adds additional evidence in favour of the model's performance being substantially different from null. In subsequent tables, *Table 7* and *Table 8*, I will also test the statistical significance of this outperformance.

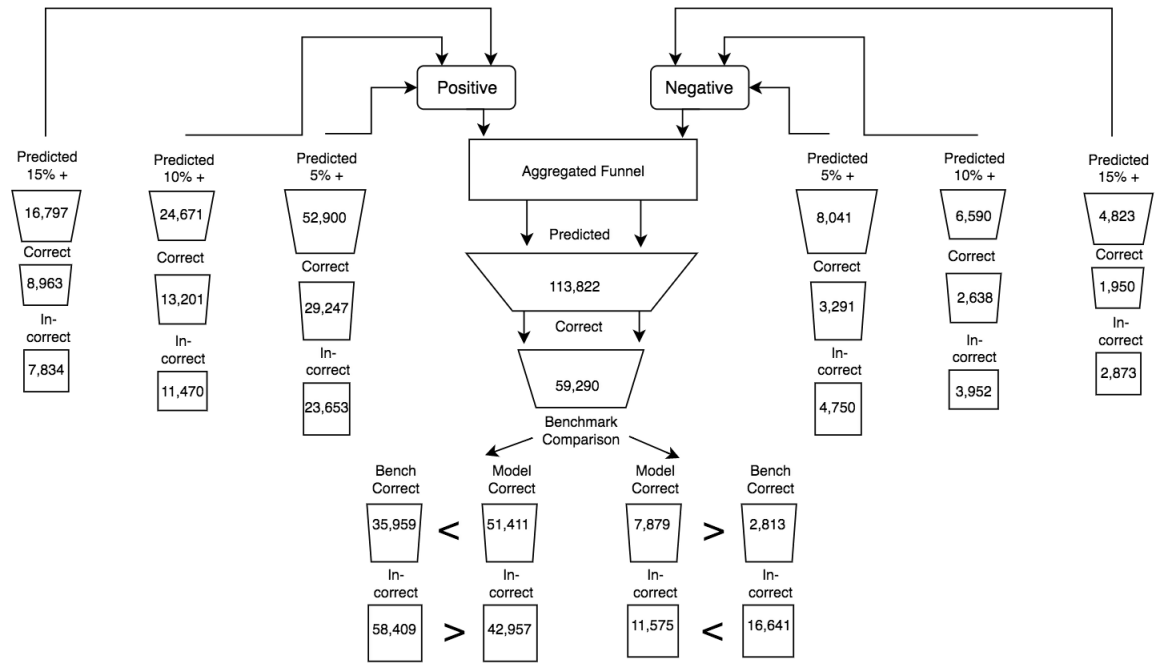
Figure 5: Multiclass Receiver Operating Characteristic (ROC) for a 15% Surprises Strategy



This figure reports the ROC and the associated area under the curve (AUC). The ROC (AUC) is measured for three different classes: class 0 is the negative surprise class, 1 is the neutral class, and 2 is the positive surprise class. The macro-average measure is the equal weighted AUC for all classes and the micro-average measure looks at each class's observation weight. The random ordering or luck line is plotted diagonally through the chart.

Figure 6 is the visual representation of the results presented in *Table 7*. This figure singles out positive and negative surprises and helps to establish an overall outlook of surprises by aggregating all results in a funnel. Random choice benchmark models are also included so that we can compare the significance of the results. This provides a further robustness test to see if, in aggregate, both types of surprises outperform the benchmark.

Figure 6: Model Surprise Prediction Funnel



This figure reports the quantitative performance of the aggregate performance of three different classifier. The left side is the positive surprises and respective thresholds and the right side is the negative surprises. This figure is slightly more involved than the confusion matrix, but it comes down to the same principle. The direction of this figure is top-to-bottom. The benefit of this prediction funnel is that it brings the predictions into an aggregate funnel to be compared against a random choice benchmark. This figure visualises the results of *Table 7*.

Table 8 finally provides interesting results of the performance of the model over different time intervals and with different levels of training data. The table identifies two important generalisations of machine learning prediction in time-series. The first is that with the inclusion of more data, the model tends to perform better (Domingos, 2012). Nevertheless, this does not always hold true for time-series data because of the potential of differences in distributions over time. Something as simple as seasonal trends can lead to bad predictions. There is, of course, ways to mitigate this, such as incorporating seasonal indicators as variables in the model. We can also difference away seasonality, but longer-term changes in analyst behaviour, such as shifts in the median analyst forecast over longer periods would remain an issue for prediction success (Brown, 2001). In machine learning, it is desirable that the distribution of the train and test data is the same, but this is not always possible, especially with financial data (Montas, Quevedo, Prieto, & Menndez, 2002). The expectation is that the results will improve if the distribution remains unchanged. This is especially true for the recall metric; as an example, we can look at the low recall rate of

negative surprises in *Table 6,* where the training set had many more "negative" surprises than the testing sets on average, hence the lower recall rate.

In *Table 8*, it can be seen that the second time-split performs worse than the first time-split, even though it has more data in its training set. This could be because of the characteristics of a particularly noisy training period, 1990-2003, that incorporates the tech bubble, which might have caused some systematic changes across the variables and change how they relate to surprises, leading to worse predictions in the future period. Before the inclusion of the period 1998-2003 in the training set, the model performed much better compared to the benchmark. A further explanation for this drop in improvement is that time-split two applies its learnings to a test set for the years 2004-2008, which in itself was a particularly noisy period containing the housing bubble and the start of the GFC (global financial crisis). The model performs better in the next two periods where both the tech bubble and the GFC are incorporated into the training set, and where the training set increased from 53,796 to 80,694 observations, further helping to improve the prediction outcomes.

Overall the predictions show that the model beats the benchmark for all the periods tested but to different extents; this is ascertained in two ways, first with ROC curves and then with statistical tests. Furthermore, it seems that the portion of correctly classified observations remains relatively stable. When looking at the inclusion of all time periods (the *All* row, last column), we yet again note only a small amount of improvement discrepancy across surprise thresholds. The results of the sub-analysis in table *Table 7* Panel B indicate that if we isolate the surprises from the neutral firms, they significantly outperform the benchmark precision scores.

In summary, splitting training and test sets by time intervals and checking for parameter stability over time is a very useful exercise in building a robust model and understanding how the model learns and predicts. In this part of the study, I have shown that a classification model can predict a surprise with much better precision than a naïve benchmark. I further revealed that this outperformance holds for various surprise thresholds and holds true over multiple periods, making the results robust. Next, I dig deeper into the significant predictors/variables driving the predictions and utilise the results to construct a profitable trading strategy.

Table 7: Class Surprises Count Statistics

Class	Surprise Deviation	Number of Predictions	(1) Model		(2) Random Guessing		(3) Improvement		Student's t-test
Panel A: Class Precision Analysis			Count	Precision	Count	Precision	Count	p.p.	
Positive	5% +	52900	29247	0.55	24030	0.45	5217	0.10	13.14
	10% +	24671	13201	0.54	7861	0.32	5340	0.22	37.49
	15% +	16797	8963	0.53	4068	0.24	4895	0.29	24.04
		94368	51411	0.54	35959	0.34	15452	0.20	24.89
Negative	5% +	8041	3291	0.41	1412	0.18	1879	0.23	4.65
	10% +	6590	2638	0.40	879	0.13	1759	0.27	5.78
	15% +	4823	1950	0.40	522	0.11	1428	0.30	6.57
		19454	7879	0.41	2813	0.14	5066	0.27	5.67
Neutral	5% +	47055	26782	0.57	18066	0.38	8716	0.19	29.17
	10% +	76132	52280	0.69	42197	0.55	10083	0.13	6.75
	15% +	85694	65173	0.76	56095	0.65	9078	0.11	5.40
		208881	144235	0.69	116358	0.53	27877	0.14	13.77
Sum/Avg (4)		322776	203525	0.63	155130	0.34	68913	0.20	14.78
Panel B: All Classes and Surprises Precision Analysis									
All Classes	5% +	107996	59320	0.55	43508	0.40	15812	0.15	15.65
	10% +	107393	68119	0.63	50937	0.47	17182	0.16	16.67
	15% +	107314	76086	0.71	60685	0.57	15401	0.14	12.00
Surprise Classes	5% +	60941	32538	0.53	25442	0.42	7096	0.33	15.28
	10% +	31261	15839	0.51	8740	0.28	7099	0.48	16.91
	15% +	21620	10913	0.50	4590	0.21	6323	0.59	6.26

(1) represents the correctly predicted count and precision of the model; precision is the percentage of correctly predicted observations for each test, while count is the number of successful predictions. (2) produces the results for random choice. (3) the improvement is presented in count and in percentage point improvement, i.e., (1) model precision minus (2) random precision. This improvement has been tested over an unequal variance paired t-test from which confidence intervals were calculated and are reported. I also produce sub-

calculations for each category separately, (4) as well as sub-calculations for all models combined, found beside 'Sum/Avg'. Panel B, All Classes, expands on (4) by deconstructing it into the thresholds (surprise deviation), giving us an indication of each threshold's performance. Since one purpose of this study is to create trading strategies, it is useful to know what the surprise performance is over and above random choice; the Surprise Classes provides the necessary precision and improvement scores of surprises. It is clear that the model outperforms random choice not only at lower surprise thresholds, but also it increasingly outperforms random choice at higher thresholds.

Table 8: Test Intervals Surprises Count Statistics

	Train Size	Train Dates	Test Dates	Train Size	Test Size	Surprise Deviation	(1) Model		(2) Random Guessing		(3) Improvement		t-test
							Count	Precision	Count	Precision	Count	p.p.	
1	20%	90-97	98-03	26898	26898	5% +	15249	0.57	10411	0.39	4838	0.18	5.26
						10% +	17866	0.66	13527	0.50	4339	0.16	4.75
						15% +	19752	0.73	15904	0.59	3848	0.14	2.64
							52867	0.66	39842	0.49	13025	0.16	4.21
2	40%	90-03	04-08	53796	26898	5% +	14447	0.54	10854	0.40	3593	0.13	3.90
						10% +	16785	0.62	13307	0.49	3478	0.13	3.81
						15% +	19012	0.71	16034	0.60	2978	0.11	2.04
							50244	0.62	40195	0.50	10049	0.12	3.25
3	60%	90-08	09-12	80694	26898	5% +	14712	0.55	11500	0.43	3212	0.12	3.49
						10% +	15840	0.59	11176	0.42	4664	0.17	5.10
						15% +	17768	0.66	13191	0.49	4577	0.17	3.14
							48320	0.60	35867	0.44	12453	0.15	3.91
4	80%	90-12	13-16	107592	26898	5% +	14912	0.55	10743	0.40	4169	0.15	4.53
						10% +	17628	0.66	12927	0.48	4701	0.17	5.15
						15% +	19554	0.73	15556	0.58	3998	0.15	2.74
							52094	0.65	39226	0.49	12868	0.16	4.14

Continued on next page

Table 10 - Continued from previous page

Total	22	18	806940	322776		203525	0.63	155130	0.48	48395	0.15	3.88
All	22	18	268980	107592	5% +	59320	0.55	43508	0.40	15812	0.15	4.30
					10% +	68119	0.63	50937	0.47	17182	0.16	4.70
					15% +	76086	0.71	60685	0.56	15401	0.14	2.64

Precision is the percentage of correctly predicted observations for each test; count is the number of successful predictions. This table identifies the correctly (1) classified surprise categories over different test periods. The table also provides for some aggregate measures over all the time-splits; these measures are found beside *Total* and *All*. *Total* is the average and or summation across all tests, while *All* still leaves the percentage surprise buckets intact. In machine learning, these measures are commonly referred to as validated metrics. (3) the improvement is presented in count and in percentage point improvements, i.e., (1) precision - (2) precision.

B. Prediction Analysis

A large part of this study is dedicated to the creation of an improved prediction model. Finance researchers are often interested in understanding the predictors. Machine learning does not make this an easy task. In this section of the study, the focus is on gaining a better grasp on the predictors to the classification model. Here the machine learning model helps us to determine biases by identifying variables that are important in predicting surprises over and above the analysts' forecast. There are 19 different varieties of technical indicators used in this study, with three different parameter timeframes, resulting in 57 technical indicators overall. The indicators are split into 1 - 4, 5 - 9, and 10 - 20-day lookback periods. After mapping signal processing and other algorithms over the indicators and doing the first and second phase of variable selection on validation sets, the process generated 677 relevant variables[11]. Of the original 57 technical indicators, 55 remained relevant. Of the 69 earnings-related variables, 62 were relevant. The overall number of relevant variables amounted to 794.

Table 9 shows a list of the five most important variables to identify surprises. This section specifically focuses on the most telling variables for predicting large positive surprises. The average score is collectively calculated from three tree models, XGBoost, AdaBoost, and Random Forest, to uncover an intersection of important variables for all tested surprise thresholds and test-train splits. This lessens the likelihood of the variables appearing by chance so that the selected variables are agnostic to the type of tree model. This process is known as tree-based variable selection. These variables have been normalised across the models, and the intersection of all variables revealed the most important subgroup.

Multicollinearity has a big influence on the importance measure. The variable importance measure used in this study is called Gini Importance. This measure is based on the number of splits each variable undergoes, weighted by the squared improvement that results from each split that gets averaged over all trees (Elith, Leathwick, & Hastie, 2008). In layman's terms, the more an attribute is used to make key decisions with decision trees, the higher its relative importance. This measure can be thought of as a 'significance' score for decision tree models. The reality is that a single variable's importance is reduced if there are a multitude of variables of similar character and correlation to the response variable. In this study, there are 732 pricing variables and 62 earnings variables. Thus, the importance of the

[11] Relevancy means that the variable has a non-zero effect on the predicted outcome.

pricing variables is widely distributed among relatively homogenous variables. Therefore, it is more useful to look at the cumulative importance of the pricing variable, which further results show to be around 20.9%. The cumulative performance of each group of variables is reported at the bottom of *Table 9* and *Table 10.* In simple terms, more than 20% of the decisions get made as a consequence of using price-related variables.[12]

The issue with many machine learning models is that their nonlinearity makes it hard to enforce monotonicity constraints to identify in which direction the relationship is between the independent variable and the machine-learned response function. In ML, the response can change in a positive or negative direction and at varying rates for changes in an independent variable, making the interpretation of variable importance much harder than simply looking at the coefficients of a linear model. Although the singular importance value of a variable can be very helpful for understanding the average direction and size of a variable to the response, it does not explain the potential nonlinear relationship of the variable with the response. Information about this relationship are of great interest to researchers and industry experts alike. To identify the relationship, we can make use of a technique called partial dependence.

Partial Dependence is a means of identifying the marginal dependence between the predictors and the outcome (Hastie, Tibshirani, & Friedman, 2009). The basic premise of this technique is to obtain a prediction for all unique values of a variable while accounting for the effects of all the other variables. Breaking this development process down, for every unique value of the variable of interest, a new dataset is created where all the observations of the variable is set equal to that unique value and all other variables are left unchanged; the new dataset then gets ingested in a decision tree where all the prediction are averaged and plotted. This process is repeated for all the values of the variable of interest to get a range of outputs for inputs, and similarly, this can be repeated for variable value pairs. This approach, as a result of incorporating all the information from other variables, has the ability to detect nonlinear relationships without the need to pre-specify them, and it allows us to visualise the relationship between an input and a response variable. See Appendix, *Method A 4* for an expanded explanation and the mathematical formulae driving this concept.

The nonlinear nature of these relationships can be seen in *Table 9*, where the direction can change signs as the variable value increases. *D* represents the direction in which an increase in output would drive output. For a classification task such as in *Table 9*, a higher

[12] Apart from the correlated variable problem, the Gini Importance measure is also biased towards variables with more categories. This variable importance measure can be inconsistent for tree ensembles as higher importance can be assigned to variables with a lower impact on the model's output.

output means a higher likelihood of a positive surprise. For both *Table 9* and *Table 10*, 1 means a 100% chance of a positive surprise, and 0, a 0% chance of a positive surprise. A better visualisation of the output value and the relationship of these values can be found in *Figure A14* for earnings-related variables and *Figure A15* for price related variables. Partial dependence between two independent variables and the outcome variable can also be interesting, especially for visualising more complex relationships. The most important of these combined interactions have been included in a graph after the analyses of each set of variables. Moreover, all the important variables have partial dependence plots, see *Figure A14* in the Appendix.

Table 9: Earnings Related Variable Importance and Response Direction for Classification

Name	**Short Description**	**Score**	**D**
est_avg_t	This time period's analyst EPS forecast	0.247	-
$diff_{-1}$	The difference between the past actual EPS, p_{-1} and p_{-2}	0.119	+/-/+
p_{-1}	Actual EPS t_{-1}	0.082	-
$d\_e\_diff_{-4}$	Difference between the past actual, p_{-4} and forecast est_avg_{t-4}	0.073	+
$diff_{-4}$	The difference between actual EPS p_{-4} and actual p_{-1}	0.060	+/-/+
Other	57 other earnings-related variables.	0.212	
Total		**0.794**	

This table identifies the most important earnings-related variables to predict a positive earnings surprise. The Gini importance, which is the average gain in information, is used to represent the variable importance (Score). *D* identifies the direction of the variable as identified by the partial dependence graph. See *Figure A14*, the graphs from which the directions are identified.

The most important earnings-related variable is the analyst consensus forecast itself, est_avg_t; this is expected because the purpose of the model is to identify deviations from this value and the true value to identify surprises. It provides a measure of reference to the other variables of the model. Countless papers have identified the performance of analysts' forecasts in forecasting earnings as recapped by Brown (1987). The lower the forecasted EPS, the more likely a surprise is to occur all else equal; 32% of the outcome is attributable to this value.

The second most important variable is the difference between the actual level of earnings between period t_{-1} and t_{-2}, called $diff_{-1}$ and the fifth most important variable, is the difference in earnings between t_{-1} and t_{-4}, called $diff_{-4}$. These are novel variables not

yet identified by past research. The extent of past increases in earnings are therefore an important variable for predicting future surprises. If the value is very high, surprises become more likely. However, surprises are also more likely if the value gets very low. In the middle range surprises become less likely. The measure is u-shaped, which is indicative of a sort of variance measure.

The next important value is the actual earnings at time t_{-1}, called p_{-1}. Research by Bradshaw, Drake, Myers, and Myers (2012), has shown that the past annual earnings often outperform not just mechanical time-series models but also analysts' forecasts. Past quarterly earnings seem to be an important value in predicting the next quarter's earnings surprise. The relationship between past earnings (p_{-1}) and the current analyst estimates (est_avg_t) shows that where p_{-1} is very large and est_avg_t is very low, then a positive surprise is likely to occur more than 90% of the time, all else being equal. Further, where p_{-1} is low and est_avg_t is high, a surprise is unlikely to occur.

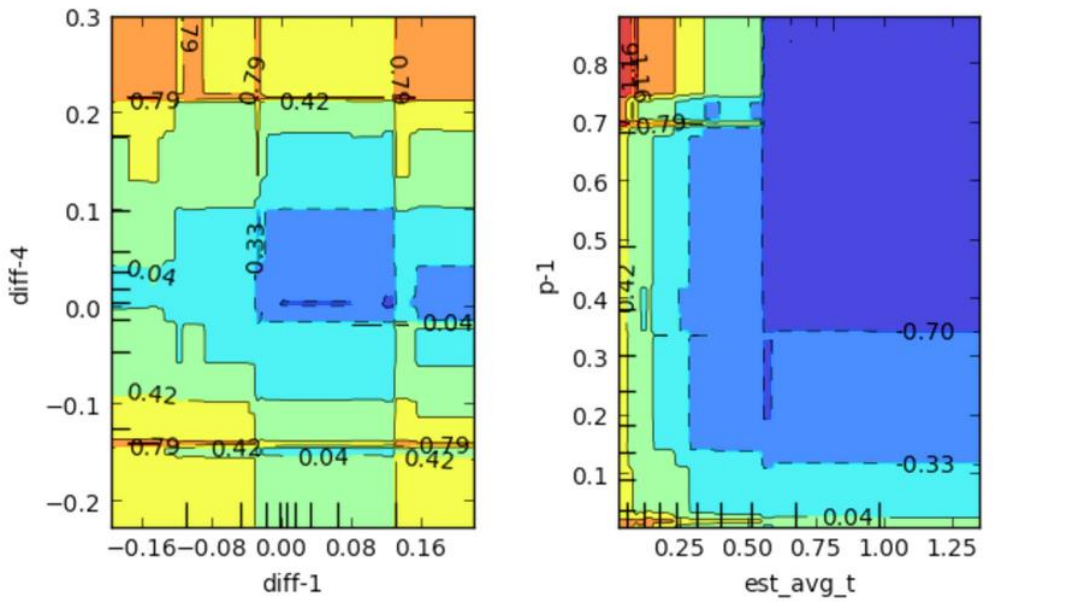
The fourth most important variable is the difference between past and forecasted earnings four quarters ago, i.e., one year between the forecast, est_avg_{t-4} and the actual value, p_{-4}. The importance of this variable was also expected since Easterwood and Nutt (1999) and Fried and Givoly (1982) separately showed that past errors tend to persist. The larger the difference, the higher the likelihood of surprise. Other variables that showed an above 2% importance include rolling averages and weighted rolling averages of the difference between past earnings and analyst forecasts, and the standard deviation of analyst forecasts.[13]

It is not always the best practice to look at the isolated effect of an input variable on the response variable; for that reason I have included the dependence plots in *Figure 7* between two input variables and the output. Out of the above list, there are more than 32 (2^5) ways to conjure up directional relationships. To conceptually understand the web of relationship, due to the nature of nonlinear relationships, it is better to describe a simple tree explanation of the above variables and relationships for a combination of variables that would lead to a perfect surprise prediction. The following explains what would lead to a close-to-

[13] More specifically, $d_{e_{p-1}}$, the difference between p_{-1} and est_avg_{t-1}. And d_e_16, the rolling mean of actual EPS, p from $t_{-16}$to t_{-1}, minus the forecast rolling mean from est_avg_t from $t_{-16}$to t_{-1}. The standard deviation of all individual forecasts, est_std. The higher the standard deviations, est_std, the larger the uncertainty among analysts. This is associated with a decrease to the machine-predicted EPS. The reason is that the base, analyst forecasts, est_avg_t, tend to be excessively positive in periods of uncertainty as has been reported by Das et al. (1998).

perfect surprise prediction. If the current analysts' forecast is low while the past earnings are high, there is a higher likelihood of a positive surprise this period; this is likely due to analysts being conservative. If in the past it has been shown that analysts are conservative and that surprises transpired, i.e., $d\_e\_diff_{-4}$, then the likelihood of surprises increases even more. When there is a large difference in earnings between the last two periods EPS, p_{-1} and p_{-2}, i.e., $diff_{-1}$, the likelihood of surprise increases. The same holds for $diff_{-4}$. Overall, these variables accounted for around 80% of the variable importance. Referring back to *Table 1*, it has also been shown, using an inductive method to theory testing, that these variables accounted for more than half of the total improvement over the benchmark. The remaining importance is distributed between pricing and price-derived variables.

Figure 7: Partial Dependence of Class Probabilities on Earnings Related Feature Combinations for Classification



The figures indicate the probability that an earnings surprise will occur, all else being equal. The dashed lines identify the space where earnings surprises are less likely to occur. The small ticks on the axes are an indication of the underlying distribution. As the colours get warmer, surprises are more likely to occur, the colder the colour the less likely a surprise is to occur. These graphs show the partial dependence relationships between two variables. On the *left*, it can be seen that a surprise is more likely if both $diff_{-1}$ and $diff_{-4}$ are large and that surprises are less likely when both of these values are around the mean. This would indicate that there is a predictable trend. Another interesting observation is that if the longer trend $diff_{-4}$ is large and shorter-term earnings decrease, $diff_{-1}$ is negative, then the small blip is short-lived and likely to be corrected in the next period, as can be seen with the high likelihood of surprise in this area, >79%, i.e., the top left corner. On the *right*, as previously noted, when p_{-1} is large and est_avg_t is low, a surprise is likely, >90%, and where p_{-1} is low and est_avg_t high a surprise is unlikely.

The variables in *Table 10* identify the shapes and patterns associated with the market before the firm makes its official earnings announcement. These variables can potentially

reflect signals of management's position of the forthcoming announcements, or it can simply reflect traders' use of privileged or public information. This information can be firm-related or more broadly economic in nature; either way, these variables reflect information not contained by the smaller scope of earnings-related variables. Many of the mapping transformations identify certain characteristics and shapes associated with technical indicators. One example is a mapper that identifies how often the price crosses the top Bollinger Band in the days leading up to the announcement. A second example is a measure of the serial sum of absolute changes for a Relative Strength Index.

In essence, a mapper is a transformation that helps to identify patterns in time-series data. Incorporating the signal-processed algorithms over the traditional technical indicators provides a significant improvement to the overall variable importance of the pricing variables, as evidenced by the below table, with a total variable importance of 20.9%. This has further been proven in the classification subsection, *Table 1*, where an inductive method to theory testing revealed that these variables account for more than one-third of the total improvement over a naive benchmark. A possible downfall is that the variables presented here are not as easily interpretable as the earnings-related variables in the preceding table.

Table 10: Pricing Related Variable Importance and Response Direction for Classification

Base Feature	Mapper	Parameters	Score	D
$Chaikin$	Mean abs change quantiles	qh:1.0, ql:0.4	0.036	+
$Donchian_{10}$	Friedrich Coefficients	m:3, r:30, coeff: 2	0.026	-/+/-
MA_3	Max Langevin Fixed Point	m:3, r:30	0.013	-
$Trix_{10}$	Autocorrelation	lag:8	0.009	-
$KelChU_5$	Max Langevin Fixed Point	m:3, r:30	0.007	-
Other (727)			0.118	
Total			**0.209**	

The above variables have been selected as the 5 most important variables out of all standard pricing, technical and signal processed variables. For many of the variables, zero seems to be an important boundary in deciding in what direction the output 'moves.' See *Figure A15* for the graphs from which the directions are identified. The mapper is a transformation that helps to identify patterns in time-series data. It transforms a series of data into a single number to be fed into the machine learning model. Parameters are the auto-selected parameters that the mapper identified to be most informative in explaining the response function. A mapper has on average 11 different parameter iterations it tests against the output on a validation set.

Looking at *Table 10*, the most important pricing variable is a transformation mapped over a Chaikin technical indicator. The Chaikin Oscillator is an indicator of an indicator, the

latter of which is derived from the stock price. The Chaikin indicator is a third-derivative indicator designed to anticipate directional changes in the Accumulation Distribution Line, a volume-based indicator, by measuring the momentum behind the movements. The purpose of the Chaikin indicator is to predict directional changes in a price trend. The mapper entails a calculation of the average distance between each consecutive value in time-series within a high (1) and low quantile (0.4) range for the $Chaikin$ measure. It, therefore, provides an absolute measure of the amount of variance the Chaikin Oscillator experienced for each subsequent value between a skewed high and low quantile, for the 30 days before the announcement. If the absolute difference is large, a surprise is more likely to occur, ceteris paribus.

The Donchian 10 technical indicator is formed by taking the highest daily high and the lowest daily low of the last k periods and computing the difference. It is a measure of volatility on the one hand but can also provide signals for long and short positions (Patel, 1980). The Friedrich coefficient mapper allows us to fit and calculate the coefficients of a polynomial equation, $h(x)$, that is fitted to the deterministic dynamics of the Langevin Model (Siegert, Friedrich, & Peinke, 1998). Langevin dynamics is essentially an approach to mathematically model the dynamics of molecular systems (Langevin, 1908). In simple terms, it is an equation that describes Brownian motion. This specific extension of the formula has been used in finance-related papers such as those describing the dynamics of market crashes and to quantify random fluctuation in the foreign exchange markets (Bouchaud & Cont, 1998; Friedrich, Peinke, & Renner, 2000). The particular parameter of interest is the Potential coefficient, which is affected by the amount of time the value does not deviate from a harmonic-like motion. If this amount is low, then surprises become less likely; if this value is in the middle ranges, then surprises are much more likely; and lastly, if this value is very high, a surprise is less likely.

MA 3 is a simple moving average indicator that tracks the last three days of pricing history. We will yet again make use of the Langevin model, this time to find the maximum fixed point for a third-degree polynomial fitted to the deterministic dynamics of the Langevin model. In a sense, we are trying to find the maximum fixed point of a rolling momentum average, modelled in the form of Brownian motion. Fixed points are, fundamentally, concepts of stability and equilibria in economics. In physics, it is often a way to predict phase transition. Even though this figure is multidimensional and less interpretable, we can say that the larger the maximum equilibrium in a series of moving averages of a third-degree polynomial, the less likely an earnings surprise will occur.

The triple exponential average, TRIX 10, is a measure used to identify oversold or overbought securities (Hutson, 1983). The measure fluctuates around a zero line. A negative (positive) TRIX value signifies an oversold (overbought) market. This indicator looks at the trend for the last 10 days. The 7-day lag autocorrelation of a rolling time-series of this value over the last 30 days is shown to be an important variable. The higher the autocorrelation of TRIX 10, the less likely you are to experience a positive surprise.

Keltner Channels are trend-following indicators used to identify reversals with channel breakouts. The upper-band of a Keltner Channel indicator, looking at the last five days in combination with a largest fixed-point forecasted from a function which has been fitted to the deterministic dynamics of the Langevin model, shows up as an important variable. A possible interpretation is that the nature of the position and extent of the maximum fixed point of the polynomial is such that a higher value leads to a lower likelihood of surprise and a lower value to a higher likelihood of a surprise.

Machine learning algorithms scour the variable space for patterns to identify associative interactions between input and output values. The relationships and variable directions may completely change when additional variables are added. These associative patterns, such as seen in *Figure A15*, should not be mistaken for causation. They do, however, make for an interesting assessment, especially the variables relating to earnings, as they are easier to interpret; the price variables are slightly less interpretable and somewhat noisy.

C. Trading Strategy

The results reported until now are mutable, in that we can adjust the loss function and surprise thresholds to achieve results that would be better suited to a trading strategy.[14] Precision can be improved but at the expense of recall and overall accuracy; a single class' metrics can be improved at the expense of other classes; even the class probability thresholds can be changed to favour one class over another. It is necessary therefore to identify what is important for a trading strategy and then to optimise for that measure. Every change to the loss function affects another part of the results. In trading, we are particularly focused on improving the precision; this preference can be expressed in the loss function or we can simply alter the sample of firms under observation.

Now that we have a model that can predict earnings surprises on paper with good success, it is necessary to see if it can translate into a trading strategy. It is always possible that the market will not react to what we consider surprises. Earnings surprises are often termed as soft-events in books on event-driven investment strategies. Event-driven investment forms a large part of the hedge fund industry, accounting for about 30% of all $2.9 trillion assets under management according to Hedge Fund Research 2015. Event-driven strategies are a great way to benefit from increased volatility. For the classification results achieved up to this point, it is possible that the consensus forecast is not defined in the same way that the market may perceive the consensus to be. The next section mostly alleviates this concern, as it is shown that the market does indeed react to what this paper defines as surprises. Furthermore, it is possible that the model simply learns on a lot of correlated and tail risk. It is, therefore, worth seeing whether a trading strategy using these surprises can earn long-term cumulative and excess returns.

An important question to ask when combining a classification task and a trading strategy is to know in what categories wrongly classified instances fall to identify the viability of a profitable strategy. For example, if firms experience a negative earnings surprise when a positive surprise is predicted, it can have a significant impact on a strategy's return. To investigate incorrect classification, we make use of a confusion matrix (often called contingency tables). *Table 6* presents a ratio called positive-to-negative outcome, which is an important figure for a potential trading strategy. The ratio can be deconstructed as follows: for positive surprises, it is the ratio of the number of positive surprises correctly

[14] This should only be done on the validation set otherwise it might lead to overfitting.

predicted to the number of negative surprises mistakenly predicted as positive surprises. The negative surprise ratio for example is the ratio of the number of negative surprises correctly predicted to the number of positive surprises mistakenly predicted as negative surprises.

For predicting positive surprises, this number is high at around 2.3, which is reasonably good and means that you would have one critical mistake for every 2.3 positive surprises you predict correctly. For negative surprises, this amount is around 1.1. This amount is too low to form a successful shorting strategy, and this has further been proven by a lacklustre performance of a potential shorting strategy as seen in *Figure 8*. This phenomena of mistakenly predicting both extreme classes are common in machine learning. Although the model does not intend to, it often classifies the surprises according to variables associated with a large variability in earnings, leading to both classes being flagged as potentially correct classifications. The specifications of the model can, however, be changed to penalise false positives more by simply adjusting the loss function thresholds. Doing this decreases the number of trading opportunities available for only a small added benefit. A further step one can take is to sort predictions by the difference in the probabilities of positive and negative surprises and then to only trade on the higher decile predictions. An even simpler approach would be to follow a long-only strategy to take advantage of the large positive-to-negative ratio it offers.

The following expresses a simple strategy by going long (short) on stocks that are expected to experience a positive (negative) surprise tomorrow (t), at closing today (t-1), and liquidating the stocks at closing tomorrow (t). The stocks are equally weighted to maintain well-diversified returns for the day, as there is, on average, only four firms in a portfolio of expected surprises, but there can be as few as one firm in a portfolio.[15] For each day, I form stocks into positive and negative surprise prediction portfolios for surprises that deviate from -50% to 50% in order to select the best performing threshold. The preferred threshold is selected based on tests done against a validation set. The results in the validation set show that the best trading strategies exist between 5%-20%, with 15% being the optimal trading strategy for positive surprises.[16] I, show that an event-driven trading strategy that takes long positions in stocks with 15% positive surprise predictions, while earning the market rate of return over non-earnings surprise days, produces an alpha of 8% relative to a five-factor asset-pricing model on and out-of-sample test set (Fama & French, 2015). There is no good

[15] For robustness, I have also tested for value-weighted returns, which showed a slight increase in improvement.
[16] See the trading strategy section on page 56 for an elaborate explanation of the methodology used.

reason to form a long-short portfolio with negative and positive earnings surprises as the constituent firms in the portfolio are not comparable in type and the positive and negative surprise predictions do not necessarily fall on the same dates.

The strategies developed in this section fully invest all capital in each event. It is therefore important to include some sort of loss minimisation strategy. As a result, one strategy incorporates a stop-loss for stocks that fell more than 10%; 10% is only the trigger, and a conservative loss of 20% is used to simulate slippage. This is done by comparing the opening with the low price of the day. An endless number of opportunities and strategies can be created; it is, therefore, important to select simple strategies not to undermine the robustness of this study. In saying that, the choice of slippage is not backed by any literature and is an arbitrary, albeit conservative, choice for the strategy.

Equation (4) is a simple return calculation for an earnings announcement of firm *i* at time *t*, where the daily low price, Sl_{it}, is not more than 10% lower than the closing price $S_{i,(t-1)}$. If it is, then a slippage loss of -20% is allocated to the return quarter for that firm. The stop-loss is only applied in one strategy in *Table 12*, all other results are reported without any risk mechanisms.

$$R_{it} = \frac{\left(S_{it} - S_{i,(t-1)}\right)}{S_{i,(t-1)}}, \qquad if \ \ \frac{Sl_{it} - S_{i,(t-1)}}{S_{i,(t-1)}} < -10\%, \ \ R_{it} = -20\% \tag{2}$$

In this equation, S is a set of all the firms in the sample, i is the firms that have been predicted to experience an earnings surprise based on preselected thresholds at time *t*.[17]

The equal weighted return of a portfolio of surprise firms is then calculated as such,

$$R_{pt} = \frac{1}{n}\sum_{i=1}^{n_{pt}} R_{it} \tag{3}$$

In this equation, i is all the firms that are predicted to experience a surprise on date t. Therefore, R_{it} is the return on the common stock of firm *i* on date *t* and n_{pt} is the number of

[17] $S_{i,(t-1)}$ is the closing price of the common stock of firm *i* on date *t-1*. Sl_{ti} is the daily low of the common stock price of firm *i* on date *t-1*.

firms in portfolio p at the close of the trading on date *t-1*. R_{Mt} is the value-weighted market return.

The equation below is the five-factor asset-pricing model. In this equation, R_{pt} is the return of portfolio p for period t. R_{Ft} is the risk-free rate. The alpha, a_i is the abnormal return of the trading strategy. R_{Mt} is the value-weighted return of the market portfolio. SMB_t, HML_t, RMW_t and CMA_t are the respective differences between diversified portfolios of small stocks and big stocks, high and low B/M stocks, robust and weak profitability stocks, and low and high investment stocks. To perform the regressions, the respective daily values were obtained from Kenneth French's website.[18]

$$R_{pt} - RF_t = a_i + b_i\,(R_{Mt} - R_{Ft}) + s_i SMB_t + h_i HML_t + r_i RMW_t + c_i CMA_t + e \qquad (4)$$

In *Table 11*, I report the results of three surprise strategies each with a different surprise threshold.[19] The 15% long strategy is the only of these three strategies that shows statistical significance for abnormal profits. The daily abnormal returns generated by the 15% long strategy is 0.037% before transaction costs. The economic viability of these strategies is represented by the cumulative return graph in *Figure 8*. These figures represent a more effective performance measure using Monte Carlo simulation. Instead of simply comparing the portfolio against the market, the performance of the strategy can be compared against a simulated average of the cumulative return from randomly picking from the sample of firms before earnings announcements. These sample firm quarters include surprises and neutral observations. The model shows its superiority at being able to pick out future surprises by beating this simulated average at its 99% significance bands, as identified by the blue channel.

$$R_{pt} = \frac{1}{n}\sum_{i=1}^{n_{pt}} R_{it}\,, \qquad where\ n = 0, R_{pt} = R_{Mt} \qquad (5)$$

[18] http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

[19] In total the parameter searched 100 different strategies, -50% to +50%, only six are reported, -15%, -10%, -5%, and +5%, +10%, +15%. 15%+ was the best performing strategy.

Table 11: Daily Abnormal Returns for Large Firms Trading Strategy

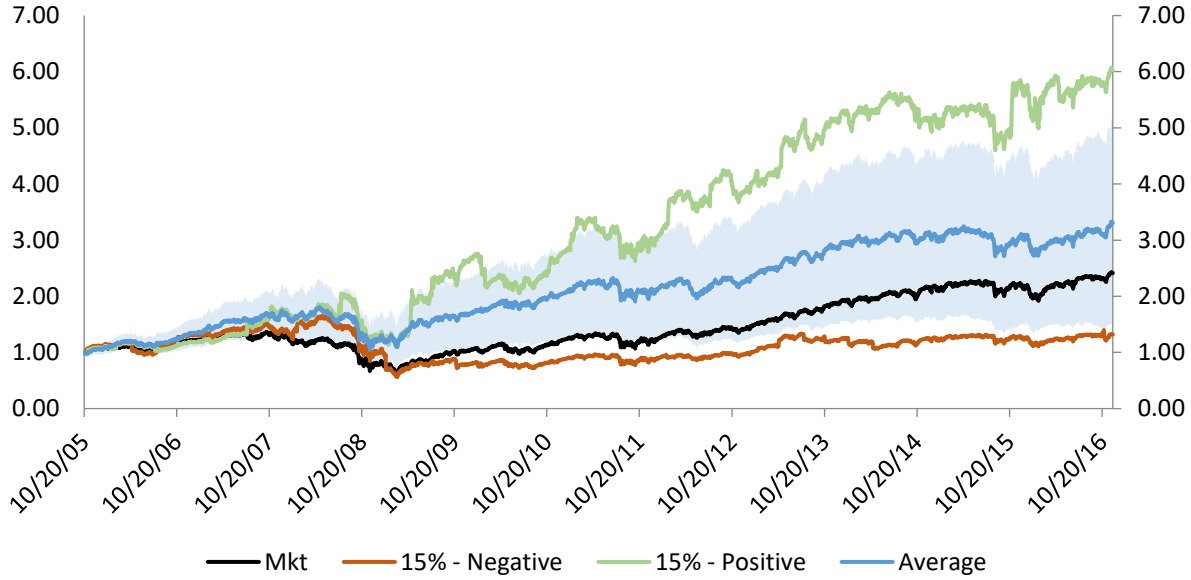
Threshold	Position	Firm Qtrs.	Surprise Days	Abnormal Returns
5%+	Long	5339	1909	0.04%
	Short	267	139	0.00%
10%+	Long	1953	863	0.03%
	Short	111	67	0.01%
15%+	Long	774	220	0.04%**
	Short	234	135	0.02%

Long identifies the positions taken in expected positive surprises portfolios; short is the short-selling of portfolios where negative surprises are expected. Firm quarters are the number of earnings quarters used across all surprise portfolios. Surprise days is the number of days on which surprises occurred and formed part of the portfolio. The table tracks the performance of a strategy over 2994 trading days. All strategies showed significance for the market coefficient. For all the long strategies, HML showed positive significance apart for the 5% long thresholds. For all the short strategies, SMB and CMA showed positive significance. See *Table A17* for a factor table that reports on the size of the positive and negative coefficients and their significance. The alphas with this strategy are somewhat low, mostly as a result of the returns fitting strongly on value firms; however, the plotted cumulative returns still show economic significance over the sample period. The cumulative returns are shown in the tables below. The same study has been performed on smaller firms, but this did not show any strong significance, likely as a result of investors being slower to react to smaller firm surprises and lower analyst coverage leading to inaccurate surprise calculations and stale forecasts. The significance values are calculated from Newey-West adjusted standard errors with 3-month lags. *10% significance **5% significance ***1% significance.

The daily abnormal return of the best performing strategy (15%+ threshold) is around 0.037% daily and 10% annualised. By adjusting for the effect of trading costs at a conservative 50 bps per round trip, both for the surprise and market portfolios, the daily abnormal return decreases to 0.029% and around 8% annualised. This result is especially good as it is driven by only 20 surprise portfolio days on average per year; each surprise portfolio day produces around 0.40% in abnormal return. The abnormal return over the sample period is directly related to the amount of successfully predicted surprises. I also look into the cumulative effects of raw portfolio returns and compare them against the market in a diagram. To calculate the cumulative return over the sample period, I compound the R_{pt} return over *m* days of the sample to yield the total cumulative return of R_{cum}. In the equation below, R_{pt} is the daily return of portfolio *p* on date *t*, and *m* is the number of sample trading days.

$$R_{cum} = \prod_{t=1}^{m} \left(1 + R_{pt}\right) - 1 \qquad (6)$$

Figure 8: Portfolio Value - Large Firms 15% Surprise Prediction Strategies



This portfolio reports the cumulative returns of buying and holding positive and negative surprise portfolios for all firms in the sample with a market value of $10 billion. On average, there are about four firms for each portfolio day. On days where no trading surprises occur, a position in the market is taken. The band in the middle is the significance band obtained from a Monte Carlo simulation by randomly taking a position in 774 firms before an earnings announcement. The chart also reports the cumulative portfolio return of the market as calculated from the market returns obtained from French's website. The chart shows that negative surprises mostly track the market portfolio. It is possible that some returns can be earned by shorting these surprises over certain periods, but on average it is not a very profitable strategy due to the small amount of shorting opportunities. In total, there are 2944 trading days, for the long strategy; 215 of these days are returns from earnings surprises comprising 774 firms, and for the short strategy 62 days comprising 234 firms.

In *Figure 8*, I go a step further than simply reporting the cumulative performance of a market portfolio. I also present another benchmark to simulate the trading performance of a random selection of stocks from the full sample of firm. I run 1000 Monte Carlo Simulations for each of the strategies by randomly selecting *n* number of firms to trade on before earnings surprises.[20] After this, I calculate the average simulated portfolio value and the 99% confidence bands. The simulation is reported in blue with the associated confidence bands. The positive surprise strategy is in green and the negative surprise strategy is in red. From the plot, it is clear that a simple buy-and-hold strategy on the full subsection of firms would also outperform a passive position in the market. This mostly has to do with the sample selection criteria of large firms that have been around for 8 years or longer. However, it is also possible

[20] *n* takes the size of the actual number of predicted positive surprises.

that randomly buying and holding firms over earnings announcement days can itself be a profitable strategy due to the inherent riskiness of these periods that may not be accounted for by the 5-factor asset-pricing model.

Figure 8 shows that the 15% strategy performs very well even though it has less than one-third of the firm quarter observations than that of the 5% strategy. At the end of the trading period, the portfolio value of the 15% strategy finished at a similar portfolio value to that of the 5% strategy. However, the 15% prediction model seems to produce the most consistent results over the sample period. It experienced no period of excessive draw-downs as has been witnessed with the other two strategies. This strategy was profitable for all but one year, with a loss smaller than 7%. Concerning the last third of the sample period, the strategy did not lose 20-40% of its value like the other strategies; instead, it maintained its value over this time. The reason has to do with the selection of firms further away from the lower bound, and it has a better good-to-bad outcome ratio than the other thresholds and a better precision rate.

The overall accuracy score is 82% for the 15% strategy and somewhat lower at 76% and 66% for the 10% and 5% strategies, respectively. This result echoes the machine learning evaluation results above, for which a higher surprise has more easily distinguishable patterns for the machine learning model to train on, leading to better prediction success. Also, because of the increased magnitude of the surprise, it almost directly translates to bigger returns. Larger surprise thresholds were also tested, but they performed poorly due to the lack of available opportunities at that high threshold. For example, when using a 30%+ surprise threshold, it led to only 214 surprise predictions as opposed to 774 for the 15% strategy. Another effect is the S-shaped surprise return curve that leads to lacklustre improvements in return for bigger surprises.

Table 12 shows that there is potential to earn abnormal returns by using a stop-loss strategy in combination with trading on the predictions of the machine learning model. The stop loss is set at the 10% level, but to be conservative, we selected 20% as the slippage stop-loss. There is very little research on an empirically justifiable level of slippage in event trading strategies; for that reason, I arbitrarily double the executed stop-loss level to settle for a 20% loss. This means that a 10% or more decrease, from the previous day's closing versus the current day's low, is the trigger, but that the effective loss is recorded at 20%. Therefore, no firm has experienced a return loss of more than 20% in the sample. Because the stock would always have been sold the next day, it does not significantly worsen the expected transaction cost. The stop-loss strategy significantly limits the downside of the strategy. In

essence, we do not have to hold on to the stocks until the end of the next day and can sell them off throughout the day as we see fit.

A stop-loss strategy is especially important for event-trading when you fully invest in each event. Grossman and Zhou (1993), show that with drawdown constraints of risk control, a continuous stop-loss strategy is optimal. Stop-loss strategies are not very popular in literature, but in practice, one would be hard-pressed to find event trading strategies where they are not used. Lhabitant (2011) notes that event traders such as merger arbitrageurs usually set up strict stop-losses rules for each transaction and that sticking to this discipline is one of only a few ways in which investors can limit their downside risk. Authors such as Kaminski and Lo (2014) show the theoretical underpinning of a stop-loss strategy and the fact that stop-loss policies can increase expected return substantially while reducing volatility. Han, Zhou and Zhu (2016) show an empirical justification of a stop-loss strategy as it is applied to a momentum strategy. They show an almost doubling in abnormal returns using a disciplined stop-loss strategy.

The current slippage assumes that if the stock falls by 10%, then the stop-loss sells at a 20% loss. The break-even slippage for long portfolios for the 5%, 10%, and 15% surprise thresholds are 38%, 45%, 48% respectively, offering a large margin of safety. Further investigation reveals that the reason the stop-loss performs so well is because, near the end of the sample period (2014-2016) there are about 4 portfolio days where a small amount of bad performing observations are the sole constituents to the portfolio each dropping in value between 60% - 80%. As a result, less than 0.25% of firms are accountable for wiping out the abnormal returns.

All the long strategies in *Table 12* show significance at the 99% level. Shorting predicted negative surprise does not seem to be a viable undertaking. This is mostly attributable to the results reported in the contingency tables in the first part of the study, showing that predicted negative surprises are often in reality positive surprises. Recall that *Table 6* specifies that for every two-predicted positive surprise there is less than one mistaken negative surprise of the same deviation, whereas, for every predicted negative surprise there is one mistaken positive surprise of the same deviation. *Table 12* shows that the performance of the long strategy improves as higher thresholds are predicted; the only issue is that the number of trading opportunities decrease from 1909 for the 5% threshold strategy to 252 for the 15% threshold strategy out of about 3000 trading days.

Table 12: Daily Abnormal Returns All Firms Stop-Loss Strategy

Threshold	Position	Firms Qtrs.	% SL	Surprise Days	Abnormal Returns
5%+	Long	34602	0.03	1909	0.26%***
	Short	1722	0.17	431	0.62%*
10%+	Long	15222	0.04	1449	0.32%***
	Short	1170	0.26	313	0.09%
15%+	Long	9996	0.06	1206	0.41%***
	Short	957	0.00	252	0.41%

In this strategy we use all the firms in the sample. Long identifies the positions taken in expected positive surprises days; short is the short-selling of a portfolio of firms where negative surprises are expected. Firm quarters are the number of earnings quarters used across all surprise portfolios. %SL is the percentage of triggered stop loss firms. Surprise days is the number of portfolios that were successfully formed over the sample period. The market coefficient is significant for all strategies. All short strategies show HML significance; all long strategies show SMB significance. *Table A16* further reports on the size of the positive and negative coefficients and their significance. The number of surprise days identifies the successfully formed portfolios where at least one firm experienced an earnings event the following day out of all 2994 possible trading days. *10% significance **5% significance ***1% significance.

VI. Analyst Bias or Something Else?

Kleinberg et al. (2018) use machine learning models to understand judges' mistakes and to offer advice on how judges can improve their decision making in trials. Similar to their study, I investigate observed biases and mistakes made by analysts. Machine learning models help us to identify biases by identifying variables that are important in predicting earnings over and above analysts' consensus forecast. The reason why the machine learning model is able to predict earnings surprises, seems to be the result of a few advantages that models have over analyst consensus forecasts.

One argument is that there is enough public information for analysts to improve their predictions, but that analysts are limited by the amount of information they can process. The machine learning model, on the other hand, can process gigabytes of data without much effort. Other limitations include time restrictions; analysts might recognise important signals before the announcement but do not revise their forecast in time. Studies on analyst inefficiencies show that analysts have a tendency not to update their forecast before earnings announcements even when important revelations have been made (Trueman, 1990). In effect, analysts are limited in resources and have bounded rationality that prohibits them from making suitable forecasts.

Following from the above example, analysts may not want to update their forecast due to conflicts of interest or other biases. A significant body of research reports a large number of biases and conflicts experienced by analysts (Bagnoli, Beneish, & Watts, 1999; Bhattacharya, Sheikh, & Thiagarajan, 2006; Ramnath, Rock, & Shane, 2008). Analysts may purposefully lower their forecasts compared to their actual expectations to keep management content at the expense of forecasting accuracy. Chan, Karceski, and Lakonishok (2007) show that analysts, at least in the years before the dot-com bubble, have had a desire to win investment banking clients, which creates a conflict of interest whereby analysts strategically adjust forecasts to avoid earnings disappointments. The advantage of machine learning models is that they look past the majority of these biases and correct for this systematic pessimism of analysts. A summary of the research and variables that were found to be important in predicting earnings over and above analyst forecasts (i.e., biases) has been listed in *Table A15*.

It is possible that analysts have not sufficiently studied the security's behaviour the days before the announcement, as various signals emanating from a rigorous set of timely technical and signal processed pricing data show a high level of importance in predicting

future earnings surprises (*Table 10*). These signals can be representative of the trading behaviour of insiders before announcements or even market-wide trading as a result of management and other noteworthy announcements. One argument is that these signals have gone unnoticed and that analysts do not include a price-related analysis as part of their estimation process. To identify whether the argument is true, we can look at long-term variables, such as the quarterly earnings measures, to see if the analysts sufficiently incorporate long-term earnings factors when making forecasts. If they do not, then it is unlikely to only be an issue of time and resource constraints.

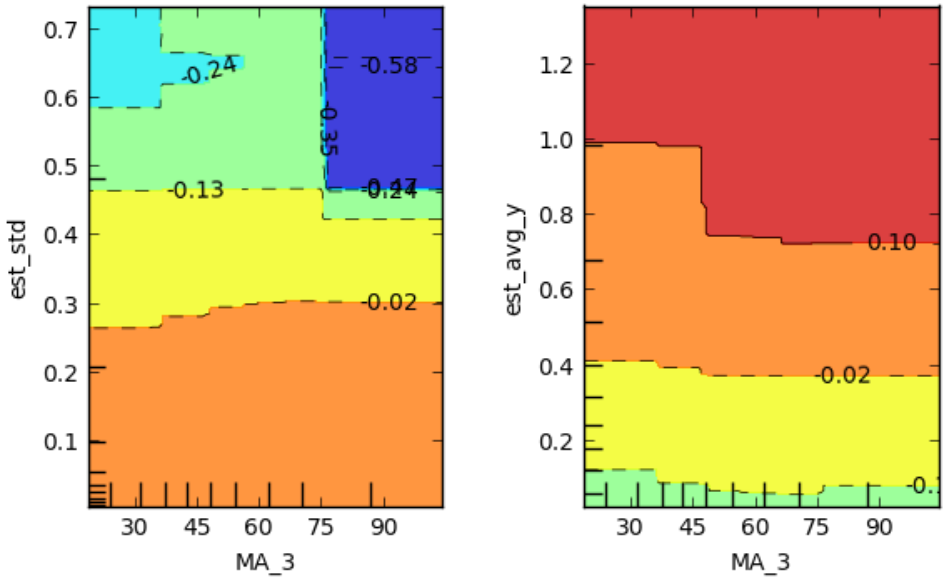
In *Table 9*, it can be seen that the consensus analysts forecast has not sufficiently accounted for the trend of earnings improvement nor changes in the short or long-term earnings ($diff_{-1}, diff_{-4}$). Analysts have also not come to grips with the interesting non-linear relationship between the two, as evidenced by *Figure 7*. Furthermore, analyst estimates do not sufficiently take into account past earnings, p_{-1}, and its interaction with the forecast, est_avg_t, when predicting earnings surprises. Some additional nonlinear relationships show that a large standard deviation among analyst forecasts are associated with lofty analyst consensus forecasts; for example, it is known that analysts show herding behaviour, leading to lower than expected standard deviation (Trueman, 1990). Analyst also do not seem to look at their own errors over time; in fact, the running past analysts forecast errors, $d\_e\_diff_{-4}$, have been shown to be important in predicting the likelihood of surprises. Analysts are very slow to adjust their mind on a company's perceived earnings development.

A growing body of literature has documented the fact that analysts do not fully incorporate or are not motivated to incorporate certain effects in their forecasts, such as the firm's efforts to manage earnings; it is, therefore, possible that these above-mentioned biases fall in the same category (Abarbanell & Lehavy, 2003). A possible outcome of this is that abnormal returns can be realised if the market is unaware of these differences. This is indeed the case; the market does not seem to be fully cognizant of existing analyst biases since abnormal returns can be earned on the predictions of surprises.

The model shows that a low est_std is an indication of herding; as a result, the next analyst would be better off putting more weight on the past earnings, p_{-1} than on the current consensus forecast, est_avg_t, when predicting the target value, p. There are other higher dimensional relationships that are too convoluted to describe and many more that we are not even aware of. As an illustrative example, *Figure 9* presents two difficult price and earnings

relationship at a higher dimension with the purpose of predicting the level of earnings as opposed to occurrence of an earnings surprise.

Figure 9: Partial Dependence of EPS Value on Firm Feature Combinations



Both the est_std and est_avg_t display a nonlinear relationship with *MA_3 (Short for MA_3__max_langevin_fixed_point__m_3__r_30).* To get an idea of the complexity, on top of the above relationship, est_std and est_avg_t also display a strong linear relationship between them as is noted by the 2nd chart in *Figure A15.*

Looking at the plot on the left, when both the standard deviation, *est_std*, and a rolling moving average of the last three days, transformed to fit and find the maximum fixed point on a Brownian motion function, i.e., a *MA_3__max_langevin_fixed_point__m_3__r_30,* are large in value, then the predicted EPS becomes much less than if only one of these variables were large. It can be argued that the combination of these two values provides a valuable measure of market uncertainty, promoting the adjustment of the predicted EPS.

The point of the above description is to show that, as humans, we have bounded rationality and struggle to understand certain relationships, especially relationships in higher dimensions. Thus, without advanced methodologies to uncover these seemingly intractable relationships, analysts would never know how to change their forecasts to become less error-prone, even when they are reported in the literature. It is likely that a large proportion of analysts do not use non-linear techniques to track their biases and that they prefer to make use of models or methodologies that make intuitive sense and are driven by firm policies; and for that reason alone, analysts may be performing worse than the machine learning model.

Although it is hard to test, it is also possible that analyst concern themselves with irrelevant information when making decisions, such as news hype or any other information unobservable to the model, which can lead to worse predictions.

A final possibility is that once analyst forecasts are made public, the target firm may manage earnings upwards in an attempt to earn a small positive surprise (Burgstahler & Dichev, 1997; Burgstahler & Eames, 2006). This argument is difficult to disprove, but this effect is unlikely to explain the significant abnormal profits that can be earned. If analysts' predictions improve over time, while consistently underpredicting earnings, it could be a sign that the firm manages earnings based on public forecasts. See *Table A15* for a summary of which biases identified by past research also show strong predictive power in this study's machine learning models.

VII. Conclusion

A machine learning model with earnings and price variables as inputs performs significantly better at predicting earnings surprises than a random choice benchmark. Surprises that deviate 15% or more can be predicted with 71% accuracy. Exploiting this predictability allows for the construction of profitable trading opportunities. The explanation for this improved performance seems to be three-fold: (1) even though there is enough public information for analysts to improve their predictions, analysts experience an information overload that does not affect machine learning models; (2) the machine learning model corrects for known unobservable biases often experienced by analysts, as evidenced by the list of important variables; (3) the model picks up inside or suspicious trading behaviour by investigating a rigorous set of timely technical and signal-processed variables derived from pricing and volume data.

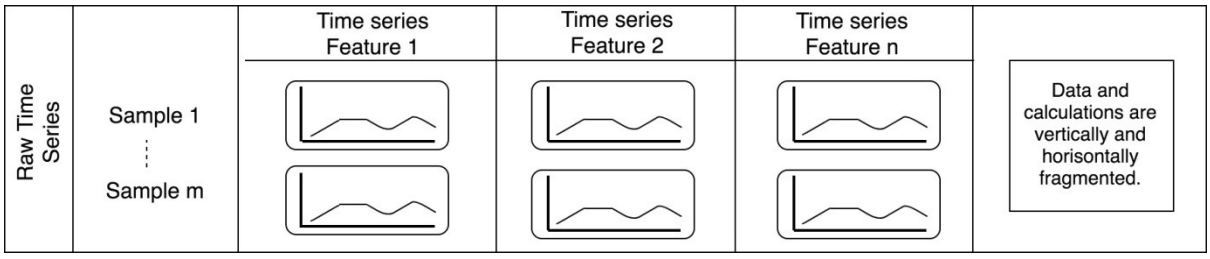
Future research should look at incorporating additional task-relevant variables to improve the prediction model. It is possible that a wider range of fundamental, sentiment, and descriptor variables will further enhance the performance of the current model. Future studies can also attempt to improve the predictions by introducing online machine learning, which in essence means the updating and/or retraining of machine learning models as soon as new data is made available. In finance, new data is created by the second, so one would have to weigh up the performance gain versus the cost of prediction. In the future, the performance can also be improved by developing stacked models where multiple models contribute to the final prediction.

VIII. Appendix

Method A 1: Signal Processing and Feature Selection

The technical indicators that I incorporate in the study include well-known and lesser-known indicators. These technical variables are essentially some decomposition of momentum indicators that quantifies the relationship between recent price changes in a given window and the long-term trend of the instrument. All indicators make use of historical data and are recalculated on a rolling basis for 30 days. Since some of these variables range many orders of magnitude, I will normalise certain variables with their mean and standard deviation merely for data compression reasons. Multiple studies show that the variable space should not just consist of technical indicators and that it can be improved by incorporating other values that are likely to be uncorrelated with the price variables (Dhar & Chou, 2001; Hellstrom & Holmstrom, 1998). On the fundamental front, this study will look at past quarters' reported earnings information such as the EPS forecast, the count, and the standard deviation of analyst forecasts. The next step is to merge all the pricing variables and to present them in a columnar format with exactly 30 trading days' worth of data before all announcements to easily compute the signal processed transformations and create a new set of variables for model inputs.

Figure A10: Columnar Time-series Format

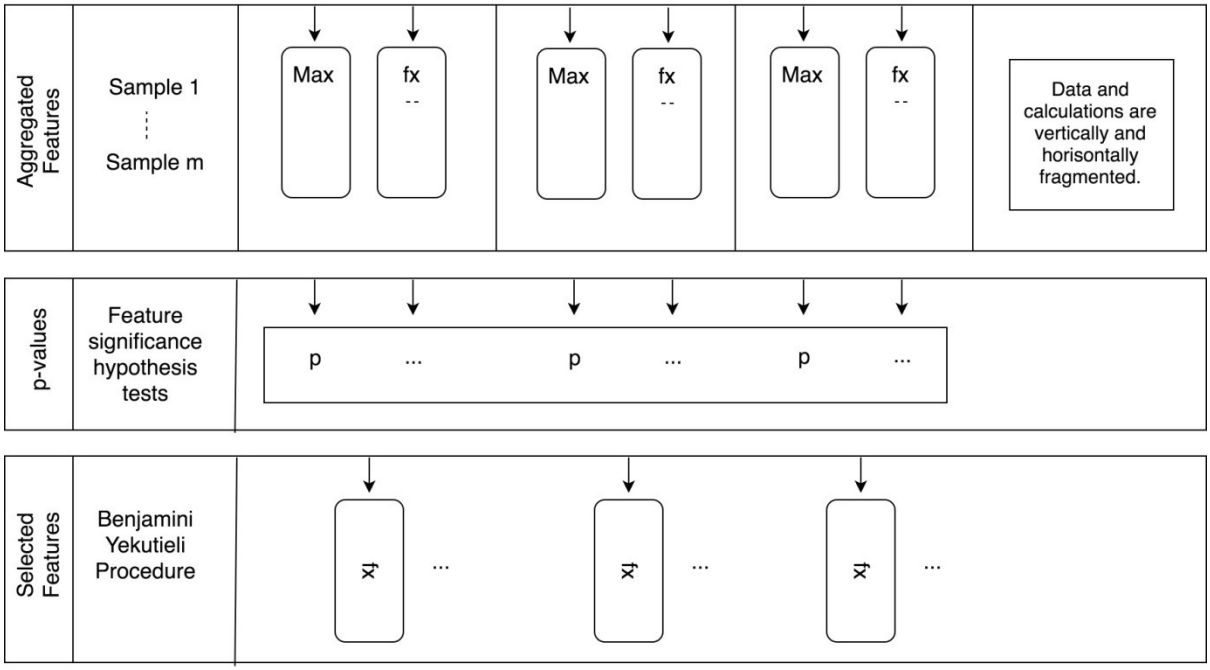


Items (1) and (2) *Figure 1* are put in columnar format, as above, for every firm quarter in the sample period. As shown in *Figure 2*, only data 30 days prior to the official announcement is incorporated. Each time-series constitutes either a series of normal (1) or technical indicators (2).

Once all the price variables have been gathered or computed and are in raw columnar format, then as *Figure A11* illustrates, functions get applied over these chunks of raw data. A final step is the test of significance. This process decides whether or not a newly transformed variable will be kept after observing its effectiveness at predicting the response variable. A p-value of 0.05 has been selected as the significance level. I further use a two-stage solution

that identifies the relevant variables. It is an efficient algorithm that filters the variables in an early stage of the machine learning process with respect to their significance to the classification task, while controlling the expected percentage of selected but non-relevant variables.

Figure A11: Signal Processing Transformations and Feature Selection



The next step is to run all the functions as listed in *Table A14* in which we calculate the significance of each signal value in predicting the target and select the variables based on the Benjamini-Yekutieli Procedure. This is the last step in this variable creation and selection process.

In the first step, all aggregated variables are separately and independently evaluated with respect to their significance for predicting the response variable under investigation using a univariate test. The result of these tests is a vector of p-values. This then quantifies the significance of each variable for predicting the target (response). In *Figure A11* below, this corresponds to the change from aggregated variables to p-values. The vector of p-values is then evaluated on the basis of the Benjamini-Yekutieli-procedure to decide which variables to keep. This is simply a multiple testing procedure that decides which variables to keep and which to cut off based solely on the use of the p-values. This test controls the false discovery rate, which is the ratio of false rejections by all rejections:

$$FDR = \mathbb{E}\left[\frac{|false\ rejections|}{|all\ rejections|}\right] \quad (7)$$

This means that the percentage of irrelevant variables among the extracted variables will be asymptotically controlled by the filtering. This study essentially makes use of classical statistical methods to select variables. This process is unique and better than other variable selection algorithms, such as Boruta, which do not give any insights into how many good or bad variables they filter out (Kursa & Rudnicki, 2010). Often these algorithms 'just work.' Lastly, the process followed in this study is highly scalable: its calculations and the data can be distributed over a cluster and can be performed in parallel.

Once all of the variables have been successfully sculpted and selected, a separate round of variable selection is performed at two critical points, once before executing the classification model and once before executing the model. These procedures incorporate all variables calculated up to this point. The aim of the procedures is simply to separate all the relevant factors from the irrelevant. Feature selection has many benefits; it decreases the computational time, often increases the accuracy of the model, and also simplifies the model to make it easier to understand (Liu & Motoda, 2012).

The second variable selection procedure occurs after running all variables through multiple models to prompt the variable importance values. This technique is different from the first technique and is based on information gain. The approach I have used to uncover the most prominent variables is a tree-based variable selection procedure. I made use of a combination of random forest, gradient boosting, and AdaBoost variable selection procedures. All the variables are calculated as a line vector and aggregated together in a matrix labelled 'variables.' These variables are individually tested by each algorithm on a hold-out set and the relevance scores are summed together after which the top 800 variables are selected for the classification task.

Method A 2: Classifier Learning

To create the overall ensemble model, such as presented by the *Classifier* pseudocode in the methodology section, IV.C.2, we have to establish a loss function, *L* to minimise, so as to optimise the structure and performance of the model. This function has to be differentiable as we want to perform a process of steepest descent, which is an iterative process of

attempting to reach the global minimum of a loss function by going down the slope until there is no more room to move closer to the minimum. We, therefore, solve for by minimizing a loss function numerically via the process of steepest descent. The focus here is on $f(\boldsymbol{x}_i)$ as this is the compressed form of the predictor of each tree *i.*

For our classification task, we use logistic regression to obtain the probabilistic outputs of the target variable.

$$L(\theta) = \sum_i [\boldsymbol{y}_i \, ln(1 + e^{f(x_i)}) + (1 - \boldsymbol{y}_i) ln(1 + e^{f(x_i)})] \tag{8}$$

$$L(\theta) = \sum_i [\boldsymbol{y}_i \, ln(1 + e^{-\hat{y}\, i}) + (1 - \boldsymbol{y}_i) ln(1 + e^{\hat{y}\, i})] \tag{9}$$

Further, it is necessary to minimise the loss over all the points in the sample, $(\boldsymbol{x}_i, y_i)$:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} L(\theta) \tag{10}$$

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} L(y_i, f(\boldsymbol{x}_i)) \tag{11}$$

At this point, we are in the position to minimise the predictor function, $f(\boldsymbol{x}_i)$, w.r.t. $\boldsymbol{x}$ since we want a predictor that minimises the total loss of $f(\boldsymbol{x})$. Here, we simply apply the iterative process of steepest descent. The minimisation is done in a few phases. These phases are better described in the next appendix section, *Method A 3*, but a short summary follows. The first process starts with adding the first and then successive trees. Adding a tree emulates adding a gradient-based correction. Making use of trees ensures that the generation of the gradient expression is successful, as we need the gradient for an unseen test point at each iteration, as part of the calculation $f(\boldsymbol{x})$. Finally, this process will return $f(\boldsymbol{x})$ with weighted parameters. The detailed design of the predictor, $f(\boldsymbol{x})$, is outside the purpose of the study, but for more extensive computational workings, see the next section.

Method A 3: Detailed XGBoost Design and Supervised Learning

This part can be skipped if you are already familiar with supervised learning and, more specifically, Gradient Boosted Trees. A large part of the detailed workings has been obtained from the official XGBoost documents but have been altered to improve understanding (Chen & Guestrin, 2016). Supervised learning refers to the mathematical structure describing how to make a prediction $\boldsymbol{y_i}$ given $\boldsymbol{x_i}$. In classification task prediction, $\boldsymbol{y_i}$ is the probability of an earnings surprise event of some specified threshold. The inputs, $\boldsymbol{x_i}$, have been selected based on the applied selection procedures. Apart from the different prediction types, in the classification task, the model gets logistic transformed to obtain a vector of probabilities for each observation and associated categories. In supervised learning, parameters play an important role. The parameters are the undetermined part that we are required to learn using the training data. For example, in a linear univariate regression, $\hat{y}_i = \sum_j \theta_j \boldsymbol{x_{ij}}$, the coefficient θ is the parameter.

The task is ultimately to find the best parameters and to choose a computationally efficient way of doing so. To measure a model's performance, given some parameter selections, we are required to define an objective function. The following is a compressed form of the objective function, $Obj(\Theta) = L(\theta) + \Omega(\Theta)$. In this equation, L is the training loss function; the regularisation term is Ω. The training loss function tests the predictive ability of the model using training data. A commonly used method to calculate the training loss is the mean squared error, $L(\theta) = \sum_i (\boldsymbol{y}_i - \hat{y}_i)^2$. Thus, the parameters get passed into a model that calculates, $\boldsymbol{\hat{y}_i}$, a series of predictions, that gets compared against the actual values in a mean squared error function to calculate the loss.

The regularisation term controls the complexity of the model, which helps to avoid overfitting. The Extreme, X, of the XGBoost model, relates to an extreme form of regularisation that controls for over-fitting, leading to improved performance over other models. There are countless ways to regularise models in essence, we constrain a model by giving it fewer degrees of freedom; for example, to regularise a polynomial model, we can transform the model to reduce the number of polynomial degrees. The tree ensemble can either be a set of classification or a set of regression trees. It is usually the case that one tree is not sufficiently predictive, hence the use of a tree ensemble model that sums the predictions of many trees together. Mathematically, the model can be written in the following form $\hat{y}_i = \sum_{k=1}^{K} f_k(\boldsymbol{x_i}), f_k \in F$. Here, K is the number of trees, and f represents one possible function

from the entire functional space *F*. *F* is a set of all possible classification and regression trees (CARTs). This expression then simply adds multiple models together that lives within the allowable CART function space. Therefore, combining the model, the training loss, and the regularisation function, we can gain our objective function and seek to optimise it. The function can be written as follows, $Obj(\theta) = \sum_i^n l\left(\boldsymbol{y_i},\ \hat{y}_i^{(t)}\right) + \sum_{k=1}^K \Omega(f_i)$. Thus far, the model is similar to that of a random forest, the difference being in how the models are trained.

For the next part, we have to let the trees learn, so for each tree, we have to describe and optimise an objective function. We can start off by assuming the following function, $Obj = \sum_i^n l(y_i,) + \sum_{i=1}^t \Omega(f_i)$. By looking at the function, it is important that we identify the parameters of the trees. We want to learn the functions, f_i, each of which contains a tree structure and associated leaf scores. This is more complex than traditional methods where you can simply take the gradient and optimise for it. Instead, Gradient Boosting uses an additive strategy, whereby we learn to adjust and add an extra tree after each iteration. We write our prediction value at step *t* as $\hat{y}_i^{(t)}$, so that we have $\hat{y}_i^{(t)} = \sum_{k=1}^t f_k(\boldsymbol{x_i}) = \hat{y}_i^{(t-1)} + f_t(\boldsymbol{x_i})$. Then we simply choose the tree that optimises our objective, $Obj^{(t)} = \sum_i^n l\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{k=1}^t \Omega(f_i) = \sum_{i=1}^n l\left(\boldsymbol{y_i}, \hat{y}_i^{(t-1)}\right) + f_t(\boldsymbol{x_i}) + \Omega(f_t) + constant$. By using MSE as the loss function, it becomes $Obj^{(t)} = \sum_{i=1}^n \left[2\left(\hat{y}_i^{(t-1)} - \boldsymbol{y_i}\right) f_t(\boldsymbol{x_i}) + f_t(\boldsymbol{x_i})^2\right] + \Omega(f_t) + constant$. The form of MSE is easy to deal with. The Taylor expansion can simply be taken to the second order. $Obj^{(t)} = \sum_{i=1}^n \left[l\left(y_i, \hat{y}_i^{(t-1)}\right) + g_i f_t(\boldsymbol{x_i}) + \frac{1}{2} h_j {f_t}^2(\boldsymbol{x_i})\right] + \Omega(f_t) + constant$, where g_i and h_i is defined as, $g_i = \partial_{\hat{y}_i^{(t-1)}} l(\boldsymbol{y_i}, \hat{y}_i^{(t-1)})$, $h_i = \partial^2_{\hat{y}_i^{(t-1)}} l(\boldsymbol{y_i}, \hat{y}_i^{(t-1)})$. After all the constants are removed, then the objective at *t* get transformed to, $\sum_{i=1}^n \left[g_i f_t(\boldsymbol{x_i}) + \frac{1}{2} h_j {f_t}^2(\boldsymbol{x_i})\right] + \Omega(f_t)$. This then becomes an adjusted optimization function for the new tree. Although we have looked at the training step, we have not looked at regularisation yet. The next step is to specify how complex the tree should be, $\Omega(f_t)$. To do this we can improve the tree definition to *F(x)*, $f_t(\boldsymbol{x}) = w_{q(\boldsymbol{x})}, w \in \mathbb{R}^T, q: \mathbb{R}^m \rightarrow \{1, 2, \dots, T\}$. Here *w* represents the scores of the leaves presented in vector form and *q* represents a function that assigns each point to the appropriate leaf; lastly *T* denotes how many leafs there are. The complexity can be defined as $a\ \Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^T w_j^2$; there are more ways to formulate and define how complex a model is or should be in practice, but this

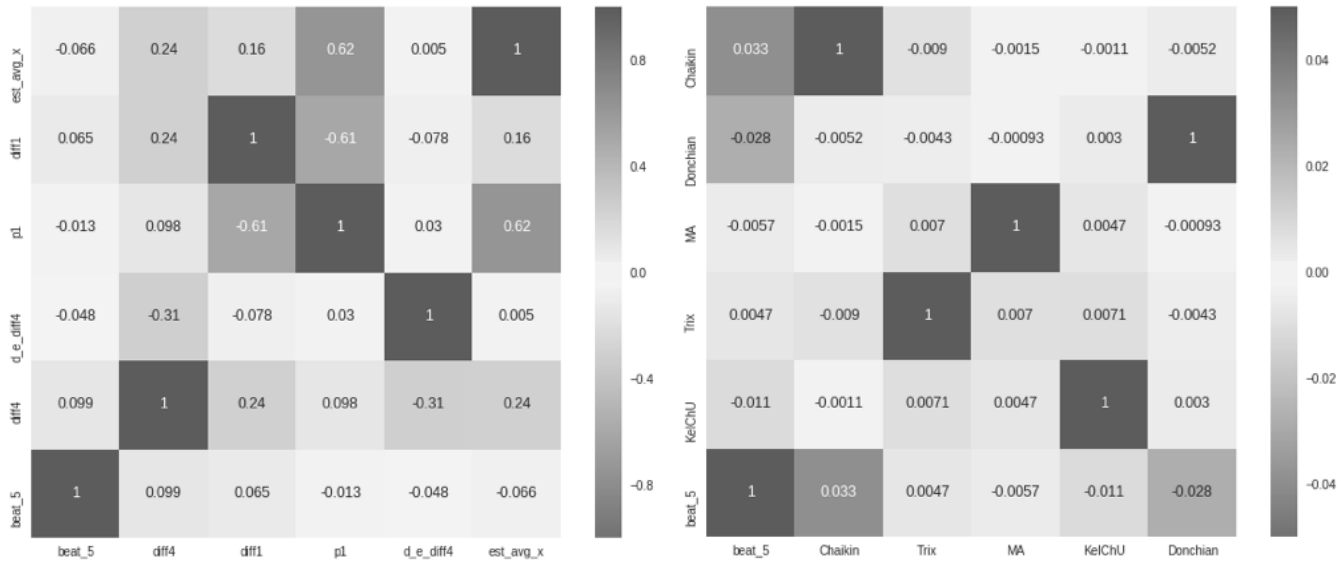
one is quite practical and easy to conceptualise. Once the tree model is described, the objective value w.r.t. the *t-th* tree can be written as follows: $Obj^{(t)} = \sum_{i=1}^{n}\left[g_i f_t(\boldsymbol{x_i}) + \frac{1}{2}h_j {f_t}^2(\boldsymbol{x_i})\right] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2 \quad = \quad \sum_{J=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right)w_j + \frac{1}{2}\left(\sum_{i\in I_j} g_i + \lambda\right)w_j^2\right] + \gamma T$, where $\mathrm{I}_j = \{\mathrm{i}|\mathrm{q}(\mathrm{x}_j) = \mathrm{j}\}$ represents a full set of all the data points as have been assigned to the *j-th* leaf. The equation can then further be compressed by describing $G_j = \sum_{i\in I_j} g_i$ and $H_J = \sum_{i\in I_j} h_i$, then $Obj^{(t)} = \sum_{J=1}^{T}\left[G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2\right] + \gamma T$. In the preceding equation the weights w_j are independent w.r.t each other, the form $G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2$ is quadratic, and the best weight for a structure *q(x)* is given by the following expression. $w_j^* = -\frac{G_j}{H_j+\lambda}$, $obj^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda} + \gamma T$. This equation measures how good a tree structure $q(x)$ is. A lower score is better for the ultimate structure of a tree. Now that we know how to measure the fittingness of a tree, we can identify all the trees and select the best one. It is, however, not possible to approach it this way and instead has to be done for one depth level of a tree at a time. This can be approached by splitting a leaf into two sections and then recording its gain. The following equation represents this process, $Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_l+G_l)^2}{H_L+H_R+\lambda}\right] - \gamma$. If the gain obtained is equal to or smaller than γ, then it would be better if we do not add the branch to the tree; this is often referred to as the pruning technique. We basically search for the ultimate split; if all instances are sorted in order, we simply scan left to right to sufficiently calculate the structure scores of all possible solutions and then identify the most efficient split.

Method A 4: Partial Dependence (D)

For $\boldsymbol{x}_j$, a variable from a vector of variables, sort the unique values $\mathrm{V} = \{\mathbf{x}_\mathrm{j}\}_\mathrm{i} \in \{1, \ldots, \mathrm{n}\}$ resulting in V^*, where $V^* = K$. Create K new matrices $\boldsymbol{X}^k = \left(\boldsymbol{x}_j = V_k^*, \boldsymbol{X}_{-j}\right), \forall \mathrm{k} = (1, \ldots, K)$. Then drop each of the K new datasets, $\boldsymbol{X}^k$**,** down the models' fitted trees predicting a new value for each observation in all *k* datasets: $\widehat{\boldsymbol{y}}^k = \widehat{f}(\boldsymbol{X}^k), \forall \mathrm{k} = (1, \ldots, K)$. Then average the prediction in each of the K datasets, $\hat{y}_k^* = \frac{1}{n}\sum_{i=1}^{N}\widehat{\boldsymbol{y}}_\mathrm{i}^k, \forall \mathrm{k} = (1, \ldots, K)$. Lastly, simply plot V^* against $\widehat{\boldsymbol{y}}^*$ to visualise the relationship. The above strategy shows the dependence of the target function on a set of target variables by marginalising over the values of other variables.

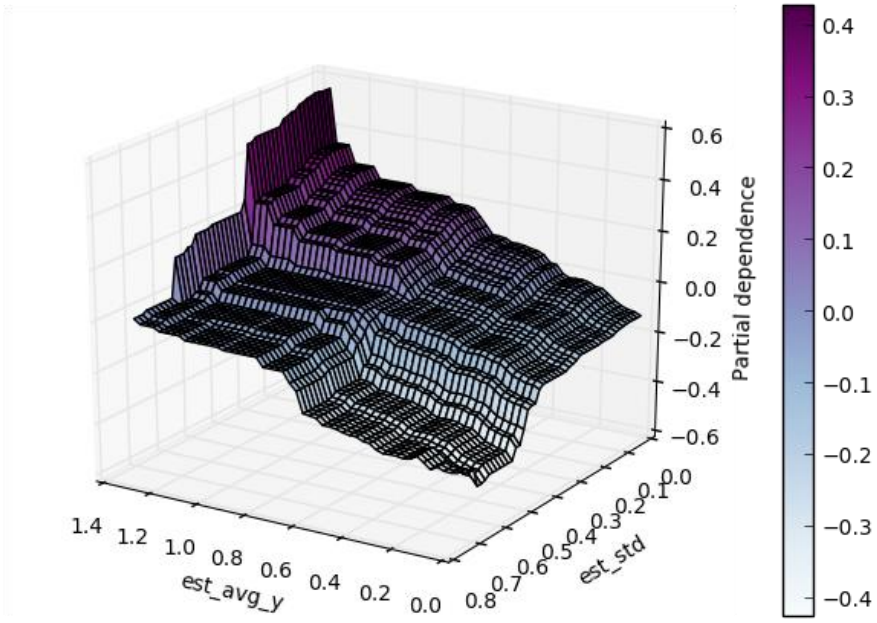
In more simple terms, the above strategy shows the dependence between the target function and a set of target variables by marginalising over the values of other variables

Figure A12: Classification Correlation Matrix for Earnings and Price Variables



This is an example of the correlation matrices for the top variables in the classification model. The left side represents the earnings-related variables and the right side the price-related variables. The purpose of this graph is to show that there is multicollinearity between the variables. This result makes us more cautious about assigning variable significance or importance; the reason is that multiple variables may represent the same dynamic.

Figure A13: Assorted Interaction Charts



This is a three-dimensional way of presenting the interaction of the analyst forecast and standard deviation and the resulting response output using the partial dependence method. This graph clearly shows that to achieve the most accurate EPS prediction, the forecast should be readjusted downward as increased uncertainty, proxied by the standard deviation of forecasts, leads to excessively positive forecasts.

Figure A14: Partial Dependence Classification - Earnings Related

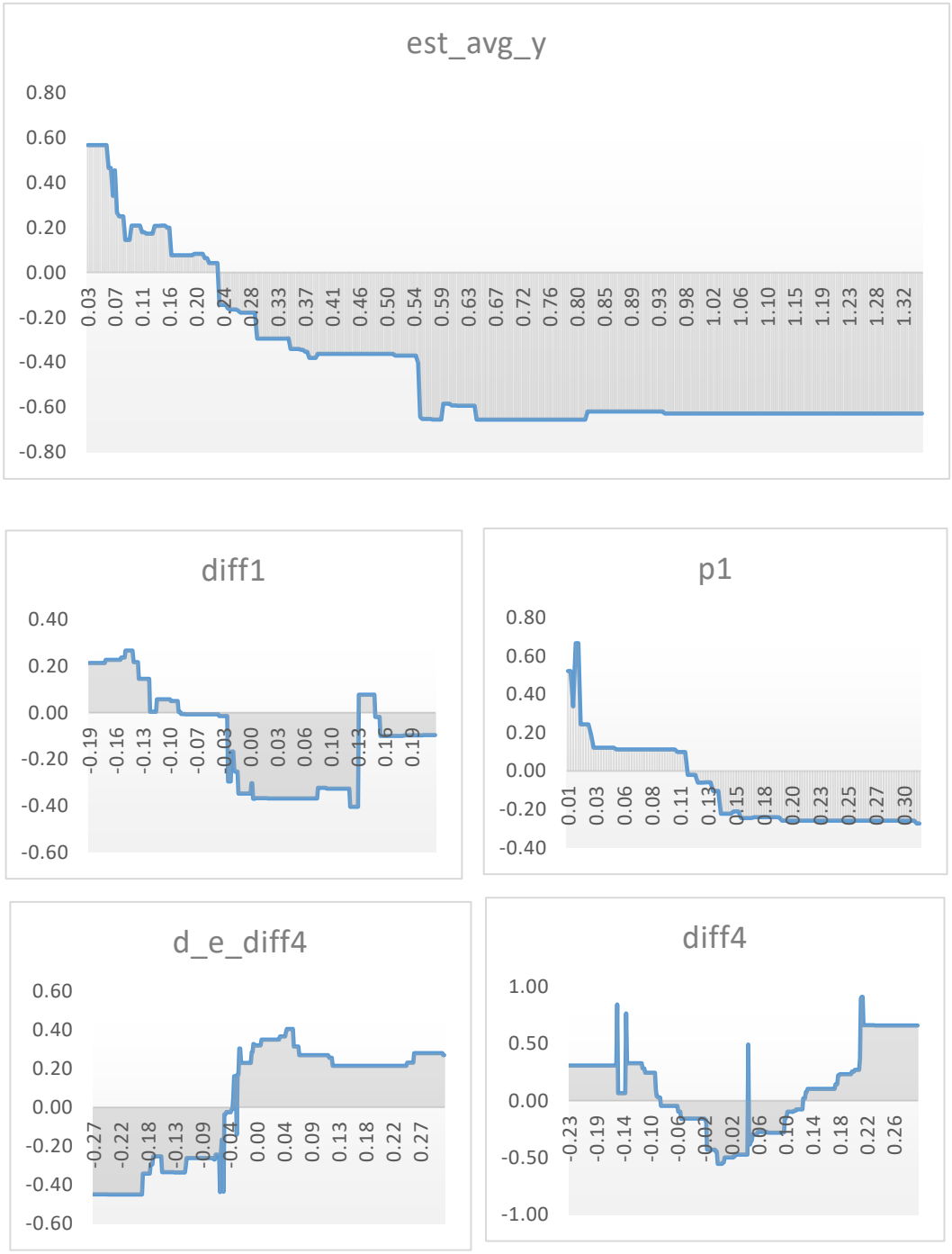


Figure A15: Partial Dependence Classification - Price Related

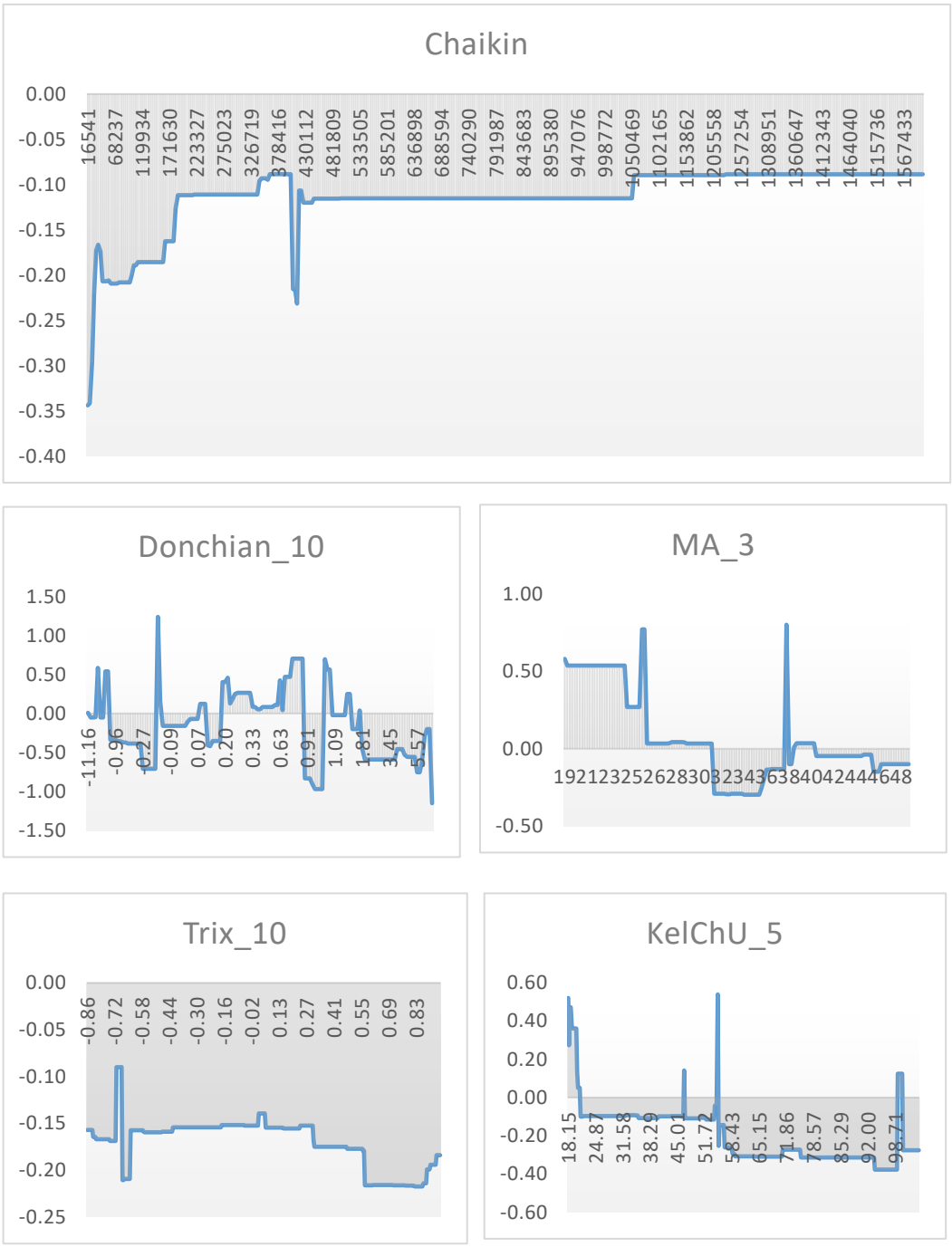


Table A13: Machine Learning for Finance Glossary

Accuracy	The rate of correct predictions made by the model over a data set. Accuracy is usually forecasted by using an independent test set that was not used at any time during the learning process.
Boosting	A technique for combining models based on adaptive resampling - where different data is given to different models. The idea is to successively omit the 'easy' data points, which are already well modelled, so that the later models focus on the left-over 'hard' data points.
Class	The same as category.
Classifier	A mapping from unlabelled instances to classes. Classifiers have a form (e.g., decision tree) plus an interpretation procedure. Some classifiers also provide probability forecasts (scores). Classifiers are used to predict class labels.
Confusion matrix	A matrix showing the predicted and actual classifications. A confusion matrix is of size LxL, where L is the number of different label values.
Cross-validation	A method for estimating the accuracy of an inducer by dividing the data into *k* mutually exclusive subsets of approximately equal size. The inducer is trained and tested *k* times. Each time, it is trained on the data set minus a fold and tested on that fold. The accuracy forecast is the average accuracy for the *k* folds.
Feature selection	The process of removing variables which seem irrelevant for modelling.
Variables	Explanatory and independent variables. It is the measurements and characteristics that represent the data.
Instance	Observations, ex. firm quarters. A single object of the world from which a model will be learned, or on which a model will be used (e.g., for prediction).
Machine learning	The field of scientific study that concentrates on inductive algorithms that can be said to 'learn.'

Mapping	Applying a function to all elements of a list in order and returning a list of results.
Model	Most inductive algorithms generate models that can then be used as Classifiers or Regressors.
Overfitting	A modelling error that occurs when a function is too closely fit to a limited set of data points leading to poor out-of-sample generalisation.
Regression	Predicting the value of random variable *y* from measurement *x*. For example, predicting EPS based on Estimates, Size and P/E. Regression is used to predict continuous values. It does not have the same meaning as in finance; it is concerned with using *regression* to determine the strength of relationships between dependent and independent variables.
Regressor	A mapping from unlabelled instances to a value within a predefined metric space, e.g., a continuous range.
Regularization	Any estimation technique designed to impose a prior assumption of 'smoothness' on the fitted function. Use to alleviate overfitting and model complexity.
Signal Processing	Concerns the analysis, synthesis, and modification of signals, which are broadly defined as functions conveying 'information about the behaviour or attributes of some phenomenon.'
Supervised learning	Techniques used to learn the relationship between independent attributes and a designated dependent attribute (the label). Most induction algorithms fall into the supervised learning category
Target Variable	Response variable, Dependent variable. It is the variable being predicted in supervised learning, whether it be categorical or continuous.

Table A14: Signal Processing and Other Functions

Name and Parameter	**Description**
abs_energy(x)	Returns the absolute energy of the time-series, which is the sum over the squared values.
absolute_sum_of_changes(x)	Returns the sum over the absolute value of consecutive changes in the series *x*.
acf(x[, unbiased, nlags, qstat, fft, alpha])	Autocorrelation function for 1d arrays.
adfuller(x[, maxlag, regression, autolag])	Augmented Dickey-Fuller unit root test.

agg_autocorrelation(x, param)	Calculates the value of an aggregation function.
agg_linear_trend(x, param)	Calculates a linear least-squares regression for values of the time-series that were aggregated over chunks versus the sequence from 0 up to the number of chunks minus one.
approximate_entropy(x, m, r)	Implements a vectorised approximate entropy algorithm.
ar_coefficient(x, param)	This variable calculator fits the unconditional maximum likelihood of an autoregressive $AR(k)$ process.
augmented_dickey_fuller(x, param)	The Augmented Dickey-Fuller test is a hypothesis test that checks whether a unit root is present in a time-series sample.
autocorrelation(x, lag)	Calculates the autocorrelation of the specified lag.
binned_entropy(x, max_bins)	First bins the values of *x* into equidistant bins.
c3(x, lag)	This function calculates the value of nonlinearity in time-series (Schreiber, 1997).
change_quantiles(x, ql, qh, isabs, f_agg)	First fixes a corridor given by the quantiles ql and qh of the distribution of *x*.
count_above_mean(x)	Returns the number of values in *x* that are higher than the mean of *x*.
count_below_mean(x)	Returns the number of values in *x* that are lower than the mean of *x*.
cwt(data, wavelet, widths)	Continuous Wavelet Transform.
cwt_coefficients(x, param)	Calculates a Continuous Wavelet Transform for the Ricker wavelet, also known as the "Mexican hat wavelet."
energy_ratio_by_chunks(x, param)	Calculates the sum of squares of chunk *i* out of *N* chunks expressed as a ratio with the sum of squares over the whole series.
fft_coefficient(x, param)	Calculates the Fourier coefficients of the one-dimensional discrete Fourier Transform.
find_peaks_cwt(vector, widths[wavelet])	Attempts to find the peaks in a 1-D array.
first_location_of_maximum(x)	Returns the first location of the maximum value of x.
first_location_of_minimum(x)	Returns the first location of the minimal value of x.
friedrich_coefficients(x, param)	Coefficients of polynomial, which has been fitted to the deterministic dynamics of Langevin model.
has_duplicate(x)	Checks if any value in *x* occurs more than once.
has_duplicate_max(x)	Checks if the maximum value of *x* is observed more than once.
has_duplicate_min(x)	Checks if the minimal value of *x* is observed

	more than once.
index_mass_quantile(x, param)	Calculates the relative index *i* where *q%* of the mass of the time-series *x* lie left of *i*.
kurtosis(x)	Returns the kurtosis of *x* (calculated with the adjusted Fisher-Pearson standardized moment coefficient *G2*).
large_standard_deviation(x, r)	Boolean variable denoting if the standard dev of *x* is higher than '*r*' times the range = difference between max and min of *x*.
last_location_of_maximum(x)	Returns the relative last location of the maximum value of *x*.
last_location_of_minimum(x)	Returns the last location of the minimal value of *x*.
length(x)	Returns the length of *x*.
linear_trend(x, param)	Calculates a linear least-squares regression for the values of the time-series versus the sequence from 0 to length of the time-series minus one.
linregress(x[, y])	Calculates a linear least-squares regression for two sets of measurements.
longest_strike_above_mean(x)	Returns the length of the longest consecutive subsequence in *x* that is bigger than the mean of *x*
longest_strike_below_mean(x)	Returns the length of the longest consecutive subsequence in *x* that is smaller than the mean of *x*.
max_langevin_fixed_point(x, r, m)	Largest fixed point of Langevin dynamics forecasted from polynomial.
maximum(x)	Calculates the highest value of the time-series *x*.
mean(x)	Returns the mean of *x*.
mean_abs_change(x)	Returns the mean over the absolute differences between subsequent time-series values.
mean_change(x)	Returns the mean over the differences between subsequent time-series values.
mean_second_derivate_central(x)	Returns the mean value of a central approximation of the second derivative.
median(x)	Returns the median of *x*.
minimum(x)	Calculates the lowest value of the time-series *x*.
number_crossing_m(x, m)	Calculates the number of crossings of x on m.
number_cwt_peaks(x, n)	This variable searches for different peaks in x.
number_peaks(x, n)	Calculates the number of peaks of at least support n in the time-series x.
pacf(x[, nlags, method, alpha])	Partial autocorrelation forecasted.
partial_autocorrelation(x, param)	Calculates the value of the partial

	autocorrelation function at the given lag.
percentage_of_reoccurring_datapoints_to_all_datapoints(x)	Returns the percentage of unique values that are present in the time-series more than once.
percentage_of_reoccurring_values_to_all_values(x)	Returns the ratio of unique values that are present in the time-series more than once.
quantile(x, q)	Calculates the *q* quantile of *x*.
range_count(x, min, max)	Count observed values within the interval [*min, max*).
ratio_beyond_r_sigma(x, r)	Ratio of values that are more than $r \times std(x)$ away from the mean of *x*.
ratio_value_number_to_time_series_length(x)	Returns a factor which is 1 if all values in the time-series occur only once, and below one if this is not the case.
ricker(points, a)	Returns a Ricker wavelet, also known as the "Mexican hat wavelet."
sample_entropy(x)	Calculates and returns sample entropy of *x*.
set_property(key, value)	Returns a decorator that sets the property key of the function to value.
skewness(x)	Returns the sample skewness of *x* (calculated with the adjusted Fisher-Pearson standardized moment coefficient *G1*).
spkt_welch_density(x, param)	Variable calculator that forecasts the cross power spectral density of the time-series *x* at different frequencies.
standard_deviation(x)	Returns the standard deviation of *x*.
sum_of_reoccurring_data_points(x)	Returns the sum of all data points that are present in the time-series more than once.
sum_of_reoccurring_values(x)	Returns the sum of all values that are present in the time-series more than once.
sum_values(x)	Calculates the sum over the time-series values.
symmetry_looking(x, param)	Boolean variable denoting if the distribution of *x* looks symmetric.
time_reversal_asymmetry_statistic(x, lag)	This function calculates the value of a complex function shown to be promising variable to extract (Fulcher, Jones, 2014).
value_count(x, value)	Counts occurrences of value in time-series *x*.
variance(x)	Returns the variance of *x*.
variance_larger_than_standard_deviation(x)	Boolean variable denoting if the variance of *x* is greater than its standard deviation.
welch(x[, fs, window, nperseg, noverlap])	Estimates power spectral density using Welch's method.

Table A15: Variables Created Based on Past Literature and Their Appearance In Top 5 Feature Categories for Both The Classification and Regression Task

Reference	Bias/Observation	Illuminating Feature	Top 10
Beaver, 1968; Givoly and Lakonishok, 1984	To identify systematic over and under-prediction.	Multiple running forecast errors.	Yes
Latane and Jones, 1977; Brown et al., 1996	Trading strategy observing that the past difference between forecasts and surprises persists.	Rolling averages of past differences.	Yes
Brown et al., 1996	Prior stock-returns is a significant factor in a Multi-factor regression strategy.	Prior stock returns.	Yes
Lys & Sohn, 1990; Trueman, 1990	Stale forecasts.	Days since last forecast.	-
Butler and Lang, 1991	Skewness in analyst consensus.	The skewness of past forecasts and observed EPS	-
Freeman and Tse, 1992; Das, Levine, and Sivaramakrishnan, 1998	Unregular forecast dispersion between analyst forecasts; analysts more optimistic in times of uncertainty.	A running standard deviation on analysts' forecasts.	Yes
Brown et al., 1996	Earnings surprises tend to repeat.	Dummy to identify past EPS surprise occurrences and threshold levels.	-
Das, Levine, and Sivaramakrishnan, 1998	Analysts more optimistic with increased uncertainty.	The running total count of null values over a range of earnings inputs.	-
Brown, 2001	Median earnings surprise shifts over the years.	Observed and forecasted EPS median.	-
Kinney, Burgstahler, & Martin, 2002	Identified the importance of analyst coverage.	Count of individual analysts per quarter observation.	-
Barron et al. 2005	Insider trading is related to the level of trading volume before announcement dates.	Volume.	-
Anilowski, Feng, and Skinner, 2007; Lys & Soo, 1995	Earnings pre-announcements.	Signals over pricing data.	Yes
Johnson and Zhao, 2012	Surprises tend to reappear in the long run.	Count of past surprises per classification threshold.	-

Table A16: 5 Factor Model Coefficients and Significance for a Stop Loss Surprise Strategy on All Firms

Threshold	Surprise	Intercept	MktRf	SMB	HML	RMW	CMA
5%+	Positive	0.258	0.930	0.978	0.414	0.291	-0.014
		(3.118)	(11.875)	(6.411)	(2.821)	(1.084)	(-0.046)
	Negative	-0.621	0.950	0.238	1.242	-0.124	-0.985
		(-1.988)	(3.426)	(0.465)	(2.23)	(-0.131)	(-0.921)
10%+	Positive	0.317	1.070	0.795	0.189	-0.177	0.265
		(2.769)	(9.902)	(3.762)	(0.907)	(-0.483)	(0.652)
	Negative	-0.086	0.690	0.764	1.834	-1.837	-0.721
		(-0.271)	(2.461)	(1.441)	(3.419)	(-1.858)	(-0.653)
15%+	Positive	0.406	0.000	0.000	0.000	0.000	0.000
		(3.086)	(10.954)	(2.928)	(-0.711)	(0.081)	(2.409)
	Negative	-0.408	0.670	1.763	1.896	-1.053	-1.681
		(-1.187)	(2.237)	(3.028)	(3.396)	(-1.07)	(-1.437)

MktRf is the difference between R_{Mt}, the value-weighted return of the market portfolio and R_{Ft}, the risk-free rate. SMB_t, HML_t, RMW_t and CMA_t are the respective differences between diversified portfolios of small stocks and big stocks, high and low B/M stocks, robust and weak profitability stocks and low and high investment stocks. To perform the regressions, the respective daily values were obtained from Kenneth French's website. The market coefficient is significant for all strategies. All short strategies show HML significance; all long strategies show SMB significance. The regressions on the factors showed that the short strategy showed some significance at the 5% threshold. This strategy includes stop loss limits, robust to slippage. Unlike the strategy in *Table A17***,** this portfolio only trades on days where surprises are expected to occur and does not substitute non-trading days by shifting the capital to a market portfolio. The coefficient size of the alpha (intercept) progressively increases with each threshold increase, although the same cannot be said for the significance scores, largely as a result of smaller samples at the higher thresholds and higher weightings on alternative coefficients.

Table A17: 5-Factor Model Coefficients and Significance for a Large Firm Surprise and Market Portfolio Strategy

Threshold	Surprise	Intercept	MktRf	SMB	HML	RMW	CMA
5%+	Positive	0.044	1.005	0.067	0.080	-0.087	-0.090
		(1.553)	(36.065)	(1.233)	(1.511)	(-0.899)	(-0.861)
	Negative	-0.002	1.010	0.083	0.012	0.075	0.181
		(-0.149)	(77.873)	(3.262)	(0.47)	(1.647)	(3.692)
10%+	Positive	0.032	1.007	0.094	0.130	-0.059	-0.040
		(1.221)	(40.146)	(1.914)	(2.703)	(-0.671)	(-0.424)
	Negative	-0.005	1.013	0.069	0.035	0.047	0.146
		(-0.395)	(81.976)	(2.841)	(1.46)	(1.086)	(3.122)
15%+	Positive	0.037	0.998	0.002	0.095	-0.026	-0.032
		(1.979)	(56.22)	(0.051)	(2.807)	(-0.421)	(-0.472)
	Negative	-0.018	1.009	0.098	0.048	0.069	0.125
		(-1.189)	(68.706)	(3.413)	(1.694)	(1.341)	(2.258)

MktRf is the difference between R_{Mt}, the value-weighted return of the market portfolio, and R_{Ft} the risk-free rate. SMB_t, HML_t, RMW_t and CMA_t are the respective differences between diversified portfolios of small stocks and big stocks, high and low B/M stocks, robust and weak profitability stocks, and low and high investment stocks. The following table tracks the performance of a strategy formed over a sample period consisting of 2994 trading days. All strategies showed large significance for the market coefficient, the reason being that a significant part of the portfolio days are formed by taking a position in the market. For all the long strategies, HML showed positive significance apart for the 5% long thresholds. For all the short strategies, SMB and CMA showed positive significance. Unlike *Table A16*, each strategy has the same number of portfolio days; the reason is that the market substitutes for the days where no expected earnings surprises are predicted to occur. But the same portfolio return as presented by *Table A16* is also included in this strategy. The alphas with this strategy are somewhat low, mostly as a result of the returns fitting strongly on value firms; however, the plotted cumulative returns still show economic significance over the sample period. Further tests showed improved significance when stop-loss triggers were incorporated into the strategy.

Table A18: Full Feature-Mapper Combination for top Signal Processed Variables

Base Feature	**Mapper**	**Parameters**	**Full Name**
Chaikin	mean_abs_change_quantiles	qh_1.0__ql_0.4	Chaikin__mean_abs_change_quantiles__qh_1.0__ql_0.4
Donchian_10	friedrich_coefficients	m_3__r_30__coeff_2	Donchian_10__friedrich_coefficients__m_3__r_30__coeff_2
MA_3	max_langevin_fixed_point	m_3__r_30	MA_3__max_langevin_fixed_point__m_3__r_30
Trix_10	Autocorrelation	lag_8	Trix_10__autocorrelation__lag_8
KelChU_5	max_langevin_fixed_point	m_3__r_30	KelChU_5__max_langevin_fixed_point__m_3__r_30
BollingerB_5	mean_abs_change_quantiles	qh_0.8__ql_0.6	BollingerB_5__mean_abs_change_quantiles__qh_0.8__ql_0.6
Momentum_10	friedrich_coefficients	m_3__r_30__coeff_1	Momentum_10__friedrich_coefficients__m_3__r_30__coeff_1
Bollinger%b_5	friedrich_coefficients	m_3__r_30__coeff_3	Bollinger%b_5__friedrich_coefficients__m_3__r_30__coeff_3
RSI_5	friedrich_coefficients	m_3__r_30__coeff_2	RSI_5__friedrich_coefficients__m_3__r_30__coeff_2
Donchian_8	autocorrelation	lag_4	Donchian_8__autocorrelation__lag_4
MA_3	max_langevin_fixed_point	m_3__r_30	MA_3__max_langevin_fixed_point__m_3__r_30
Chaikin	mean_abs_change_quantiles	__qh_1.0__ql_0.4	Chaikin__mean_abs_change_quantiles__qh_1.0__ql...
TSI_3_2	cwt_coefficients	widths_(2, 5, 10, 2)	TSI_3_2__cwt_coefficients__widths_(2, 5, 10, 2)
EMA_3	cwt_coefficients	EMA_3__cwt_coefficients__widths_(2, 5, 10, 20)	EMA_3__cwt_coefficients__widths_(2, 5, 10, 20)
ADX_4_3	fft_coefficient	coeff_6	ADX_4_3__fft_coefficient__coeff_6

Bibliography*

Abarbanell, J. S., & Lehavy, R. Differences in commercial database reported earnings: Implications for empirical research. *SSRN Electronic Journal,* doi:10.2139/ssrn.228918

Abarbanell, J., & Lehavy, R. (2003). Biased forecasts or biased earnings? the role of reported earnings in explaining apparent bias and over/underreaction in analysts' earnings forecasts. *Journal of Accounting and Economics, 36*(1), 105-146.

Almamy, J., Aston, J., & Ngwa, L. N. (2016). An evaluation of Altman's Z-score using cash flow ratio to predict corporate failure amid the recent financial crisis: Evidence from the UK. *Journal of Corporate Finance, 36*, 278-285.

Altman, E. I. (1968). Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The Journal of Finance, 23*(4), 589-609.

Altman, E. I. (1984). A further empirical investigation of the bankruptcy cost question. *The Journal of Finance, 39*(4), 1067-1089.

Altman, E. I. (2002). *Bankruptcy, credit risk, and high yield junk bonds.* Wiley-Blackwell.

Altman, E. I., Marco, G., & Varetto, F. (1994). Corporate distress diagnosis: Comparisons using linear discriminant analysis and neural networks (the Italian experience). *Journal of Banking & Finance, 18*(3), 505-529.

Anderson, A., Kleinberg, J., & Mullainathan, S. (2017). Assessing human error against a benchmark of perfection. *ACM Transactions on Knowledge Discovery from Data (TKDD), 11*(4), 45.

Anilowski, C., Feng, M., & Skinner, D. J. (2007). Does earnings guidance affect market returns? the nature and information content of aggregate earnings guidance. *Journal of Accounting and Economics, 44*(1), 36-63.

Antonelli, C. (1989). A failure-inducement model of research and development expenditure: Italian evidence from the early 1980s. *Journal of Economic Behavior & Organization, 12*(2), 159-180.

Aziz, A., Emanuel, D. C., & Lawson, G. H. (1988). Bankruptcy prediction-an investigation of cash flow based models. *Journal of Management Studies, 25*(5), 419-437.

Bagheri, A., Peyhani, H. M., & Akbari, M. (2014). Financial forecasting using ANFIS networks with quantum-behaved particle swarm optimization. *Expert Systems with Applications, 41*(14), 6235-6250.

Bagnoli, M., Beneish, M. D., & Watts, S. G. (1999). Whisper forecasts of quarterly earnings per share. *Journal of Accounting and Economics, 28*(1), 27-50.

Baird, D. G., & Morrison, E. R. (2011). Dodd-frank for bankruptcy lawyers. *Am.Bankr.Inst.L.Rev., 19*, 287.

Baird, J. (2017). Renaissance technologies: Generating alpha without wall street veterans or MBAs. Retrieved from https://digit.hbs.org/submission/renaissance-technologies-generating-alpha-without-wall-street-veterans-or-mbas/

Ball, R., & Brown, P. (1968). An empirical evaluation of accounting income numbers. *Journal of Accounting Research, 6*(2), 159-178. Retrieved from https://search.proquest.com/docview/1420660825

Barber, B., Lehavy, R., McNichols, M., & Trueman, B. (2001). Can investors profit from the prophets? Security analyst recommendations and stock returns. *The Journal of Finance, 56*(2), 531-563.

Barboza, F., Kimura, H., & Altman, E. (2017). Machine learning models and bankruptcy prediction. *Expert Systems with Applications, 83*, 405-417.

Barniv, R., Agarwal, A., & Leach, R. (2002). Predicting bankruptcy resolution. *Journal of Business Finance & Accounting, 29*(3-4), 497-520.

Barron, O. E., Harris, D. G., & Stanford, M. (2005). Evidence that investors trade on private event-period information around earnings announcements. *The Accounting Review, 80*(2), 403-421.

Barth, M. E., & Hutton, A. P. (2004). Analyst earnings forecast revisions and the pricing of accruals. *Review of Accounting Studies, 9*(1), 59-96.

Bartov, E., Givoly, D., & Hayn, C. (2002). The rewards to meeting or beating earnings expectations. *Journal of Accounting and Economics, 33*(2), 173-204.

Bauer, J., & Agarwal, V. (2014). Are hazard models superior to traditional bankruptcy prediction approaches? A comprehensive test. *Journal of Banking & Finance, 40*, 432-442.

Beaver, W. H. (1968). The information content of annual earnings announcements. *Journal of Accounting Research, 6*, 67-92.

Beaver, W. H., McNichols, M. F., & Rhie, J. (2005). Have financial statements become less informative? evidence from the ability of financial ratios to predict bankruptcy. *Review of Accounting Studies, 10*(1), 93-122.

Behn, B. K., Choi, J., & Kang, T. (2008). Audit quality and properties of analyst earnings forecasts. *The Accounting Review, 83*(2), 327-349.

Behr, A., & Weinblat, J. (2017). Default patterns in seven EU countries: A Random Forest approach. *International Journal of the Economics of Business, 24*(2), 181-222.

Berger, P. G., Ham, C. C., & Kaplan, Z. (2016). Do analysts say anything about earnings without revising their earnings forecasts?

Bergmeir, C., & Bentez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences, 191,* 192-213.

Bernard, V. L., & Thomas, J. K. (1990). Evidence that stock prices do not fully reflect the implications of current earnings for future earnings. *Journal of Accounting and Economics, 13*(4), 305-340.

Bernstein, S., & Sheen, A. (2016). The operational consequences of private equity buyouts: Evidence from the restaurant industry. *The Review of Financial Studies, 29*(9), 2387-2418.

Bhattacharya, N., Sheikh, A., & Thiagarajan, S. R. (2006). Does the market listen to whispers? *The Journal of Investing, 15*(1), 16-24.

Bialik, C. (2017). *Local economic outlook.* Yelp. Retrieved from https://www.yelpblog.com/wp-content/uploads/2017/10/Yelp-Local-Economic-Outlook-Report_October-2017.pdf

Bibler, G. A. (1987). The status of unaccrued tort claims in Chapter 11 bankruptcy proceedings. *Am.Bankr.LJ, 61,* 145.

Booth, A., Gerding, E., & Mcgroarty, F. (2014). Automated trading with performance weighted random forests and seasonality. *Expert Systems with Applications, 41*(8), 3651-3661.

Bouchaud, J., & Cont, R. (1998). A Langevin approach to stock market fluctuations and crashes. *The European Physical Journal B-Condensed Matter and Complex Systems, 6*(4), 543-550.

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7), 1145-1159.

Bradshaw, M. T., Drake, M. S., Myers, J. N., & Myers, L. A. (2012). A re-examination of analysts' superiority over time-series forecasts of annual earnings. *Review of Accounting Studies, 17*(4), 944-968.

Branson, B. C., Lorek, K. S., & Pagach, D. P. (1995). Evidence on the superiority of analysts quarterly earnings forecasts for small capitalization firms. *Decision Sciences, 26*(2), 243-263.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees* CRC press.

Bris, A., Welch, I., & Zhu, N. (2004). The costs of bankruptcy Chapter 7 cash auctions vs. Chapter 11 bargaining.

Bris, A., Welch, I., & Zhu, N. (2006). The costs of bankruptcy: Chapter 7 liquidation versus chapter 11 reorganization. *The Journal of Finance, 61*(3), 1253-1303.

Brown, L. D. (2001). A temporal analysis of earnings surprises: Profits versus losses. *Journal of Accounting Research, 39*(2), 221-241.

Brown, L. D., Han, J. C., Keon Jr, E. F., & Quinn, W. H. (1996). Predicting analysts' earnings surprise. *The Journal of Investing, 5*(1), 17-23.

Burgstahler, D., & Dichev, I. (1997). Earnings management to avoid earnings decreases and losses. *Journal of Accounting and Economics, 24*(1), 99-126.

Burgstahler, D., & Eames, M. (2006). Management of earnings and analysts' forecasts to achieve zero and small positive earnings surprises. *Journal of Business Finance & Accounting, 33*(5-6), 633-652.

Callen, J. L., Kwan, C. C., Yip, P. C., & Yuan, Y. (1996). Neural network forecasting of quarterly accounting earnings. *International Journal of Forecasting, 12*(4), 475-482.

Chan, L. K., Karceski, J., & Lakonishok, J. (2007). Analysts' conflicts of interest and biases in earnings forecasts. *Journal of Financial and Quantitative Analysis, 42*(4), 893-913.

Chandra, D. K., Ravi, V., & Bose, I. (2009). Failure prediction of dotcom companies using hybrid intelligent techniques. *Expert Systems with Applications, 36*(3), 4830-4837.

Chaudhuri, A., & De, K. (2011). Fuzzy support vector machine for bankruptcy prediction. *Applied Soft Computing, 11*(2), 2472-2486.

Chen, J., Chen, W., Huang, C., Huang, S., & Chen, A. (2016). Financial time-series data analysis using deep convolutional neural networks. Paper presented at the *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, 87-92.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Paper presented at the *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, 785-794.

Chudson, W. A. (1945). The pattern of corporate financial structure: A cross-section view of manufacturing, mining, trade, and construction, 1937. *NBER Books*.

Claus, J., & Thomas, J. (2001). Equity premia as low as three percent? evidence from analysts' earnings forecasts for domestic and international stock markets. *The Journal of Finance, 56*(5), 1629-1666.

Cornelius Casey, & Norman Bartczak. (1985). Using operating cash flow data to predict financial distress: Some extensions. *Journal of Accounting Research, 23*(1), 384-401. doi:10.2307/2490926

Dambolena, I. G., & Khoury, S. J. (1980). Ratio stability and corporate failure. *The Journal of Finance, 35*(4), 1017-1026.

David R. Anderson, & Kenneth P. Burnham. (2002). Avoiding pitfalls when using information-theoretic methods. *The Journal of Wildlife Management, 66*(3), 912-918. doi:10.2307/3803155

Deligianni, D., & Kotsiantis, S. (2012). Forecasting corporate bankruptcy with an ensemble of classifiers. Paper presented at the *Hellenic Conference on Artificial Intelligence,* 65-72.

Dhar, V., & Chou, D. (2001). A comparison of nonlinear methods for predicting earnings surprises and returns. *IEEE Transactions on Neural Networks, 12*(4), 907-921.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning, 40*(2), 139-157.

DiPietro, R. B., Parsa, H. G., & Gregory, A. (2011). Restaurant QSC inspections and financial performance: An empirical investigation. *International Journal of Contemporary Hospitality Management, 23*(7), 982-999.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM, 55*(10), 78-87.

du Jardin, P. (2017). Dynamics of firm financial evolution and bankruptcy prediction. *Expert Systems with Applications, 75*, 25-43.

Duffie, D., Pedersen, L. H., & Singleton, K. J. (2003). Modeling sovereign yield spreads: A case study of russian debt. *The Journal of Finance, 58*(1), 119-159.

Easterwood, J. C., & Nutt, S. R. (1999). Inefficiency in analysts' earnings forecasts: Systematic misreaction or systematic optimism? *The Journal of Finance, 54*(5), 1777-1797.

Eliazar, I. (2017). Lindy's law. *Physica A: Statistical Mechanics and its Applications, 486*, 797-805.

Elith, J., Leathwick, J. R., & Hastie, T. (2008). A working guide to boosted regression trees. *Journal of Animal Ecology, 77*(4), 802-813.

Fama, E. F., & French, K. R. (2015). A five-factor asset pricing model. *Journal of Financial Economics, 116*(1), 1-22.

Fama, E. F., & Blume, M. E. (1966). Filter rules and stock-market trading. *The Journal of Business, 39*(1), 226-241.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters*, 27(8), 861-874.

Ferri, C., Flach, P., & Hernndez-Orallo, J. (2002). Learning decision trees using the area under the ROC curve. Paper presented at the *Icml, 2*, 139-146.

Fidler, F., Geoff, C., Mark, B., & Neil, T. (2004). Statistical reform in medicine, psychology and ecology. *The Journal of Socio-Economics, 33*(5), 615-630.

FitzPatrick, P. J. (1932). *A comparison of the ratios of successful industrial enterprises with those of failed companies*.

Foster, G. (1973). Stock market reaction to estimates of earnings per share by company officials. *Journal of accounting research, 11*(1), 25-37. Retrieved from http://www.econis.eu/PPNSET?PPN=39273009X

Foster, G. (1986). *Financial statement analysis, 2/e* Pearson Education.

Franks, J. R., & Torous, W. N. (1989). An empirical investigation of US firms in reorganization. *The Journal of Finance, 44*(3), 747-769.

Fried, D., & Givoly, D. (1982). Financial analysts' forecasts of earnings: A better surrogate for market expectations. *Journal of Accounting and Economics, 4*(2), 85-107.

Friedman, J. H., & Meulman, J. J. (2003). Multiple additive regression trees with application in epidemiology. *Statistics in Medicine, 22*(9), 1365-1381.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning* Springer series in statistics New York.

Friedrich, R., Peinke, J., & Renner, C. (2000). How to quantify deterministic and random influences on the statistics of the foreign exchange market. *Physical Review Letters, 84*(22), 5224.

George Foster, Chris Olsen, & Terry Shevlin. (1984). Earnings releases, anomalies, and the behavior of security returns. *The Accounting Review, 59*(4), 574-603. Retrieved from https://www.jstor.org/stable/247321

Givoly, D., & Lakonishok, J. (1984). The quality of analysts' forecasts of earnings. *Financial Analysts Journal, 40*(5), 40-47.

Glaeser, E., Kim, H., & Luca, M. (2017). Nowcasting the Local Economy: Using Yelp Data to Measure Economic Activity (No. 24010). *National Bureau of Economic Research, Inc.*

Gu, S., Kelly, B. T., & Xiu, D.Empirical asset pricing via machine learning. *SSRN Electronic Journal,* doi:10.2139/ssrn.3159577

Graham, B., & Dodd, D. L. (1934). *Security analysis: Principles and technique* McGraw-Hill.

Graham, J. R., Harvey, C. R., & Rajgopal, S. (2005). The economic implications of corporate financial reporting. *Journal of Accounting and Economics, 40*(1), 3-73.

Grossman, S. J., & Zhou, Z. (1993). Optimal investment strategies for controlling drawdowns. *Mathematical Finance, 3*(3), 241-276.

Han, Y., & Zhou, G.Taming momentum crashes: A simple stop-loss strategy. *SSRN Electronic Journal,* doi:10.2139/ssrn.2407199

Hand, D. J. (2009). Measuring classifier performance: A coherent alternative to the area under the ROC curve. *Machine Learning, 77*(1), 103-123.

Hardiman, A. (1985). Toxic torts and chapter 11 reorganization: The problem of future claims. *Vand.L.Rev., 38,* 1369-2013.

Hart, O. (2000). Different approaches to bankruptcy (No. w7921). *National Bureau of Economic Research*.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Overview of supervised learning. *The elements of statistical learning* (pp. 9-41). New York: Springer.

Hellstrom, T., & Holmstrom, K. (1998). Predicting the stock market. *Unpublished Thesis, Malardalen University, Department of Mathematics and Physics, Vasteras, Sweden,*

Hillegeist, S. A., Keating, E. K., Cram, D. P., & Lundstedt, K. G. (2004). Assessing the probability of bankruptcy. *Review of Accounting Studies, 9*(1), 5-34.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2*(5), 359-366.

Hu, Y., Feng, B., Zhang, X., Ngai, E., & Liu, M. (2015). Stock trading rule discovery with an evolutionary trend following model. *Expert Systems with Applications, 42*(1), 212-222.

Huang, T., Leu, Y., & Pan, W. (2016). Constructing ZSCORE-based financial crisis warning models using fruit fly optimization algorithm and general regression neural network. *Kybernetes, 45*(4), 650-665.

Hutson, J. K. (1983). TRIX-triple exponential smoothing oscillator. *Technical Analysis of Stocks and Commodities, ,* 105-108.

Jerome H. Friedman. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics, 29*(5), 1189-1232. doi:10.2307/2699986

Johnson, W. B., & Zhao, R. (2012). Contrarian share price reactions to earnings surprises. *Journal of Accounting, Auditing & Finance, 27*(2), 236-266.

Jones, C. P., & Litzenberger, R. H. (1970). Quarterly earnings reports and intermediate stock price trends. *The Journal of Finance, 25*(1), 143-148.

Jones, S. (2017). Corporate bankruptcy prediction: A high dimensional analysis. *Review of Accounting Studies, 22*(3), 1366-1422.

Jones, S., & Hensher, D. A. (2004). Predicting firm financial distress: A mixed logit model. *The Accounting Review, 79*(4), 1011-1038.

Jones, S., Johnstone, D., & Wilson, R. (2017). Predicting corporate bankruptcy: An evaluation of alternative statistical frameworks. *Journal of Business Finance & Accounting, 44*(1-2), 3-34.

Joret, G., Micek, P., Milans, K. G., Trotter, W. T., Walczak, B., & Wang, R. (2016). Tree-width and dimension. *Combinatorica, 36*(4), 431-450.

Jung, J., Concannon, C., Shroff, R., Goel, S., & Goldstein, D. G. (2017). *Simple rules for complex decisions*

Kaminski, K. M., & Lo, A. W. (2014). *When do stop-loss rules stop losses?* doi://doi-org.ezproxy.auckland.ac.nz/10.1016/j.finmar.2013.07.001

Kang, J. S., Kuznetsova, P., Luca, M., & Choi, Y. (2013). Where not to eat? improving public policy by predicting hygiene inspections using online reviews. Paper presented at the *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1443-1448.

Karas, M., & Reznakova, M. (2017). Predicting the bankruptcy of construction companies: A CART-based model. *Engineering Economics, 28*(2), 145-154. doi:10.5755/j01.ee.28.2.16353

Kasparov, G. (2010). The chess master and the computer. *The New York Review of Books, 57*(2), 16-19.

Kasznik, R., & McNichols, M. F. (2002). Does meeting earnings expectations matter? evidence from analyst forecast revisions and share prices. *Journal of Accounting Research, 40*(3), 727-759.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., . . . Liu, T. (2017). LightGBM: A highly efficient gradient boosting decision tree. Paper presented at the *Advances in Neural Information Processing Systems*, 3149-3157.

Kim, K. (2003). Financial time series forecasting using support vector machines. *Neurocomputing, 55*(1), 307-319.

Kim, M., & Kang, D. (2010). Ensemble with neural networks for bankruptcy prediction. *Expert Systems with Applications, 37*(4), 3373-3379.

Kim, S. Y., & Upneja, A. (2014). Predicting restaurant financial distress using decision tree and AdaBoosted decision tree models. *Economic Modelling, 36*, 354-362.

Kinney, W., Burgstahler, D., & Martin, R. (2002). Earnings surprise "materiality" as measured by stock returns. *Journal of Accounting Research, 40*(5), 1297-1329.

Kirt C. Butler, & Larry H. P. Lang. (1991). The forecast accuracy of individual analysts: Evidence of systematic optimism and pessimism. *Journal of Accounting Research, 29*(1), 150-156.

Kleinberg, J., Lakkaraju, H., Leskovec, J., Ludwig, J., & Mullainathan, S. (2018). Human decisions and machine predictions. *The Quarterly Journal of Economics, 133*(1), 237-293.

Kogan, S., Levin, D., Routledge, B. R., Sagi, J. S., & Smith, N. A. (2009). Predicting risk from financial reports with regression. Paper presented at the *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 272-280.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. Paper presented at the *Ijcai,, 14*(2) 1137-1145.

Kreeger, J. C., Parsa, H. G., Smith, S. J., & Kubickova, M. (2018). Calendar effect and the role of seasonality in consumer comment behavior: A longitudinal study in the restaurant industry. *Journal of Foodservice Business Research, 21*(3), 342-357.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Paper presented at the *Advances in Neural Information Processing Systems*, 1097-1105.

Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling* Springer.

Kursa, M. B., & Rudnicki, W. R. (2010). Feature selection with the Boruta package. *J Stat Softw, 36*(11), 1-13.

Laney, D. B. (2017). *Infonomics: How to monetize, manage, and measure information as an asset for competitive advantage.* Routledge.

Langevin, P. (1908). Sur la thorie du mouvement brownien. *CR Acad.Sci.Paris, 146*(530-533), 530.

Latane, H. A., & Jones, C. P. (1977). Standardized unexpected earnings—A progress report. *The Journal of Finance, 32*(5), 1457-1465.

Lawrence D. Brown, Gordon D. Richardson, & Steven J. Schwager. (1987). An information interpretation of financial analyst superiority in forecasting earnings. *Journal of Accounting Research, 25*(1), 49-67. doi:10.2307/2491258

Leathwick, J. R., Elith, J., Francis, M. P., Hastie, T., & Taylor, P. (2006). Variation in demersal fish species richness in the oceans surrounding New Zealand: An analysis using boosted regression trees. *Marine Ecology Progress Series, 321*, 267-281.

Lhabitant, F. (2011). *Handbook of hedge funds* John Wiley & Sons.

Li, J. (2012). Prediction of corporate bankruptcy from 2008 through 2011. *Journal of Accounting and Finance, 12*(1), 31-41.

Li, K. (1999). Bayesian analysis of duration models: An application to chapter 11 bankruptcy. *Economics Letters, 63*(3), 305-312.

Lian, J., Zhang, F., Xie, X., & Sun, G. (2017). Restaurant survival analysis with heterogeneous information. Paper presented at the *Proceedings of the 26th International Conference on World Wide Web Companion*, 993-1002.

Liang, D., Lu, C., Tsai, C., & Shih, G. (2016). Financial ratios and corporate governance indicators in bankruptcy prediction: A comprehensive study. *European Journal of Operational Research, 252*(2), 561-572.

Liu, H., & Motoda, H. (2012). *Feature selection for knowledge discovery and data mining* Springer Science & Business Media.

LoPucki, L. M., & Doherty, J. W. (2008). Professional overcharging in large bankruptcy reorganization cases. *Journal of Empirical Legal Studies, 5*(4), 983-1017.

LoPucki, L. M., & Doherty, J. W. (2015). Bankruptcy survival. *UCLA L.Rev., 62*, 969.

LoPucki, L. M., & Kalin, S. D. (2001). Failure of public company bankruptcies in Delaware and New York: Empirical evidence of a race to the bottom. *Vand.L.Rev., 54*, 231.

Luca, M. (2011). *Reviews, reputation, and revenue: The case of yelp.com.* Retrieved from http://econpapers.repec.org/paper/hbswpaper/12-016.htm

Luca, M., & Zervas, G. (2016). Fake it till you make it: Reputation, competition, and Yelp review fraud. *Management Science, 62*(12), 3412-3427.

Lundberg, S. M., & Lee, S. (2017). A unified approach to interpreting model predictions. Paper presented at the *Advances in Neural Information Processing Systems*, 4768-4777.

Luo, T., & Stark, P. B. (2015). Nine out of 10 restaurants fail? check, please. *Significance, 12*(2), 25-29.

Lys, T., & Sohn, S. (1990). The association between revisions of financial analysts' earnings forecasts and security-price changes. *Journal of Accounting and Economics, 13*(4), 341-363.

Lys, T., & Soo, L. G. (1995). Analysts' forecast precision as a response to competition. *Journal of Accounting, Auditing & Finance, 10*(4), 751-765.

MacGregor, N., & Lo, D. (2015). The effect of chain size and customer type on company survival: Evidence from multi-unit restaurants. *Academy of Management Proceedings, 2015*(1), 17991.

MacKay, D. J. (1992). The evidence framework applied to classification networks. *Neural Computation, 4*(5), 720-736.

Matras, K. (2016). Zacks investment research - video: Zacks earnings ESP (expected surprise prediction). Retrieved from https://search.proquest.com/docview/1838754510

Mejia, J., Mankad, S., & Gopal, A. (2015). More than just words: Latent semantic analysis, online reviews and restaurant closure. *Academy of Management Proceedings, 2015*(1), 13912. doi:10.5465/AMBPP.2015.13912abstract

Merwin, C. L. 1. (1942). *Financing small corporations in five manufacturing industries, 1926-36*. United States: Retrieved from http://catalog.hathitrust.org/Record/001128380

Meyer, P. A. (1967). Price discrimination, regional loan rates, and the structure of the banking industry. *The Journal of Finance, 22*(1), 37-48.

Mian, S. (2013). *Zacks earnings ESP*. Zacks. Retrieved from https://staticzacks.net/pdf/EESPreport1V3.pdf

Michael J. Gombola, & J. Edward Ketz. (1983). A note on cash flow and classification patterns of financial ratios. *The Accounting Review, 58*(1), 105-114. Retrieved from https://www.jstor.org/stable/246645

Montas, E., Quevedo, J. R., Prieto, M. M., & Menndez, C. O. (2002). Forecasting time series combining machine learning and Box-Jenkins time series. Paper presented at the *Ibero-American Conference on Artificial Intelligence*, 491-499.

Mselmi, N., Lahiani, A., & Hamza, T. (2017). Financial distress prediction: The case of French small and medium-sized firms. *International Review of Financial Analysis, 50*, 67-80.

Mukherjee, S., Golland, P., & Panchenko, D. (2003). Permutation tests for classification, 11-21

Mullainathan, S., & Spiess, J. (2017). Machine learning: An applied econometric approach. *Journal of Economic Perspectives, 31*(2), 87-106.

National Restaurant Association. (2017). *Restaurant industry outlook*. Retrieved from http://www.restaurant.org/Downloads/PDFs/News-Research/2017_Restaurant_outlook_summary-FINAL.pdf

Neves, J. C., & Vieira, A. (2006). Improving bankruptcy prediction with hidden layer learning vector quantization. *European Accounting Review, 15*(2), 253-271.

O'Brien, P. C. (1998). Discussion of international variation in accounting measurement rules and analysts' earnings forecast errors. *Journal of Business Finance & Accounting, 25*(9-10), 1249-1254.

Ohlson, J. A. (2009). Financial ratios and the probabilistic prediction of bankruptcy. *Financial Accounting and Investment Management, 18(1),* 363-385. Retrieved from http://www.econis.eu/PPNSET?PPN=603823629

Olson, D. L., Delen, D., & Meng, Y. (2012). Comparative analysis of data mining methods for bankruptcy prediction. *Decision Support Systems, 52*(2), 464-473.

Parsa, H. G., Self, J. T., Njite, D., & King, T. (2005). Why restaurants fail. *Cornell Hotel and Restaurant Administration Quarterly, 46*(3), 304-322.

Parsa, H. G., Self, J., Sydnor-Busso, S., & Yoon, H. J. (2011). Why restaurants fail? part II-the impact of affiliation, location, and size on restaurant failures: Results from a survival analysis. *Journal of Foodservice Business Research, 14*(4), 360-379.

Parsa, H. G., van der Rest, Jean-Pierre I, Smith, S. R., Parsa, R. A., & Bujisic, M. (2015). Why restaurants fail? part IV: The relationship between restaurant failures and demographic factors. *Cornell Hospitality Quarterly, 56*(1), 80-90.

Patel, C. (1980). *Technical trading systems for commodities and stocks.* Trading Systems Research.

Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications, 42*(1), 259-268.

Payne, J. L., & Thomas, W. B. (2003). The implications of using stock-split adjusted I/B/E/S data in empirical research. *The Accounting Review, 78*(4), 1049-1067.

Pervan, I., Pervan, M., & Vukoja, B. (2011). Prediction of company bankruptcy using statistical techniques–Case of croatia. *Croatian Operational Research Review, 2*(1), 158-167.

Powers, D. M. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies, 2*, 37-63.

Provost, F., & Kohavi, R. (1998). Glossary of terms. *Journal of Machine Learning, 30*(2-3), 271-274.

Puga, D. (1999). The rise and fall of regional inequalities. *European Economic Review, 43*(2), 303-334.

Purdy, D., Chen, L., & Sumers, T. R. (2017). *Cascaded Boosted Predictive Models.*

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*(1), 81-106.

Raileanu, L. E., & Stoffel, K. (2004). Theoretical comparison between the Gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence, 41*(1), 77-93.

Ramnath, S., Rock, S., & Shane, P. (2008). The financial analyst forecasting literature: A taxonomy with suggestions for further research. *International Journal of Forecasting, 24*(1), 34-75.

Ramsey, J. B., & Zhang, Z. (1997). The analysis of foreign exchange data using waveform dictionaries. *Journal of Empirical Finance, 4*(4), 341-372.

Robert N. Freeman, & Senyo Y. Tse. (1992). A nonlinear model of security price responses to unexpected earnings. *Journal of Accounting Research, 30*(2), 185-209. doi:10.2307/2491123

Rose, P. S., & Giroux, G. A. (1984). Predicting corporate bankruptcy: An analytical and empirical evaluation. *Review of Financial Economics, 19*(2), 1.

Sable, R. G., Roeschenthaler, M. J., & Blanks, D. F. (2006). When the 363 sale is the best route. *J.Bankr.L.& Prac., 15*, 2.

Schaaf, A. H. (1966). Regional differences in mortgage financing costs. *The Journal of Finance, 21*(1), 85-94.

Schapire, R. E., & Freund, Y. (2012). *Boosting: Foundations and algorithms* MIT press.

Scott, J. (1981). The probability of bankruptcy: A comparison of empirical predictions and theoretical models. *Journal of Banking & Finance, 5*(3), 317-344.

Sharif Razavian, A., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). CNN variables off-the-shelf: An astounding baseline for recognition. Paper presented at the *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 806-813.

Shumway, T. (2001). Forecasting bankruptcy more accurately: A simple hazard model. *The Journal of Business, 74*(1), 101-124.

Siegert, S., Friedrich, R., & Peinke, J. (1998). Analysis of data sets of stochastic systems. *Physics Letters A, 243*(5-6), 275-280.

Soffer, L. C., Thiagarajan, S. R., & Walther, B. R. (2000). Earnings preannouncement strategies. *Review of Accounting Studies, 5*(1), 5-26.

Somnath Das, Carolyn B. Levine, & K. Sivaramakrishnan. (1998). Earnings predictability and bias in analysts' earnings forecasts. *The Accounting Review, 73*(2), 277-294. Retrieved from https://www.jstor.org/stable/248469

Stauth, J. (2013). Trading (and predicting) earnings surprises. *Thomson Reuters*. Retrieved from https://blog.quantopian.com/wp-content/uploads/2013/05/Trading_Earnings_surprises-jess.pdf

Stehman, S. V. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment, 62*(1), 77-89.

Stickel, S. E. (1995). The anatomy of the performance of buy and sell recommendations. *Financial Analysts Journal, 51*(5), 25-39.

Sun, J., & Li, H. (2012). Financial distress prediction using support vector machines: Ensemble vs. individual. *Applied Soft Computing, 12*(8), 2254-2265.

Tan, C., Lee, L., & Pang, B. (2014). The effect of wording on message propagation: Topic- and author-controlled natural experiments on twitter. *arXiv Preprint arXiv:1405.1438,*

Taylor, D. C., & Aday, J. B. (2016). Consumer generated restaurant ratings: A preliminary look at OpenTable.com. *Journal of New Business Ideas and Trends, 14*(1), 14-23.

Teixeira, L. A., & De Oliveira, Adriano Lorena Inacio. (2010). A method for automatic stock trading combining technical analysis and nearest neighbor classification. *Expert Systems with Applications, 37*(10), 6885-6890.

Trueman, B. (1990). Theories of earnings-announcement timing. *Journal of Accounting and Economics, 13*(3), 285-301.

Volkov, A., Benoit, D. F., & Van den Poel, D. (2017). Incorporating sequential information in bankruptcy prediction with predictors based on markov for discrimination. *Decision Support Systems, 98*, 59-68.

W. S. Hopwood, J. C. Mckeown, & P. Newbold. (1982). The additional information content of quarterly earnings reports: Intertemporal disaggregation. *Journal of Accounting Research, 20*(2), 343-349. doi:10.2307/2490744

Waymire, G. (1986). Additional evidence on the accuracy of analyst forecasts before and after voluntary management earnings forecasts. *Accounting Review, 61*, 129-142.

Weiss, L. A., Bhandari, J. S., & Robins, R. (2000). An analysis of state-wide variation in bankruptcy rates in the united states. *Bankr.Dev.J., 17*, 407.

Whittingham, M. J., Stephens, P. A., Bradbury, R. B., & Freckleton, R. P. (2006). Why do we still use stepwise modelling in ecology and behaviour? *Journal of Animal Ecology, 75*(5), 1182-1189.

William A. Collins, & William S. Hopwood. (1980). A multivariate analysis of annual earnings forecasts generated from quarterly forecasts of financial analysts and univariate time-series models. *Journal of Accounting Research, 18*(2), 390-406. doi:10.2307/2490585

William H. Beaver. (1966). Financial ratios as predictors of failure. *Journal of Accounting Research, 4*, 71-111. doi:10.2307/2490171

William Kross, Byung Ro, & Douglas Schroeder. (1990). Earnings expectations: The analysts' information advantage. *The Accounting Review, 65*(2), 461-476. Retrieved from https://www.jstor.org/stable/247634

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Data mining: Practical machine learning tools and techniques, *Morgan Kaufmann*.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*(1), 67-82.

Womack, K. L. (1996). Do brokerage analysts' recommendations have investment value? *The Journal of Finance, 51*(1), 137-167.

Xiao, Y., Xiao, J., Lu, F., & Wang, S. (2013). Ensemble ANNs-PSO-GA approach for day-ahead stock E-exchange prices forecasting. *International Journal of Computational Intelligence Systems, 6*(1), 96-114.

Youn, H., & Gu, Z. (2010). Predict US restaurant firm failures: The artificial neural network model versus logistic regression model. *Tourism and Hospitality Research, 10*(3), 171-187.

Zhang, M., & Luo, L.Can user generated content predict restaurant survival: Deep learning of yelp photos and reviews. *SSRN Electronic Journal*, doi:10.2139/ssrn.3108288

Zhao, D., Huang, C., Wei, Y., Yu, F., Wang, M., & Chen, H. (2017). An effective computational model for bankruptcy prediction using kernel extreme learning machine approach. *Computational Economics, 49*(2), 325-341.

Zhou, L. (2013). Performance of corporate bankruptcy prediction models on imbalanced dataset: The effect of sampling methods. *Knowledge-Based Systems, 41,* 16-25.

Zięba, M., Tomczak, S. K., & Tomczak, J. M. (2016). Ensemble boosted trees with synthetic variables generation in application to bankruptcy prediction. *Expert Systems with Applications, 58,* 93-101.

Zmijewski, M. E. (1984). Methodological issues related to the estimation of financial distress prediction models. *Journal of Accounting Research, 22,* 59-82.

*All references as part of a three-chapter thesis.