Capturing dynamics of post-earnings-announcement drift using genetic algorithm-optimized supervised learnings

Zhengxin Joseph Ye[1]
Department of Computing, Imperial College

(Dated: 13 November 2019)

Post-Earnings-Announcement Drift (PEAD) is a stock market phenomenon when a stock's cumulative abnormal returns have a tendency to drift in the direction of an earnings surprise in the near term following an earnings announcement. Although it is one of the most studied stock market anomalies and its existence is well understood, the current literature is limited in explaining this phenomenon by a small number of factors using simpler regression methods. In this paper we aim to use supervised learning models instead to try and capture the PEAD dynamics of groups of stocks using a wider range of both fundamental and technical factors. We test a deep neural network (DNN), the Extreme Gradient Boosting model (XGBoost) as well as support vector machines (SVM) with different kernels using a long list of carefully prepared and engineered input features including quarterly earning announcement data from 1106 companies from the Russell 1000 index between 1997 and 2018. Our experiments show that XGBoost performs better in predicting the PEAD direction of out-of-sample test stocks than DNN and SVM in a range of experiments. More significantly, since we put the focus of our experiments and analysis at the portfolio level, for the first time in the literature we've produced post-earnings stock return predictions that are strongly tied with actual return of portfolios consisting of out-of-sample stocks. Our methods and results consistently show that we can use our model's predictability following company earnings releases to allocate out-of-sample stocks to form portfolios with high positive returns to long and portfolios with low negative returns to short in the process of constructing market neutral strategies.

I. INTRODUCTION

The stock market is characterized by nonlinearities, discontinuities, and multi-polynomial components because it continuously interacts with many factors such as individual company's news, political events, macro economic conditions, and general supply and demand, etc [1]. The non-stationary nature of the stock market is supported by a widely accepted but still hotly contested economic theory *Efficient Market Hypothesis* which states that asset prices fully reflect all available information and the market only moves by reacting to new information. Such a theory implies that the stock market behaves like a martingale and knowledge of all past prices is not informative regarding the expectation of future prices.

Ball and Brown [2] were the first to note that after earnings are announced, estimated cumulative abnormal returns continue to drift up for firms that are perceived to have reported good financial results for the preceding quarter and drift down for firms whose results have turned out worse than the market had expected. The discovery of Post Earnings Announcement Drift, which is a violation of semi-strong Efficient Market Hypothesis, seems to suggest that while stock markets are generally efficient, there may be information leakages around the announcement dates, coupled with post-earnings drift, resulting in price movement anomalies. It also seems to suggest that past stock price information or other past economic or financial information can potentially be used to predict price movement following a significant economic event such as an earnings announcement.

We have noticed that a lot of researches on PEAD came out in the late 1980s and 1990s. Fama and French [3] shows that average stock returns co-vary with three factors, namely, the market risk factor, the book-to-market factor, and the size factor. Bhushan suggests that the existence of sophisticated and unsophisticated investors, transaction costs and economies of scale in managing money can explain the market's delayed response to earnings [4]. Nearly all previous research pooled companies with negative and positive earnings surprises when measuring the effect of earnings surprises on abnormal returns and regress the absolute value of earnings surprise as well as other factors against the absolute value of abnormal return [5]. However, we believe that stock markets don't just react symmetrically to negative and positive earnings surprises and there are a lot more factors in play that drive the near term risk adjusted returns of a stock following an earnings release.

Rather than trying to analyse the link between PEAD and more economic and accounting factors as commonly seen in the literature, by using machine learning models we manage to leap straight to the more important goal of predicting the direction of PEAD. In this process we've overcome a number of constraints commonly seen in previous researches: we are including a much wider range of factors including both fundamental and technical/momentum factors; we achieve a higher level of generality without having to pre-group companies by the value of their earnings surprises or other attributes prior to the analysis or prediction (*subsample analysis*) [6]. Additionally we've chosen 1106 stocks that are or once existed as components of the Russell 1000 index (which tracks approximately the 1000 largest public companies in the US) during the chosen test time period between 1997 and 2018. Our selection includes companies that either went

bankrupt or dropped out of Russell 1000, significantly reducing survivorship bias in our training data. This test population is larger than a lot of previous studies of similar nature. For example Beyaz and co only chose 140 stocks from S&P500 when they attempted to forecast stock prices both six months and a year out based on fundamental analysis and technical analysis [7] and Bradbury used a sample of only 172 firms to research the relationships among voluntary semi-annual earnings disclosures, earnings volatility, unexpected earnings, and firm size [8].

Recognising the highly nonlinear nature of stock price movements, we've chosen a variety of supervised learning models in search of one or some which can best work through the high noises embedded in the price data. We've experimented with a deep neural network, a varieties of support vector machines with different kernels, and an Extreme Gradient Boosting (XGB) model. In our experiments with all these models, we divide the training data into in-sample and out-of-sample periods of varying lengths and use the in-sample data set to tune a model's hyperparameters. Our early experiments show that a traditional grid search way of finding optimal parameter set is inexhaustive and can be very slow. Instead we've chosen to use the highly adaptable Genetic Algorithm to tune our models [9]. Since the search range and granularity of each model's tunable hyperparameters examined by the Genetic Algorithm are often unknown beforehand and they directly determine the complexity of the resulting model, they must be chosen sensibly. Searching with a limited sets of parameter will result in a nonoptimal model which will not able to fit the essential structure of the training data set. To avoid that potential problem we have chosen to use a broad value range and a small granular step for each of the hyperparameters. We employ a 5-fold cross validation (CV) within each Genetic Algorithm iteration for estimating the optimal combination of each model's hyperparameters.

As inspired by Chung's work [10] who successfully evidenced that short term return predictability captures other factors, such as volatility, information asymmetry, investor sophistication, volume, size, and trading costs that affect arbitrage activities and the extent to which information is impounded in prices, we would like to explore the possibility of using post-earnings return predictability as a broader measure of market efficiency in the context of PEAD. Our machine learning based approach is in direct contrast to most earlier work in the literature as typified by [11] which sought to predevise different portfolios by different characteristics of the factors under analysis and tried to analyse and determine the relationship between respective portfolios' return and the corresponding economic factors that segregated the portfolios. Instead we choose an innovative model $XGBoost + GA$ to make sense of the comprehensive input features and produce such output predictability to arbitrage in the context of PEAD. This is our main contribution to the literature.

II. RELATED WORK

Since the discovery of Post Earnings Announcement Drift as a stock market anomaly by Ball and Brown [2] who documented the return predictability for up to two months after the annual earnings announcements, extensive research has been carried out in the literature though with varying results. For example Foster, Olsen and Shevlin [12] found systematic post-announcement drifts in security returns are only found for a subset of earnings expectations models when testing drifts in the [+1, +60] trading day period. In recent years the literature has become less limited to the specific study of PEAD and instead put more focus on the direct predictions of stock price movement using stocks' fundamental and/or technical information, again with varying rates of success. Malkiel studied the impact of price/earnings (P/E) ratios and dividend yields on stock prices using the Campbell-Shiller model. He conceded his work demonstrated that exploitable arbitrage didn't exist for investors to earn excess risk-adjusted returns and he could not find a market timing strategy capable of producing returns exceeding buying and hold a broad market index [13]. Olson and Mossman on the other hand not only showed that artificial neural network outperforms traditional regression based methods when forecasting 12-month returns by examining 61 financial ratios for 2352 Canadian stocks but more importantly shows that by using fundamental metrics sourced from earning reports they were able to achieve excessive risk-adjusted returns [14].

Other authors went beyond metrics from earnings reports and attempted stock forecast using both fundamental and technical analysis. Sheta, et al. explored the use of ANN, SVM and Multiple Linear Regression for prediction of S&P500 market index. They selected 27 technical indicators as well as macro economic indicators and reported that SVM contributed to better predictions than the other models tested [15]. Hafezi et all considered both fundamental and technical analyses in a novel model called Bat-neural Network Multi-agent System when forecasting stock returns. The resulted MAPE statistic showed that the new model performed better than typical Neural Network coupled with Genetic Algorithm [16]. Alternative data are becoming popular too. Solberg and Karlsen investigated the possibility to predict the direction of stock prices using scripts of earnings conference calls. By analysing 29330 different earnings call scripts between 2014 and 2017 using four different machine learning algorithms they managed to achieve a classification error rate of 43.8% using logistic regression and beat the SP500 benchmark using both logistic regression and gradient boosting. Their results showed that earnings calls contain predictive power for next day's stock price direction post earnings release [17].

When it comes to selecting machine learning models for event driven stock price forecast the literature has looked a lot at Support Vector Machines. Zhang constructed a novel ensemble method integrated with Ad-

aBoost algorithm, probabilistic Support Vector Machine and Genetic Algorithm and verified its performance over 20 shares from the SZSE and 16 stocks from NASDAQ. He showed the new ensemble method achieved preferable profit in simulation of stock investment [18]. Madge used daily closing price for 34 technology stocks on a SVM model with radial kernel to calculate price volatility and momentum for individual stocks and for the overall sector. The model attempts to predict whether a stock price sometime in the future will be higher or lower than it is on a given day. They found little predictive ability in the short-run but definite predictive ability in the long-run [19]. Tsai and Cheng focused on testing the impact of feature selection while using GA to optimize SVR which is the regression version of SVM. They formulated stock price prediction as a time series problem, used a variety of technical indicators and other time series data as model inputs and evaluated a number of methods including Kernel Ridge Regression and Multivariate Adaptive Regression Splines etc for feature selection. They were able to show some feature selection methods were better than others which in turn meant certain inputs were more impactful than others [20].

Researchers also studied how machine learning would directly benefit financial trading. Through a series of applications involving hundreds of predictors and stocks, Huck looked at how to implement some of the state-of-the-art machine learning techniques to manage a long-short portfolio. In that process he also explored a series of practical questions with regard to the predictor data and was able to show that the techniques he examined generated useful trading signals for portfolios with short holding periods [21]. Sant'Anna and Caldeira applied Lasso regression for index tracking and long-short investing strategies. They used stocks from three benchmarks, S&P100, Russell 1000 and the Ibovespa Index from Brazil from 2010 to 2017 to assess the quality of Lasso-based tracking portfolios. By using cointegration as a benchmark method to solve the same problems they showed that the Lasso regression based approach was able to form portfolios that produced similar returns compared to using cointegration but incurred significantly less transaction costs [22].

We have not found any creditable research on stock forecast using XGBoost and we are contributing to the literature for that.

III. MODEL FEATURES GENERATION

We have chosen in total 1106 Russell 1000 companies for analysis. The chosen time frame is between the first financial quarter of 1997 (1997 Q1) and the fourth financial quarter of 2018 (Q4 2018). While the model output is a n day Cumulative Abnormal Return of a stock, the input to our models consists of the following sets of unadjusted data which we've sourced from Bloomberg:

- Financial statements data
- Earnings Surprise data
- Momentum indicator data
- Short interest data

In total we've sourced 97901 quarterly financial statements from our chosen companies over the test time frame. The final population of valid data points used for training and testing whose input features include both financial statement metrics and other economic metrics stands close to 50,000, depending on the test cases. There are a number of reasons for the reduced population: (a) there are no Earnings data, Short interest data or other input feature data on Bloomberg for a good number of historical financial quarters within the test time frame; (b) we've discarded certain companies in certain historical quarters when the earnings reports suffered badly from missing data; (c) We've been very careful with whether an earnings report was released before market opened, after market closed or during trading hours as such a difference is significant as we'd need to alter the forecast starting point accordingly. Bloomberg is missing such information for some financial quarters in earlier years and we've discarded those quarters.

A. Financial Statements data

Table 1 shows 24 metrics from earnings reports have been chosen to create training data.

Cash	Operating Margin
Cash from Operating Activities	Price to Book Ratios
Cost of Revenue	Price to Cashflow Ratios
Current Ratio	Price to Sales Ratios
Dividend Payout Ratio	Quick Ratio
Dividend Yield	Return On Assets
Free Cash Flow	Return On Common Equity
Gross Profit	Revenue
Income from Continued Operations	Short Term Debt
Inventory Turnover	Total Asset
Net Debt to EBIT	Total Asset
Net Income	Total Debt to Total Assets
Operating Expenses	Total Debt to Total Equity
Operating Income	Total Inventory
	Total Liabilities

TABLE I. Earnings report metrics chosen as input features

Based on the reported value of these metrics we've engineered new features as quarterly change and yearly change of each of all the 29 report metrics.

B. Earnings Surprise data

Earnings Surprise represents how much a company's actual reported Earnings Per Share (EPS) is more (or

less) than the average of a selected group of stock analysts' estimates on that quarter's EPS. We are not calculating Earnings Surprise as a %change between the reported EPS and market estimated EPS because (a) %change is too volatile as a very small change when the actual EPS levels is close to zero will lead to a misleading large %change, and (b) we would like to avoid the change-of-signs problem when EPS turns from negative to positive or vice versa.

We've subsequently engineered the following three features related to Earnings Surprise:

- Current quarter's Earnings Surprise (reported EPS minus market estimated EPS);
- Difference between current quarter's Earnings Surprise and that of the previous quarter;
- Difference between current quarter's Earnings Surprise and the average Earnings surprise of the preceding three quarters;

C. Momentum Indicators

We've chosen the following technical/momentum indicator values calculated on the same day an individual company's quarterly earnings data was released:

- 9-day Relative Strength Index (RSI)
- 30-day Relative Strength Index
- 5-day Moving Average / 50-day Moving Average
- 5-day Moving Average /200-day Moving Average
- 50-day Moving Average / 200-day Moving Average

We believe all these indicators should in a way measure how a stock's recent short term movements compare to its historical movements further back in time. The inclusion of momentum indicators is to allow the prediction process of future stock movements to take into account a stock's recent movement trend as information leakage does happen prior to financial reportings. We've engineered the three ratios of short term moving averages to near or long term moving averages as proxies to the *goldencrosses* indicators.

D. Short Interest data

Short interest ratio is released for most companies twice a month and is calculated by dividing the number of shares short in a stock by the stock's average daily trading volume. The short interest ratio is a good gauge on how heavily shorted a stock may be versus its trading volume. The most recent short interest ratio for each company prior to its earnings release is sourced as an input feature to the model for that company.

IV. DATA PRE-PROCESSING

With totally 1106 companies involved over 21 years, there is a lot of data representing input features for each company at each quarter. In order for them to be understood by the models we put them into a matrix-like data structure $A \in M_{m\times n}(\mathbb{R})$ where each of the m rows represents a n dimensional training data point, indexed by the pairing of a company name and a historical quarter , and each column holds data of the same feature from all the data points.

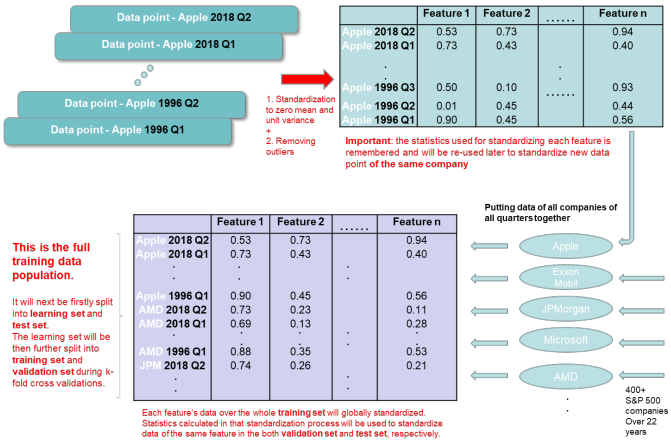

FIG. 1. Steps of Data Pre-processing

Before we put the data of all the companies and of all the quarters into a matrix, we pre-process each company's data to deal with outliers and to standardize data of every company. Firstly, we employ Winsorization [23] to reduce the number of outliers present in the input features. This is carried out on the feature data of each individual company. Secondly, we standardize a selective group of features of each company. Every company's standardized features will then be stacked back into a full training data set. The pre-processing process is illustrated in Figure 1.

V. MODELS AND METHODS

In order to forecast post earnings price drift we've chosen to experiment with a deep neural network, an Extreme Gradient Boosting mdoel, and support vector machines with different kernels.

A. Deep Neural Network

Deep learning has been a cornerstone in machine learning for the last few decades. Deep neutral networks have been used to achieve a lot of state-of-the-art results and are playing a big part either on their owns in many fields

such as image process or as part of other machine learning disciplines such as natural language process and reinforcement learning. In our experiments deep neural network whose final structure will be decided through the Genetic Algorithm optimization process, will serve as a benchmark model for SVM and XGBoost. Training neural network is a convex optimization problem with the loss function defined as:

$$L(\omega) \triangleq \sum_{i=1}^{M} L_i(\omega) \tag{1}$$

Where $L_i(\omega)$ is a loss function for data point $i \in \{1, 2, ..., M\}$ and ω are the model weights being optimized. A neural network is a nonlinear system due to the presence of an activation function on each neuron (except for the output neuron as it can be linear). An activation function $\sigma(x)$ can take a lot of forms such as sigmoid, tanh, ReLU, etc and it makes the output of a neuron look like $f(x) = \sigma(\omega^T X + b)$ with X being inputs to the neuron and b being the bias. To minimize the loss function, Stochastic Gradient Descend (SGD) and its varients (stochastic, batch and mini-batch SGD) are used to optimize these weights during the training process [24]. This is done in an iterative fashion and by using learning rate α and the Jacobian matrix of derivatives of the loss function with respect to all the model weights $\nabla L(\omega) = \left(\frac{\partial L}{\partial \omega_1}, \frac{\partial L}{\partial \omega_2}, ..., \frac{\partial}{\partial \omega_N}\right)$:

$$\omega_i := \omega_i - \alpha \nabla L(\omega) \tag{2}$$

B. Support Vector Machine

Support Vector Machine was first invented by Vladimir Vapnik and his colleagues in 1963 with its current standard form ϵ-SVM proposed by Cortes and Vapnik in 1995 [25]. Unlike regression based methods which aim at minimising the error function, SVM finds a hypothesis function $f(x)$ which represents a hyperplane in the input feature space whose prediction output $\widehat{y_i}$ deviates away from the actually observed value y_i by at most ϵ. Its linear form can be simply written as

$$f(x, \omega) = \sum x_j \omega_j + b \tag{3}$$

Seeking a flat hyperplane means minimizing ω and we can achieve this by finding the minimal norm value of ω , $\frac{1}{2} \parallel \omega \parallel^2$, effectively formulating it as a convex optimizing problem. Also to ensure optimization convergence one can use slack variables ξ_i, ξ_i^* [25] to introduce soft margin to the loss function which turns out in this form:

$$\text{Minimise } \frac{1}{2} \parallel \omega \parallel^2 + C \sum_{i}^{\ell} (\xi_i + \xi_i^*) \tag{4}$$

$$\text{Subject to } \begin{cases} \omega_i x_i + b - y_i \leq \epsilon + \xi_i \\ y_i - \omega_i x_i - b \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \tag{5}$$

Where C adjusts the balance between the flatness of the hypothesis f and how much deviation further from ϵ can be tolerated. We call this loss function the *primal* loss function. We can make the computation simpler by turning the primal loss function to its Lagrange *dual* formulation whose solution provides a lower bound to the solution of the primal problem and the dual form is expressed as:

$$\text{Maximise } \begin{cases} -\frac{1}{2} \sum_{i,j=1}^{\ell} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \langle x_i, x_j \rangle \\ -\epsilon \sum_{i=1}^{\ell} y_i (\alpha_i + \alpha_i^*) + \sum_{i=1}^{\ell} y_i (\alpha_i - \alpha_i^*) \\ \xi_i, \xi_i^* \geq 0 \end{cases} \tag{6}$$

$$\text{Subject to } \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C] \tag{7}$$

Here $\langle x_i, x_j \rangle$ is only for the linear form of SVM. This part of the loss function can be extended to using a kernel function $K(x_i, x_j) = \langle \psi(x_i), \psi(x_j) \rangle$ when a linear model can not adequately describe a regression problem where $\psi(x)$ is a transformation that maps x to a high-dimensional space. In our experiments we've tested the linear kernel as well as a sigmoid kernel, four polynomial kernels with degrees from 2 to 5 and a Radial Basis Function (RBF) kernel:

$$\begin{aligned} \text{Signmoid: } & K(x_i, x_j) = tanh(\alpha x_i^T x_j + c) \\ \text{Polynomial: } & K(x_i, x_j) = (x_i^T x_j + 1)^d \\ \text{RBF: } & K(x_i, x_j) = exp(-\frac{1}{2\sigma^2} \parallel x_i - x_j \parallel^2) \end{aligned}$$

C. Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is a scalable machine learning system for tree boosting invented by Tianqi Chen [26] which has gained much prominence in recent years. It distinguishes itself from other existing tree boosting methods [27] [28] by having cache-aware and sparsity-aware learnings. The former technology gives the system twice the speed against running a non-cache-aware but otherwise identical greedy tree splitting algorithm and the latter gives an amazing 50 times speed boosting against a naive implementation handling an Allstate-10k dataset [26]. More importantly XGBoost has achieved algorithmic optimizations by introducing regularized learning objective within a tree

structure which helps achieve smart tree splitting and branch pruning.

For a data set in matrix form $A \in M_{m\times n}(\mathbb{R})$ with m data points and n features, a tree ensemble model uses K base leaner functions to predict the output:

$$\widetilde{y_i} = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathbb{F} \tag{8}$$

Where $\mathbb{F}$ is the space of regression trees. Each hypothesis f_k corresponds to an independent tree structure q with leaf scores ω. XGBoost utilises regression trees each of which contains a score on each of its leaves. These scores help form the decision rules in the trees to classify each set of inputs into leaves and calculate the final predicted output by summing up the scores in the related leaves. Unlike other standard gradient boosting models such as AdaBoost and GBM which don't intrinsically perform regularization, XGBoost minimises a *regularized* loss function in order to learn the set of functions:

$$L(\phi) = \sum_i \ell(\widehat{y_i}, y_i) + \sum_k \Omega(f_k) \tag{9}$$

Here ℓ is a differentiable convex loss function for the model output and the regularization term is defined as (though not limited to) $\Omega(f) = \gamma T + \frac{1}{2}\lambda||\omega||^2$ which reduces the chance of overfitting. As in a typical gradient tree boosting model a new base learner regression tree f_i which most minimizes the loss function in equation 9 is greedily and iteratively added to the final loss function. Let $\widehat{y}_{i,t}$ be the model output of the i-th instance at the t-th iteration the loss function can be re-written as

$$L_t(\phi) = \sum_{i,k} \ell(y_i, \widehat{y}_{i,t-1} + f_t(x_i)) + \sum_k \Omega(f_{k,t}) \tag{10}$$

By taking the Taylor expansion on this loss function up to the second order and removing the constant terms as a result of the expansion the loss function can be simplied to:

$$L_t(\phi) = \sum_{j=1}^{T}[G_j\omega_j + \frac{1}{2}(H_j + \lambda)\omega_j^2] + \lambda T \tag{11}$$

Where

$$\begin{aligned} G_j &= \sum_{i\in I_j} g_i \\ H_j &= \sum_{i\in I_j} h_i \\ I_j &= \{i|q(x_i) = j\} \\ g_i &= \partial_{\widehat{y}_{i,t-1}}\ell(y_i, \widehat{y}_{i,t-1}) \\ h_i &= \partial^2_{\widehat{y}_{i,t-1}}\ell(y_i, \widehat{y}_{i,t-1}) \end{aligned}$$

Here T is the number of leaves in the tree. With ω_j being independent with respect to others, Tianqi [26] has proven that the best ω_j for a given tree structure $q(x)$ should be

$$\omega_j^* = -\frac{G_j}{H_j + \lambda} \tag{12}$$

which in turn makes the objective function come to its final form:

$$L_j^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T \tag{13}$$

Ideally the model would enumerate all possible tree structures with a quality score and pick the best one to be added iteratively. In reality this is intractable and optimization has to be done one tree level at a time. This is made available by the final form of the loss function as the model uses it as a scoring function to decide the optimal leaf splitting point. Assume that I_L and I_R are the instance sets of left and right nodes after the split. Letting $I = I_L \cup I_R$, the scoring function for leaf splitting is

$$L_{split} = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} + \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma \tag{14}$$

These scores are then used by a method called the *exact greedy algorithm* to enumerate all the possible splits for continuous features, allowing each level of a tree to be optimized and the overall loss function to be minimised in the process. When deployed on a distributed platform XGBoost employs approximate algorithms instead to alleviate the huge memory consumption demanded by the exact greedy algorithm although this is not needed in our experiments which run on a single machine.

D. Model Tuning

The whole training data population is split into training set and test set. We have devised a number of test cases. In each case we forecast post earning cumulative abnormal returns on all the stocks that released data either in the same financial quarter or on the same date. Consequently the test set includes those stocks from the test quarter or test date and the training set includes all the data points prior to the test quarter or test date. The training set is used to tune the models with the help of Genetic Algorithm (GA) and cross validation (CV). Model tuning is one of the two most important steps (the other being data cleansing) in ensuring the model output can meaningfully capture the underlying dynamics of the dependent variable. In search of optimal hyperparameter

Deep Neural Network	XGBoost	Support Vector Machine
Number of epochs	*Max depth*	*Kernel method*
Hidden layer neuron count	*Sub sample*	*Gamma*
Dropout rate	*Column sample by tree*	*C (model's penalty parameter)*
Regularization Lambda	*Gamma*	*Epsilon*
Learning rate	*Learning Rate*	
Hidden layer count	*Minimum child weight*	

TABLE II. Model hyperparameters optimized by GA + CV

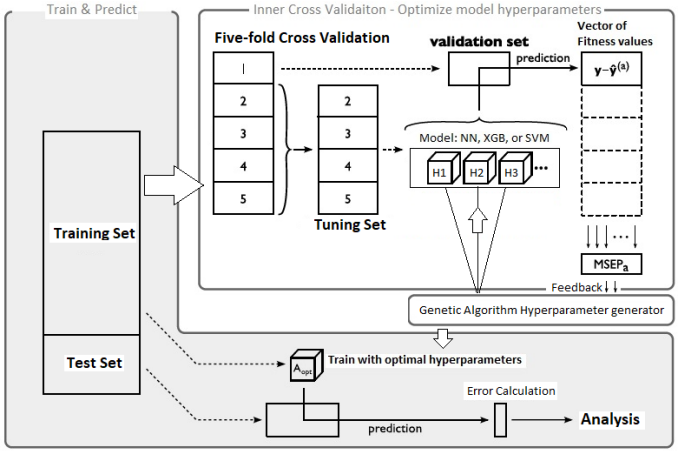

FIG. 2. Hyperparameter Tuning using GA + CV

sets we experimented a more straightforward approach of grid search but found it less effective in its performance and inexhuastive in the search results. Genetic Algorithm as an adaptable and easily extensible heuristic optimization method has been chosen to perform model tuning on all the selected models under experiment. Table II gives the list of hyperparameters of every model that we've put through GA for tuning:

We would like to note that researchers in the literature typically focus on one or two kernel methods to go with the Support Vector Machine models. For instance Tay and Gao chose Gaussian kernel with SVM to forecast financial time series [17] and Madge used Radial basis function (RBF) kernel in his attempt to forecast stock price movement [14]. Instead we've chosen 7 different kernels (including RBF, Sigmoid, Linear, and Polynomial of degrees 2 to 5) and use GA to optimize SVM's output accuracy out of all these kernels. This ensures we are not limited to a small number of common kernels like we've seen in the literature and instead we take full advantage of GA's optimization prowess to help us identify the best kernel and its accompanying model parameters for our SVM model. Similarly, when using multi-layer Neural Network researchers in the literature typically pre-fix the number of hidden layers or the number of neurons in each hidden layers for their models and only carry out model tuning on common hyperparameters such as dropout rate and learning rate. Again this practice can be subjected to sub-optimal model accuracy as the modeller has not included the model structure as part of the model optimization process and instead only focus on the hyper parameters of a predefined structure. Recognising the deficiency of this model calibration process we are including the number of hidden layers and the number of neurons in each hidden layer as our tuning targets effectively tuning both the Neural Network model structure as well as model hyperparameters simultaneously. We have given both the hidden layer count and neuron count in each layer a large enough range so that a NN can go deeper if the GA optimization finds it necessary.

When tuning a model each of the tuning targets is randomly initialised according to its own valid range of values. This initialization is repeated 40 times so that we have 40 sets of randomly initialised hyperparameters to start the GA process with. Each set is called a *population* and each hyperparameter within a set is called a *chromosome*. All of the 40 populations are considered to be part of the current *generation*. The GA process carries out a 5-fold cross-validation on a model using each of the 40 populations and when finished, keep the 20 populations that have produced the smallest fitness values in the cross validation step. These 20 sets or *populations* of hyperparameters are considered to have performed better in forecasting post-announcement drifts with the current model than the 20 discarded ones. These 20 better populations are then used to *cross-breed* into 20 new populations and in this process *mutation* is allowed to happen to the cross-bred populations, i.e. chromosomes in the 20 newly created populations are allowed to randomly change value following a predefined level of probability. At the end of this process we have produced a new and potentially better set of 40 populations of hyperparameters and we call them the new generation. The new generation are then fed through a second iteration of the GA process until eventually the minimum fitness value produced by the cross-validation step no longer changes its value within tolerance and at this point we've arrived at the optimal set of tuning targets which produces the smallest fitness value when being used in the current model. Figure 2 shows how Genetic Algorithm and Cross Validation work together to produce the set of hyperparameters of each model which result in the highest prediction accuracy (smallest fitness value) on the validation set.

VI. RESULTS

Having prepared and engineered a wide range of feature data, we carefully tune a deep Neural Network, an Extreme Gradient Boosting model, as well as Support Vector Machine (SVM) models of different kernels. Our first experiment is to forecast 30 day post-earnings Cu-

Model	RMSE	Classification Success Rate
XGB	1.03%	58%
DNN	1.23%	53%
SVM	1.74%	52%

TABLE III. Prediction result metrics by models

mulative Abnormal Return (CAR) as a measure of risk adjusted stock price return. The purpose of this experiment is to measure which model may perform better than other ones and we will then choose this same model for other experiments. An abnormal return is between the actual return of a security and its expected rate of return.

$$AR_{it} = r_{it} - E(r_{it}) \tag{15}$$

Where AR_{it} is the one-day abnormal return for company i on day t, r_{it} is the actual one-day stock return and $E(r_{it})$ is the expected return of stock i. As explored by Kim [11] there are a variety of ways of evaluating the expected return including using quantitative models such as the one-factor CAPM model and the Fama French three-factor model [3]. In our experiments we choose to use the S&P500 index return to represent the broader market's return and use that to proxy a stock's expected return. Consequently our model output for stock i is simply

$$AR_i = \sum_{t=1}^{n} (r_{it} - E(r_{it})) \tag{16}$$

We have chosen to perform the test on all the stocks that filed for Q4 2018 quarterly earnings with U.S. Securities and Exchange Commission (SEC). That means all the companies in the 83 quarters from Q1 1997 to Q3 2018, totalling 47367 data points, are used as the training data points, whereas the 924 data points from Q4 2018 are used for out-of-sample testing. Although this is a regression problem because we forecast the CAR directly, we also examine the classification success rate by checking if both a predicted return and an actual return are of the same sign. As demonstrated by the Root Mean Square Error (RMSE) error metric and the classification success rate in table III, XGBoost has significantly outperformed both the deep Neural Networks and SVM in forecasting 30 day Cumulative Abnormal Return immediately after each company released its financial statements. As a result XGBoost has been chosen to conduct a series of subsequent experiments. Table IV lists all the model hyperparameters produced by the Genetic Algorithm routines which have produced the best results in respective models.

Our test population possesses over 1000 stocks include large cap, mid cap and small cap stocks sourced from 10 distinct industry sectors. Characteristics of any stock can fundamentally change over time as a company can develop from a relatively small cap growth stock into a large cap blue chip or decline from a market's darling to a penny stock. The underlying distribution of most companies' quarterly readings and the correlation between the readings and subsequent PEAD are constantly changing. All of these challenging factors have made it more meaningful to evaluate the dynamics of post-earnings drifts in the context of portfolios and we hope to capture an unseen *collective* trend of movement by certain stocks as triggered by their earnings release and other relevant economic factors.

XGB params

Learning Rate	0.14
Max Depth	9
Column Sample By Tree	0.84
Sub sample	0.76
Gamma	0.3
Min Child Weight	0

DNN params

Epoch	5
Hidden Neuron	47
Dropout Rate	0.1
Learning Rate	0.02
Hidden Layer	2

SVM params

Kernel	RBF
Gamma	0.02
C	0.02
Epsilon	0.15

TABLE IV. Model parameters used in test on 30 day PEAD forecasting accuracy

Using XGBoost + GA, We've tested sets of stocks that released their earnings during the entire Q3 2018 earnings release season, the Q4 2018 season as well as on a couple of specific dates throughout 2018. We've also looked at stocks from the Financial sector. All the data points (a certain company's quarterly earnings and its price performance after the report) in our training and test sets are independent in that each data point's inputs and output are independent of other data points. This in theory makes it possible to perform forecasting tests on random data points. However practically speaking a portfolio building system would not go about finding past stocks to include in any portfolio or trading strategy. We therefore have chosen data points from the two most recent quarters Q3 2018 and Q4 2018 as test candidates. A more important reason of choosing these two quarters is that the US stock market went through two polar opposite phases of development in these two quarters with the S&P500 shedding 20% in the last quarter of 2018 (around the time most Q3 2018 earnings were reported) for fear of Fed rate rises and trade war escalation among other things but gaining a major rebound in the first quarter of 2019 (when most US companies reported Q4 2018 earnings). Our intention is to evaluate if our model can successfully capture those very different

PEAD dynamics given very different macro conditions and different company specific accounts.

A. All stocks from the same quarter

We've first discovered that by working with our carefully selected and engineered input features the XGBoost+GA model produces prediction results which have successfully captured certain dynamics of PEAD in portfolios of stocks. By firstly ranking the out-of-sample stocks according to their predicted n day post earning cumulative abnormal returns, we are observing that portfolios which are made up of stocks from top quantiles of the ranked stock list are consistently producing top positive actual returns whereas portfolios which are made up of stocks from bottom quantiles are consistently producing low negative returns. A long-short market neutral strategy [29] could be formed through longing the top quantile portfolio and shorting the bottom one. Tables VI and VII provide the stats with regard to the forecasts carried out on Q3 2018 and Q4 2018 earnings season. Despite under drastically different market conditions, tests on stocks from these two quarters consistently show similar classification success rate of 58%-59% in a 30-day forecast tests after many of the same tests have been run. Each point on figures 3 and 4 is the actual return of a moving portfolio consisting of 100 stocks when moving along the full list of out-of-sample stocks that have been sorted by their predicted 30-day risk-adjusted returns. Of all of the same tests we've run, the points at the front and the back of the graphs are consistently showing downward pattern of actual portfolio returns. Although the middle part of the figures is not showing pronounced pattern, in latter experiments when less number of stocks are involved and the stocks chosen share more commonalities, we'll see that in fact the entire graph is showing a trend of downward returns, and that is clearly driven by ranking the stocks by their predicted returns from high to low.

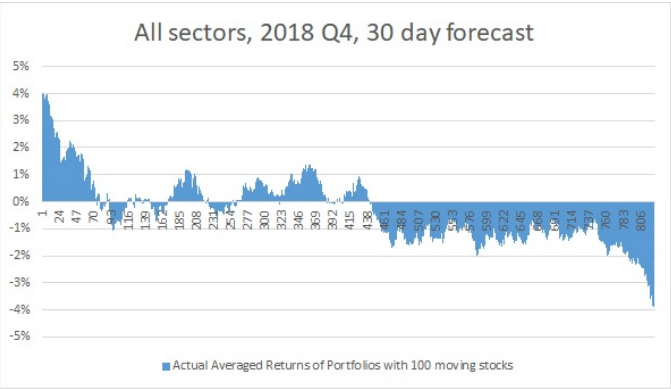

FIG. 3. 2018 Q4 test result. Actual average returns of portfolios with 100 moving stocks which have been ranked by their predicted returns

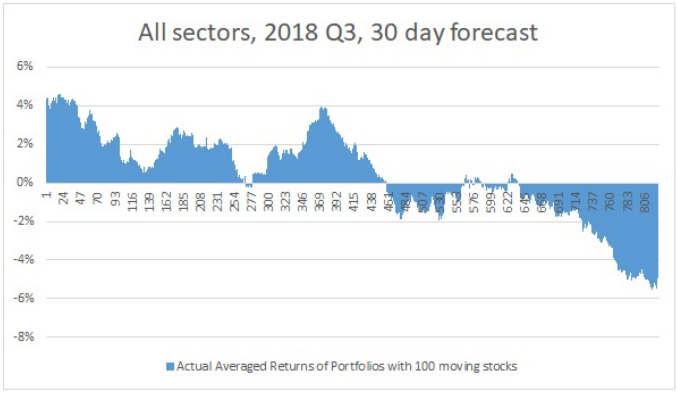

FIG. 4. 2018 Q3 test result. Actual average returns of portfolios with 100 moving stocks which have been ranked by their predicted returns

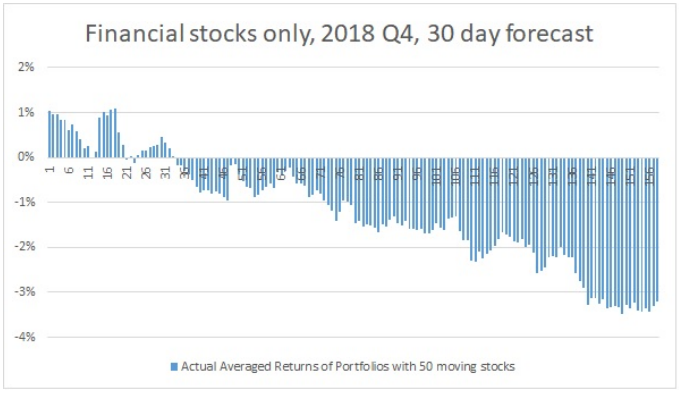

FIG. 5. 2018 Q4 test result. Actual average returns of portfolios with 50 moving Financial stocks which have been ranked by their predicted returns

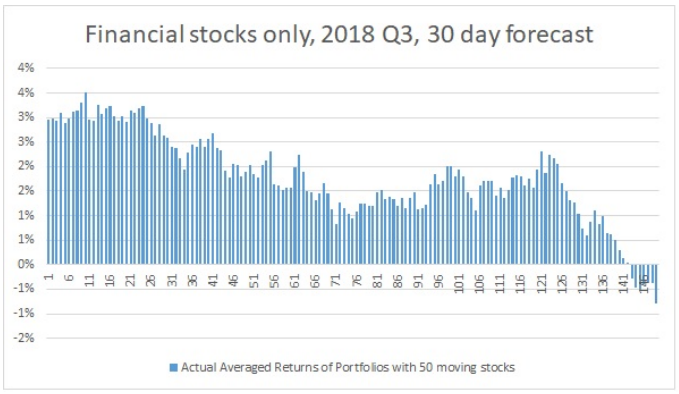

FIG. 6. Actual average returns of portfolios with 50 moving stocks which have been ranked by their predicted returns

B. Financial stocks from the same quarter

This observation is also seen when we examine stocks from only one industrial sector. With less number of out-of-sample stocks involved, we are constructing mov-

Out-of-sample Time Frame	Industries	Forecast Holding Period	Top Quantile Portfolio Return	Average Actual Return	Bottom Quantile Portfolio Return
Q4 2018	All	30 days	3.90%	-0.29%	-3.76%
Q4 2018	Financials	30 days	0.92%	-1.13%	-3.35%
Q3 2018	All	30 days	4.09%	0.36%	-4.78%
Q3 2018	Financials	30 days	2.97%	1.46%	-0.56%
25-Oct-18	All	1 day	0.78%	-1.05%	-2.15%
26-Apr-18	All	1 day	0.49%	-0.51%	-1.60%
02-Aug-18	All	30 days	2.75%	1.16%	0.54%
25-Oct-18	All	30 days	3.83%	1.06%	-1.64%

TABLE V. Actual returns of portfolios consisting of top and bottom quantile stocks

Out-of-sample Time Frame	Industries	Forecast Holding Period	In-sample Data Points	Out-of-sample Data Points	Classification Success Rate
Q4 2018	All	30 days	47367	924	58%
Q4 2018	Financials	30 days	7923	207	63%

TABLE VI. Stats on forecasting stock 30 day PEAD in Q4 2018

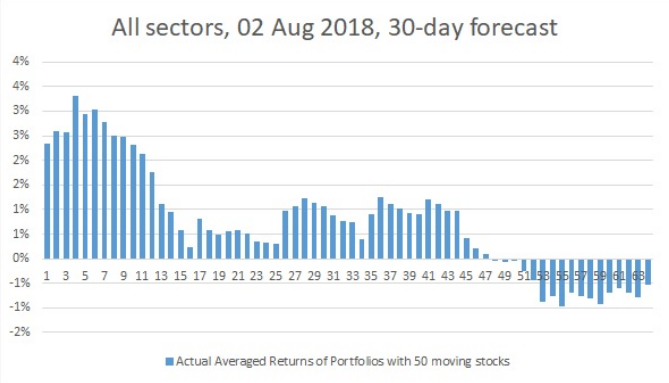

FIG. 7. Actual average returns of portfolios with 50 moving stocks which have been ranked by their predicted returns

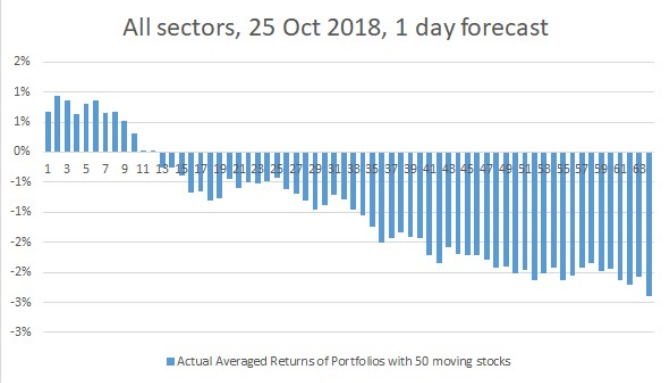

FIG. 9. Actual average returns of portfolios with 50 moving stocks which have been ranked by their predicted returns

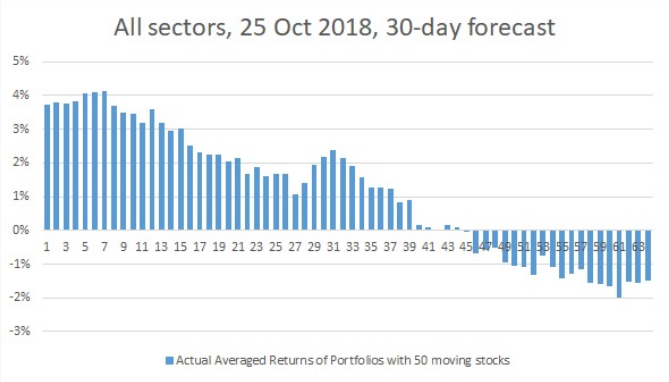

FIG. 8. Actual average returns of portfolios with 50 moving stocks which have been ranked by their predicted returns

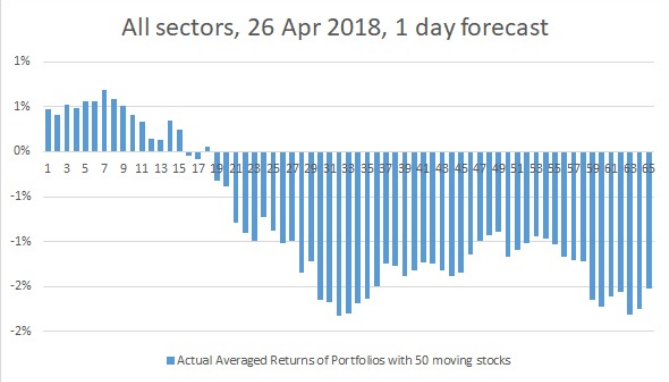

FIG. 10. Actual average returns of portfolios with 50 moving stocks which have been ranked by their predicted returns

ing portfolios of 50 stocks. Figures 5 and 6 are showing a more pronounced downward trend of actual portfolio returns in the Financial sector. Additionally a more interesting dynamics has been correctly captured by the

Out-of-sample Time Frame	Industries	Forecast Holding Period	In-sample Data Points	Out-of-sample Data Points	Classification Success Rate
Q3 2018	All	30 days	46468	899	59%
Q3 2018	Financials	30 days	7716	207	62%

TABLE VII. Stats on forecasting stock 30 day PEAD in Q3 2018

Out-of-sample Time Frame	Industries	Forecast Holding Period	In-sample Data Points	Out-of-sample Data Points	Classification Success Rate
25-Oct-18	All	1 day	48278	113	61%
26-Apr-18	All	1 day	48278	116	58%
02-Aug-18	All	30 days	48195	96	65%
25-Oct-18	All	30 days	48178	113	63%

TABLE VIII. Stats on forecasting stock PEAD on specific dates

results: With the same downward trend of actual portfolio returns captured by the model results, the 2018 Q4 result graph of Financial stocks has produced more portfolios with negative actual returns whereas the 2018 Q3 result has done the opposite. This phenomena is in fact in line with what happened in the real markets. The 207 Financial stocks chosen for the 2018 Q4 test produced an average of -1.13% real life 30-day risk-adjusted return against the S&P500 index (they were lagging behind the rallying market) whereas the same group of stocks outperformed S&P500 by 1.46% during the 2018 Q3 earnings release season during which period S&P500 lost 20% of value. Our result has not only correctly distinguished stocks that could form top performing portfolios on both sides of the zero to long and to short but has done so in totally contrasting market conditions.

C. All stocks from the same date

To demonstrate the applicability of using prediction results to help form tradable portfolios, we also show that we can achieve the same kind of results when we predict post-earnings Cumulative Abnormal Returns on stocks which report their earnings on the same day. In order to form tradable portfolios with a fixed holding period whose returns can be viewed from high to low, practically speaking it only makes sense if we are able to construct market neutral portfolios and execute the buying and short-selling of model-chosen stocks within a short time frame, such as within a day or ideally less. We've chosen three dates in 2018 when more stocks filed for earnings with SEC than on other dates. We don't discriminate industrial sectors on these dates. Table VIII shows even better classification success rates than earlier tests. Figures 7, 8, 9, 10 are portfolio results created based on predicted 30-day or 1-day CAR forecast and all show the model's ability to rank stocks by their predicted risk-adjusted returns in a way portfolios can be constructed which would produce top positive returns or low negative returns.

Table V gives the stats on how the top quantile portfolios and bottom quantile portfolios are performing relative to each other in all of the eight tests discussed in the *results* section. The Top Quantile Protfolio Return column is the average of the top five portfolios' actual risk-adjusted returns following earnings release and the Bottom Quantile Portfolio Return is the average of the bottom five portfolios' actual risk-adjusted returns. They are all on either side of the average actual return of all the stocks in the test population and in some cases significantly higher or lower than the test population's average. Such patterns of portfolio returns make them good candidates for a long-short strategy capitalising on the events of earnings release.

It's also worth mentioning that most of these tests have been run with deep neural nets and SVM but their results have been more inferior.

VII. CONCLUSION

Post-earnings announcement drift is a well known and well studied stock market anomaly when a stock's risk adjusted price can continue in the direction of an earnings surprise in the near to mid term following an earnings release. Pass research was however often limited in using simpler regression based methods to explain this phenomenon and was often confined to using a limited set of explaining factors. Even fewer research was carried out on how to potentially take advantage of this known anomaly and conduct forecast on stock price movements following such a significant economic event to companies. Attempting to plug this gap in the literature, our experiment is including a much bigger set of carefully selected input factors of various types with some being specifically engineered, sourcing the data over a longer historical time frame and attempting to forecast Cumulative Abnormal Returns (CAR) having *learnt* the internal links. We've adopted some state-of-the-art models and put them through rigorous tuning process. We not only look at specific forecast success rates but also examine if there is a *collective* trend of movement enjoyed by a group of stocks following their individual earnings

release. Our results first show that when properly configured using Generic Algorithm, XGBoost performs best compared to deep neural network and SVM in forecasting post-earnings CAR. We show that our selected input features are genuinely driving the direction of PEAD with a classification success rate ranging between 58% and 65% depending on the test scenario. Guided by the model's prediction outputs we are successful in building portfolios which consistently offer high positive returns and low negative returns and can potentially be used to form market neutral long-short trading strategy.

BIBLIOGRAPHY

[1] Gocken et al. "Integrating metaheuristics and artificial neural networks for improved stock rice prediction". In: *Expert Systems With Applications* 44 (2016).

[2] R. Ball and P Brown. "An Empirical Evaluation of Accounting Income Numbers". In: *Journal of Accounting Research* (1968), pp. 159–78.

[3] E. Fama and K. French. "Common Risk Factors in the Returns on Stocks and Bonds". In: *Journal of Financial Economics* 22 (1993).

[4] R. Bhushan. "An informational efficiency perspective on the post-eanings announcement drift". In: *Journal o f Accounting and Economics* 18 (1994), pp. 45–65.

[5] Luke Qiu. *Earnings Announcement and Abnormal Return of S&P 500 Companies*. 2014. DOI: https://openscholarship.wustl.edu/wushta_spr2014/74.

[6] H. Kent Baker and Yang Ni. "Competitive earnings news and post-earnings announcement drift". In: *International Review of Financial Analysis* (2016).

[7] Erhan Beyaz et al. "Comparing Technical and Fundamental indicators in stock price forecasting". In: *IEEE 4th International Conference on Data Science and Systems* (June 2018), pp. 1607–1613.

[8] Michael E. Bradbury. "Voluntary Semiannual Earnings Disclosures, Earnings Volatility, Unexpected Earnings, and Firm Size". In: *Journal o f Accounting Research* 30 (Spring 1992).

[9] Shangkun Deng and Kazuki Yoshiyama. "Hybrid Method of Multiple Kernel Learning and Genetic Algorithm for Forecasting Short-Term Foreign Exchange Rates". In: *Computational Economics* 45 (2013).

[10] Dennis Y. Chung and Karel Hrazdil. "Market Efficiency and the Post-Earnings Announcement Drift". In: *Contemporary Accounting Research* 28 (2011), pp. 926–956.

[11] Dongcheol Kim and Myungsun Kim. "A Multifactor Explanation of Post-Earnings Announcement Drift". In: *The Journal of Financial and Quantitative Analysis* 38 (2003), pp. 383–398.

[12] G. Foster, C. Olsen, and T. Shevlin. "Earnings releases, anomalies, and the behavior of security returns". In: *The Accounting Review* 59 (1984).

[13] Burton G. Malkiel. "Models of stock market predictability". In: *Journal of Financial Research* 27.4 (2004), pp. 449–459.

[14] Dennis Olson and Charles Mossman. "Neural network forecasts of Canadian stock returns using accounting ratios". In: *International Journal of Forecasting* 19(3) (2003), pp. 453–465.

[15] Alaa F. Sheta, Sara Elsir M. Ahmed, and Hossam Faris. "A comparison between regression, artificial neural networks and support vector machines for predicting stock market index". In: *Soft Computing* 8 (2015).

[16] Reza Hafezi, Jamal Shahrabi, and Esmaeil Hadavandi. "A batneural network multi-agent system (BNNMAS) for stock price prediction: Case study of DAX stock price". In: *Applied Soft Computing* 29 (2015), pp. 196–210.

[17] Lars Erik Solberg and Jørgen Karlsen. "The predictive power of earnings conference calls : predicting stock price movement with earnings call transcripts". MA thesis. Norwegian School of Economics, 2018.

[18] Zhang X., Li A., and Pan R. "Stock trend prediction based on a new status box method and AdaBoost probabilistic support vector machine". In: *Applied Soft Computing* 49 (2016), pp. 385–398.

[19] S. Madge. *Predicting Stock Price Direction using Support Vector Machines*. Independent Work Report. 2015.

[20] Ming-Chi Tsai et al. *Forecasting leading industry stock prices based on a hybrid time-series forecast model*. PLoS ONE 13(12). 2018. DOI: https://doi.org/10.1371/journal.pone.0209922.

[21] Nicolas Huck. "Large data sets and machine learning: Applications to statistical arbitrage". In: *European Journal of Operational Research* 278 (2019), pp. 330–342.

[22] Leonardo Sant'Anna and Jofffdfffdo Frois Caldeira. "Lasso-based index tracking and statistical arbitrage long-short strategies". In: *The North American Journal of Economics and Finance* (2019).

[23] Bin Duan and William P. Dunlap. *The Robustness of Trimming and Winsorization When the Population Distribution Is Skewed*. ProQuest Dissertations and Theses. 1998.

[24] Nitish Keskar and Dheevatsa Mudigere. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima". In: *5th International Conference on Learning Representations* (2017).

[25] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". In: *Machine Learning* 20 (Sept. 1995), pp. 273–297.

[26] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), pp. 785–794.

[27] Stephen Tyree et al. "Parallel boosted regression trees for web search ranking". In: *Proceedings of the 20th international conference on World wide web* (2011), pp. 387–396.

[28] Jerry Ye et al. "Stochastic gradient boosted distributed decision trees". In: *Proceedings of the 18th ACM conference on Information and knowledge management* (2009), pp. 2061–2064.

[29] Lars Erik Solberg and Jørgen Karlsen. "Pairs Trading Using Machine Learning: An Empirical Study". MA thesis. Erasmus University Rotterdam, 2017.