An Adaptive Master-Slave Regularized Model for Unexpected Revenue Prediction Enhanced with Alternative Data

Jin Xu[‡ †] Jingbo Zhou[*†] Yongpo Jia[§] Jian Li[‡] Xiong Hui[*† ¶]
[†]*Business Intelligence Lab, Baidu Research, China*
[‡]*Institute for Interdisciplinary Information Sciences, Tsinghua University, China*,
[†]*National Engineering Laboratory of Deep Learning Technology and Application, China*,
[§]*Department of Earth System Science, Tsinghua University, China*, [¶]*Rutgers University*,
Email: [†]zhoujingbo@baidu.com, {[‡]jxu3425,[‡]lapordge,[†]xionghui}@gmail.com,[§]jiayongpo@tsinghua.edu.cn

***Abstract*—Revenue prediction is an essential component in security analysis since the revenue of a company has a great impact on the performance of its stock. For investment, one of the most valuable pieces of information is the company's unexpected revenue, which is the difference between the officially reported revenue and the consensus estimate for revenue predicted by analysts. Since it is the unexpected revenue that indicates something exceeding or under analysts' expectation, it is an indispensable factor that influences the performance of a stock. Besides conventional trading data from stock market and companies' financial reports, recent years have witnessed an extensive application of alternative data for gaining an information edge in stock investment.**

In this paper, we study the challenging problem of better predicting unexpected revenue of a company via machine learning with alternative data. To the best of our knowledge, this is the first work studying this problem in literature. However, it is nontrivial to quantitatively model the relations between the unexpected revenue and the information provided by alternative data with a machine learning approach. Thus we proposed an adaptive master-slave regularized model, called AMS for short, to effectively leverage alternative data for unexpected revenue prediction. AMS first trains a master model upon a company graph, which captures the relations among companies, using a graph neural network (GNN). Then for a target company, the master model generates an adaptive slave-model, which is specially optimized for this target company. Finally, we use this slave-model to predict the unexpected revenue of the target company. Besides its excellent prediction performance, another critical advantage of our AMS model lies in its superior interpretability, which is crucial for portfolio managers to understand the predicted results. With extensive experiments using two real-world alternative datasets, we have demonstrated the effectiveness of our model against a set of competitors.

I. Introduction

Since the revenue of a company is closely related to its stock's future worth, revenue prediction has been widely considered as an essential but difficult component of security analysis [1]. The revenue is a company's total income from its normal business activities including sales of goods and services online and offline. In order to predict a company's revenue as accurate as possible, financial analysts usually research thoroughly on a significant amount of data, including the company's business models, financial reports, industrial case studies, and other relevant data for a compilation. Analysts also take different factors into consideration, such as competition from similar business, seasonal and periodic trends. Since different analysts often have different analysis results of the revenue of a company, we name the average of the analysts' estimation of a company's revenue as *consensus*.

Unexpected revenue refers to the amount of the officially reported revenue of a company exceeding or under its consensus. Since the unexpected revenue of a company often indicates something beyond analysts' expectation, it is one of the most valuable revenue information for investment. The relation between unexpected revenue and stock return has been extensively investigated in the literature [2]–[6]. Existing studies show that revenue surprises caused by unexpected revenue can bring significant abnormal returns in stock market [3], [5]. According to a recent study using alternative data by J.P. Morgan [7], the unexpected revenue has been proven to be an effective signal for trading.

Besides conventional data from stock market and financial reports, alternative data, such as connections between companies and transaction records, has attracted great attention for financial investment [7]. Alternative data contains information beyond traditional financial and economic sources, and such data is considered as the least utilized alpha source today [8]. Generally speaking, the alternative data used for investment can be classified into three categories: data produced by individuals (such as social media posts), data generated through business processes (such as credit card or online payment App transaction data), and data generated by sensors (such as GPS) [7]. Take the online transaction data as an example of alternative data. The sum of online transaction amount usually has a positive relationship with a company's revenue. In addition, location data belongs to the third category of alternative data generated by mobile application. It records the number of people appearing around an offline store over a period of time by GPS devices. Generally, more people appear around a company's offline stores, more revenue the company

[*]Jingbo Zhou and Hui Xiong are corresponding authors.

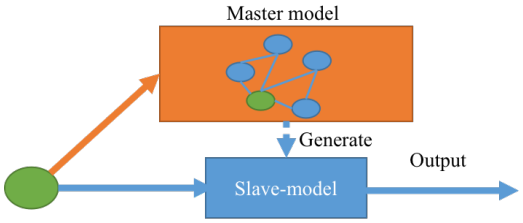


Fig. 1. Model overview. The green node is the target company and the blue nodes are companies related to the target company (explained in Section III-C). The master model generates the parameters of the slave-model. Then the slave-model receives the target company and predicts the unexpected revenue of the target company.

can get. As we can see from above discussion, alternative data can provide additional insights of a company's revenue. More importantly, harnessing such data can help investors gain a significant information edge over others.

However, it is nontrivial to use machine learning techniques to analyze alternative data and it has attracted great attention recently [9]–[14]. Existing studies focus on using alternative data to predict the market trends [9]–[12] or the individual stock trends [13], [14], mainly based on social media or news.

Arguably, predicting the unexpected revenue is one of the most important problems in predicting the stocks' future return. As alternative data provides different perspectives and insights for investment, it is desirable to use alternative data to enhance the performance of unexpected revenue prediction. As far as we know, there are few existing works to study on predicting the unexpected revenue with alternative data. One exception is the case introduced by Kolanovic et al. [7] who simply aggregated alternative data (like the foot traffic of a company) as a reference to help portfolio manager (PM) make investment decisions. Therefore, an advanced model which can quantitatively measure the relationship between the revenue and alternative data is strongly desired. To the best of our knowledge, our work is the first study on building a machine learning model to quantitatively forecast the unexpected revenue of a company with alternative data.

There are several challenges in predicting unexpected revenue using a machine learning model with alternative data. First, alternative data can be noisy due to the uncertainties in data collection and processing, which may result in a model that overfits the noise but does not generalize well. Second, the revenue data of a company is usually sparse because of its limited accessibility. Each company has at most four reported revenue every year, and the reported revenue over the past several years may become invalid since the business of the company may have been changed. Thus it is non-trivial to train a complex machine learning model on the alternative data with the limited labelled financial data. Third, it is also challenging to devise a model with a high interpretability. Since existing deep learning based models are a kind of black-box, their results are hard for PMs to understand. The interpretability of the model's forecast result is important for PMs as it can provide sources of uncertainties and hence help PMs control the risk better [15], [16].

To tackle the above challenges, in this paper, we first propose a novel model, called Adaptive Master-Slave regularized model (called AMS for short) for unexpected revenue prediction with alternative data. To deal with the challenges of data noise and data sparsity, the AMS model makes a trade off between the model's complexity of utilizing alternative data and its simplicity for unexpected revenue prediction. An overview of our AMS model is illustrated in Figure 1. The key idea of AMS is to train a master model first, then the master model generates an adaptive slave-model for each target company on the fly. At the same time, the master model also regularizes the parameters of the generated slave-model. Note that different slave-models are generated and adaptively optimized for each company.

In this paper, we first construct a graph of company correlations, where each node corresponds to a company, and each edge indicates the similar trend of historical revenue between two companies. The details of constructing such a company correlation graph is discussed in Section III-C. Then we use a Graph Neural Network (GNN) on the company correlation graph to construct the master model of AMS. The GNN based master model can not only can gain more valuable insights hidden in the company correlations, but also offset the data noise (of alternative data) and data sparsity (of revenue data) problems, and finally improve the capability of AMS. To avoid overfitting of training the slave-model of AMS, we also propose two novel regularization techniques, called supervised linear regression (LR) generation and model assembly, to regulate the flexibility of slave-models, which improves the performance of AMS in practice.

For the last challenge, namely the model's interpretability, the result of AMS is more interpretable compared with other black-box deep learning models. In AMS, we use linear regression (LR) as the slave-model. Since the final result of the unexpected revenue is predicted by an LR model, it is easy for PMs to understand the result and estimate the sensitivity of each feature (by observing the changes of the outcome by increasing a small delta unit of the feature value). Our AMS model can improve the capacity of the conventional linear regression model while keeping its interpretability.

We summarize our contributions as follows:

- We are the first to study the unexpected revenue prediction problem with machine learning approach using alternative data.
- We first propose a novel AMS model for unexpected revenue prediction. Specifically, we use a GNN on company correlation graph as a key component to construct the master model. Meanwhile, we also propose two novel regularization techniques, supervised LR generation and model assembly, to optimize the generated slave-models.
- Compared with other black-box deep learning models, the output of our AMS model is more interpretable.
- We conducted extensive experiments on two alternative datasets – online transaction amount data and map query data – to demonstrate the effectiveness of our model.

The rest of the paper is organized as follows. Next, we discuss preliminaries in Section II, followed by the detail of AMS in Section III. Then we evaluate our model in Section IV. Finally we discuss related work in Section V and conclude the paper in Section VI.

II. Preliminaries

In this section, we introduce preliminaries about this paper. First, we provide an introduction to the unexpected revenue prediction problem. Then we discuss metrics used to evaluate our model. Finally we present a formal problem definition.

A. Unexpected revenue prediction

Unexpected revenue is defined as the amount of the reported revenue of a company minus analysts' consensus of its revenue, which is the expectation of the analysts' estimated revenue whose formal definition is $UR = R - E(R)$ where UR is the unexpected revenue, R is the actual revenue, and $E(R)$ is the expected revenue of the company. Hereafter, $E(R)$ is also called *consensus*, which refers to the expectation of the analysts' estimation of the revenue of a company.

The unexpected revenue of a company usually leads to abnormal stock returns. For example, if a company obtains a higher revenue than expected, it will achieve a lower price to earnings (PE) than the one on the current market. When the information of its increased revenue is released, its stock price may consequently increase, resulting in a higher return.

B. Metrics

Here we introduce two metrics to measure the quality of the predicted unexpected revenue. Hereafter, let us denote the predicted revenue as $\hat{R}$ and the actual revenue as R, the predicted unexpected revenue as $\hat{UR}$ and the actual unexpected revenue as UR. We can observe that only measuring the difference between the $\hat{UR}$ and UR is not enough since whether $\hat{UR}$ and UR are in the same direction is also very important to tell whether the company meets, exceeds or misses the market's expectation. Therefore, here we define two metrics, Bounded Correction (BC) and Surprise Ratio (SR), to measure the quality of the prediction result.

Definition II.1. *BC (Bounded Correction) BC is defined as* $BC = I(|\hat{UR} - UR| < |UR|)$ *where* $I(\cdot)$ *is an indicator function whose value is 1 when condition is true, and 0 otherwise.*

Lemma II.1. *If* $BC = 1$*,* $\hat{UR}$ *and* UR *are in the same direction and* $|\hat{R} - R| < |E(R) - R|$.

Proof. When $BC = 1$, there are four cases as follows: (C1) if $UR > 0$ and $\hat{UR} - UR > 0$, we can get $\hat{UR} > 0$ directly. (C2) if $UR > 0$ and $\hat{UR} - UR < 0$, we can get $UR - \hat{UR} < UR$ because $BC = 1$. Then we can get $\hat{UR} > 0$. (C3) if $UR < 0$ and $\hat{UR} - UR > 0$, we can get $\hat{UR} - UR < -UR$ because $BC = 1$. Then we can get $\hat{UR} < 0$. (C4) if $UR < 0$ and $\hat{UR} - UR < 0$, we can get $\hat{UR} < 0$ directly. In these cases, we can always get that $sign(\hat{UR}) = sign(UR)$ which means $\hat{UR}$ and UR are in the same direction.

We have two observations: (O1) $|UR| = |R - E(R)|$ is just the absolute error between analysts' consensus and actual revenue. (O2) $|\hat{UR} - UR| = |(\hat{R} - E(R)) - (R - E(R))| = |\hat{R} - R|$ is just the absolute error between predicted revenue and actual revenue. Then, based on (O1) and (O2), we can conclude that when BC is true, $|\hat{R} - R| < |E(R) - R|$. □

It provides a new perspective to understand BC that the predicted revenue is closer to actual revenue than analysts' consensus when $BC = 1$.

Given a set of companies, we can further compute **BA (Bounded Accuracy)**:

$$BA = \frac{1}{N}\sum_{i}^{N} BC_i$$

where N is the number of companies. A high BA value means that the prediction model can predict the correct direction of the unexpected revenue with high possibility. Note that the BA of a bad prediction model (like random guess) is almost zero (some people may mistakenly consider it as 0.5). For example, let's imagine someone gives random revenue prediction R_P and professional analysts give consensus R_C before the revenue announcement of the company. After the company announces the revenue R, we get the consensus's error (unexpected revenue) $err = R - R_C$. According to the definition of BC, we check whether R_P locates in $[R - |err|, R + |err|]$ which means that random prediction is closer to revenue than consensus, and the same time we also need to check whether both R_P and R are larger or smaller than R_C which means the direction of prediction is correct. Due to the prediction R_P can be arbitrary value by random guess, the probability that R_P locates in $[R - |err|, R + |err|]$ is quite low (and can be zero). So the BA of random prediction is almost 0 (not 0.5).

We also need a quantitative metric to evaluate the model's prediction performance compared with analysts' consensus. Thus we define another metric **SR (Surprise Ratio)** as follows:

Definition II.2. *SR (Surprise Ratio) SR is defined as* $SR = \frac{|\hat{UR} - UR|}{|UR|}$.

$|\hat{UR} - UR| = |\hat{R} - R|$ represents the absolute error between the predicted revenue and the actual revenue of a company. If $SR > 1$, the absolute error between the predicted revenue and the actual revenue is larger than the one between the analysts' consensus and the actual revenue; and vice versa. So the smaller SR, the closer a model's predicted revenue is to actual revenue than analysts' consensus. Given a set of companies, we can further compute the average of SR.

Given a set of data, we use two metrics **BA (Bounded Accuracy)** and **SR (Surprise Ratio)** to evaluate a model's performance. The higher BA, the lower SR, the better the prediction result of the model is. A high BA score means that the model can predict the correct direction of the unexpected revenue within bounded error with high possibility. And a low

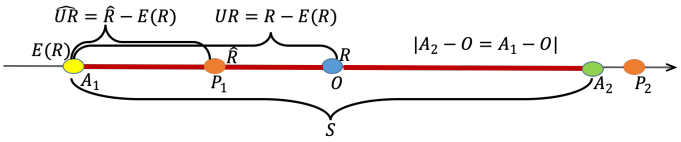


Fig. 2. An illustration of BC metric. If $\hat{R}$ locates in range S (like P_1), Bounded Correction(BC) is 1. If $\hat{R}$ (like P_2) locates out of range S, BC is 0.

SR score indicates that model's predicted revenue is closer to actual revenue than analysts' consensus.

C. Problem Definition

Our objective is predicting the unexpected revenue of a company using its historical financial data and alternative data. For the financial data of a company, we use two types of data: the historical revenue data and the analyst's consensus.

Formally, we have a set of data that $\{R_i^t, SE_i^t, SA_i^t\}$ where R_i^t is the revenue of company i at quarter t, SE_i^t is a set of estimated revenues of this company at quarter t by different analysts, and SA_i^t is a set of alternative data of the company at quarter t. For a set of estimated revenue SE_i^t, we extract the mean of the estimation E_i^t which is also named analysts' consensus, the lowest estimation LE_i^t and the highest estimation HE_i^t as *estimation features vector* $VE_i^t = (E_i^t, LE_i^t, HE_i^t)$. For a set of alternative data, we can only obtain an aggregate value A_i^t which is the total value of the whole alternative data SA_i^t (e.g. we cannot obtain the position of each user for privacy issues, but we can get the total number of users to a store in a quarter). We denote such aggregated value A_i^t as *alternative data features*. Therefore, the features of a company at quarter t can be expressed as $C_i^t = \{R_i^t, VE_i^t, A_i^t\}$. Then the unexpected revenue prediction task can be defined as:

Definition II.3 (Unexpected Revenue Prediction)**.** *A company has a sequence of features* $C_i^{t-k:t-1} = C_i^{t-k}, C_i^{t-k+1}, ..., C_i^{t-1}$ *and* (VE_i^t, A_i^t) *where* VE_i^t *is the estimation features at quarter t, and* A_i^t *is the alternative data features at quarter t. We define the financial features of the company at quarter t as* $X_i^t = \{C_i^{t-k:t-1}, VE_i^t, A_i^t\}$*. Our objective is to train a predictive model* $f(\cdot)$ *which can predict the unexpected revenue* $UR_i^t = R_i^t - E_i^t$ *that* $\hat{UR}_i^t = f(X_i^t)$*.*

Since every symbol in the above definition has a timestamp t, by default we omit the superscript t hereafter for convenience.

D. Alternative Datasets and Feature Extraction

The alternative data is the information gathered beyond traditional financial and economic sources. We use two real-world alternative datasets to evaluate our model.

- **Online transaction amount** is the amount of online credit card transactions collected by the China UnionPay. For each company, we collected the sum of transaction amount in each quarter. In our dataset, there are 71 companies with transaction amount features collected from 2014q3 to 2018q2, namely 16 quarters of data in total.
- **Map query** is the number of people querying a particular place, and such data is collected by Baidu Maps. Usually a map query of a store on online map service indicates a visit of this store by this user. It is important to many types of business as higher visitation number of users can lead to higher sales. For each company, we collect two kinds of data – map query to store and to parking lot, and separately collect the sum of the data in each quarter. Store refers to the offline stores and parking lot refers to car parking area of a company. There are 62 companies with both map query to store and to parking lot. We have 9 quarters of companies' map query data from 2016q2 to 2018q2.

There are almost no overlap companies between the two datasets. In addition, such alternative data can be noisy due to the uncertainties in data collection and processing. It is nontrivial to use data mining and machine learning techniques to analyze alternative data, as we can see the YoY and QoQ (Section IV-B)'s poor performance from Table I and Table II. This indicates the prediction of unexpected revenue requires well-designed models.

For each company, we have its historical financial features including revenues R_i^t and historical analysts' estimations $VE_i^t = (E_i^t, LE_i^t, HE_i^t)$ which are the mean, lowest and highest estimation of its revenue at each quarter. For a company, each analyst may give more than one estimations of its revenue. We adopt his/her estimation at end of the company's fiscal quarter (which is before revenue announcement) into the dataset. In addition, we apply one-hot coding for the quarter, month and sector of the company as additional features.

As introduced in Section II-C, the financial features of the company i at quarter t is $X_i^t = \{C_i^{t-k:t-1}, VE_i^t, A_i^t\}$. In our experiment, we set $k = 4$ in order to guarantee X_i^t to have at least features of one year. Since each sample should contain historical information for one year. All data is normalized by dividing the value of the oldest features in order to capture relative changes in price features and alternative data features, e.g. we normalize $R_i^{t-k:t}$ by dividing R_i^{t-k}, $A_i^{t-k:t}$ by dividing A_i^{t-k}. In order to strictly ensure that any information in the future is not used, we normalize dataset with the mean and variance from the training set in each cross-validation step. All of the models share the same features talked above.

III. Adaptive Master-Slave Regularized Model

A. Framework overview

In this section, we first provide an overview of our AMS framework. As shown in Figure 1, to forecast the unexpected revenue, the first step is slave-model generation, where the master model generates an adaptive slave-model for a company. Then the company's financial features are fed into the slave-model as input to predict its unexpected revenue.

In this paper, we use linear regression as the slave-model, because: 1) linear regression has good interpretability for

analysts to observe the composition of the final prediction result; and 2) linear regression has a simple structure which can be handled by the master model. Therefore, hereafter we also denote the slave-model as slave-LR. Note that in our model, each company has its own unique slave-LR model for predicting its unexpected revenue.

Figure 3 shows the whole process of AMS to generate the slave-LR model for each company through the master model. As shown in Figure 3, given a company X_i, the company's information is transformed by a *node transform* operation. Then under the structure defined by the *company correlation graph*, the company i is fed into a graph neural network to learn a node representation. Following that, the slave-model generation step will generate an adaptive slave-LR for each company to predict the unexpected revenue. In AMS, the master model has three main components, which are 1) graph neural network on company correlation graph, 2) anchored LR model, and 3) slave-model generation. We will introduce the detail of the AMS model in the following sections.

B. Node transformation

Before inputting a company into the GNN, we first conduct a node transformation on the company's financial features. Therefore, we use several forward layers to map the financial features of each company into the same feature space. For each layer of the node transformation, we use the following transformation function:

$$X_i' = \phi^{nt}(W^{nt}X_i) \tag{1}$$

where $X_i \in \mathbb{R}^F, X_i' \in \mathbb{R}^{F'}$ and $W^{nt} \in \mathbb{R}^{F' \times F}$. ϕ^{nt} is the activation function and we use $ReLu(\cdot)$ (Rectified Linear Unit) as nonlinear activation function. After several such forward layers, the final output features of the company i after the node transformation are defined as $nt(X_i)$. Note that node transformation is trained in an end-to-end process as part of the AMS model.

C. Graph neural network on company correlation graph

To generate the slave-model for each company, besides the company's financial features, we further use the company graph structure information to help generate the slave-LR model. Companies who share a similar trend of historical revenue may have some potential relationships. Thus, exploiting such graph structure can help the master model to generate a more effective slave-LR model.

Figure 4 shows an illustration of a company correlation graph. In the graph, the node represents a company, and the edge of the graph represents that two companies have high revenue correlation in history. For a company A, we first calculate the Pearson Correlation about revenue in history between company A and all other companies. Then we select k companies with the largest correlations with company A, and add an edge between the company A and these top k companies. Here k is a hyperparameter. Note that we only use the historical revenue to build the graph at every time series cross-validation step to avoid data leakage.

Upon the company correlation graph, we build a graph neural network to learn the representation of each company. The representation of a company should take the relations among the correlation graph into consideration for unexpected revenue prediction. We adopt a graph attention neural network (GAT) [17] on the company correlation graph to learn a representation of the company based on the company graph structure. The GAT can transform the input features of the company into high-level representations which can be used to generate better slave-models. Note that we use a supervised method to learn such representations by GAT, and the whole training process of the GAT is discussed in Section III-F

The reasons to adopt GAT as the GNN model to learn a company's features from the company correlation graph are 1) GAT is the state-of-the-art GNN model for representation learning on graph; 2) GAT can be applied to graph nodes having different degrees which is just a property of the company correlation graph.

The building block layer of GAT is the graph attention layer. For each graph attention layer, the input of the layer is a set of node features $S = \{X_1, X_2, ..., X_n\}, X_n \in \mathbb{R}^F$, where n is the number of companies and F is the number of features of each node.[1] If it is the first graph attention layer, then F is just the number of node features after node transformation. The graph attention layer outputs a new set of node features that $S' = \{X_1', X_2', ..., X_n'\}, X_n' \in \mathbb{R}^{F'}$, and it is possible $F \neq F'$. For each layer, we define a shared weighted transformation matrix $W^g \in \mathbb{R}^{F' \times F}$. If node j is adjacent to node i in the correlation graph, then we use the shared attention mechanism to compute attention coefficients $a_{ij} = softmax_j(e_{ij}) = softmax_j(a(W^gX_i, W^gX_j))$ to indicate the impact of node j's features to node i, where the attention mechanism a is a single-layer feedforward neural network with activation function [17] and $softmax$ is a normalized exponential function named softmax function . The final output features of each company is:

$$X_i' = \phi^g(\sum_{j \in \mathfrak{N}_i} a_{ij} W^g X_j) \tag{2}$$

where $\mathfrak{N}_i$ is in the neighbourhood of company i in the company correlation graph, and ϕ^g is the activation function. In our experiment, we set ϕ^g as $ReLu(\cdot)$. In the graph attention layer, we also adopt the multi-head attention mechanism [17] which uses H independent attention transformation of Equation 2. In this case the final output of the graph attention layer is:

$$X_i' = \|_{h=1}^{H} \phi^g(\sum_{j \in \mathfrak{N}_i} a_{ij}^h W_h^g X_j) \tag{3}$$

where $\|$ represents concatenation. The final output layer of GAT is a single attention head layer.

To sum up, given a set of the company S (organized as a correlation graph), the final output features of the company i after the GNN is denoted as $g(x_i) = GNN(nt(X_i)|S)$.

[1]Here the input features X_i of GAT is actually $nt(X_i)$, which is the output of node transformation function after several layer transform of $\phi^{nt}(W^{nt}X_i)$, however, we still use X_i to denote the input node features to avoid introducing too many unnecessary symbols.

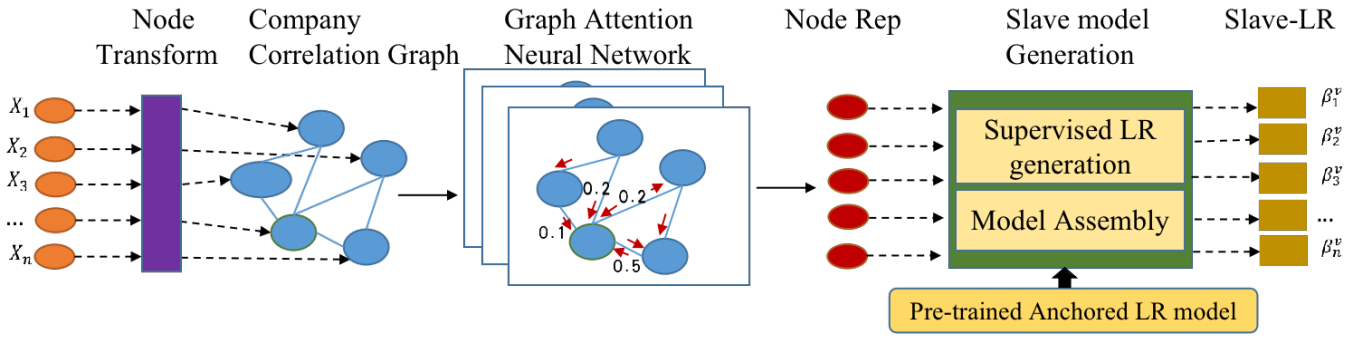


Fig. 3. The process of AMS model.

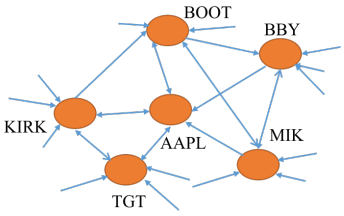


Fig. 4. An illustration of correlation graph ($k = 5$).

D. Anchored LR model training

In the master model, we also train an anchored linear regression model on the whole training data. The anchored LR is just a common LR model assuming the relationship between the dependent variable UR_i and the financial vector X_i is linear, which has the form:

$$\hat{UR_i} = X_i^T B_{acr} \tag{4}$$

To optimize the parameter of B_{acr}, we should minimize the optimization function Γ_{acr} defined as follows:

$$\Gamma_{acr} = \min \frac{1}{2N} \sum_{i=1}^{N} ||X_i^T B_{acr} - UR_i||^2 + \frac{1}{2}\lambda ||B_{acr}||^2 \tag{5}$$

As we explain in next section, we will add a regularization term for generating slave-models to ensure the parameters of the slave models are consistent with B_{acr} as much as possible. This is just the reason we call this LR model as an "anchored" LR which is further discussed in Section III-E1.

E. Slave-model generation

After learning the node representation of a company from GNN, as shown in Figure 3, the next step is to generate the slave-model for this company. The objective of this step is to generate an adaptive linear regression slave-model for each company. Though it is simple, the linear regression method has shown great potential in many financial predictive applications. One reason is that it is hard to train a complex model due to the limited amount of financial data, and another reason is that the LR has good interpretability which is important for the financial application scenario.

In our model, given a company i with financial features X_i, we will generate an adaptive linear regression model for this company. The model generation is based on the representation of the company learned by GNN in previous section, which can be expressed as:

$$\beta_v(X_i) = M(g(X_i)) = M(GNN(X_i|S)) \tag{6}$$

where the predicted unexpected revenue of the company i is $\hat{UR_i} = X_i^T \beta_v(X_i)$.

Now the key problem is how to train a generation model M to predict the slave-LR model. We first define a nonlinear transformation model with each layer having the transformation function $X_i' = \phi^m(W^m X_i)$.[2] The final output of $M(\cdot)$ is $\beta_v(X_i) = W^m X_i$. Note that there is no activation function in the final layer since our objective is to predict the parameters of the slave-LR model.

The straightforward way to optimize $M(\cdot)$ is to optimize the following objective function:

$$\Gamma_1 = \min \frac{1}{2N} \sum_{i=1}^{N} ||UR_i - X_i^T M(g(X_i))||^2 + \frac{1}{2}\lambda_1 ||M \cdot g||^2 \tag{7}$$

However, simply predicting a linear regression model for a company can easily lead to overfitting in the training data since $\beta_v(X_i)$ has too much flexibility as a model for X_i. We propose two regularization techniques to tackle this problem, which are called supervised LR generation and model assembly, respectively.

1) Supervised LR generation: The first regularization technique is called "supervised" LR generation which restricts the parameter space of the slave-LR's parameters. Its essential idea is that we make the generated LR's parameter $\beta_v(X_i)$ approach to parameters of anchored LR B_{acr} as much as possible. To achieve this purpose, we add a regularization term Γ_{slg} on the objective function to optimize the model for LR generation, which is:

$$\begin{aligned} \Gamma_{slg} &= \min \frac{1}{2N} \sum_{i=1}^{N} ||\beta_v(X_i) - B_{acr}||^2 \\ &= \min \frac{1}{2N} \sum_{i=1}^{N} ||M(g(X_i)) - B_{acr}||^2 \end{aligned} \tag{8}$$

And the objective function can be formulated as:

$$\Gamma_2 = \Gamma_1 + \lambda_{slg} \Gamma_{slg} \tag{9}$$

[2]Here the input features X_i is actually the output of GAT, however, we still use X_i to denote the input node features to avoid introducing too many unnecessary symbols.

The reasons to devise such a supervised LR generation technique can be seen from two perspectives: 1) the parameter of B_{acr} is already optimized over all the training dataset which can make $\beta_v(X_i)$ be optimized only in near-optimal parameter space; 2) the $\beta_v(X_i)$ can be adjustable with its features to search a better parameter than the ones of B_{acr}.

2) Model assembly: The second regularization technique to avoid overfitting of the slave-LR is called model assembly which assumes $M(\cdot)$ is assembled by an adaptively generated LR model and a globally optimized LR model. In this case, the generated slave-LR model can be expressed as:

$$\beta_v(X_i) = \gamma M(g(X_i)) + (1-\gamma)\beta_c \tag{10}$$

The advantage of model assembly is that it can balance the contradiction between the model's flexibility (which may result in overfitting) and inadaptability (which is not adaptive to different companies). γ is a hyperparameter and it makes a trade-off between the fully adapted slave-LR model and the solely globally optimized LR model. Therefore, the model assembly technique can stabilize the flexibility of the generated slave-LR model.

To combine the supervised LR generation and model assembly, the final objective function is:

$$\begin{aligned}\Gamma_{master} = \min \frac{1}{2N}\sum_{i=1}^{N} ||UR_i - X_i^T(\gamma M(g(X_i)) + (1-\gamma)\beta_c)||^2 \\ + \min \lambda_{slg}\frac{1}{2N}\sum_{i=1}^{N} ||M(g(X_i)) - B_{acr}||^2 \\ + \frac{1}{2}\lambda_1(||M\cdot g||^2 + ||\beta_c||^2)\end{aligned} \tag{11}$$

where B_{acr} is a pre-trained constant vector; γ, λ_{slg}, λ_1 are hyperparameters, and the last term of the objective function is L2 regularization.

F. Training Process of AMS

We summarize AMS's training process in this section. There are two steps to train the master model for generating the slave-LR model. The first step is to train a globally optimized anchored LR B_{acr} by minimizing the objective function Γ_{acr} defined in Equation 5.

After obtaining the parameter of B_{acr}, the next step is to optimize the objective function Γ_{master} as defined in Equation 11. We use gradient descent with Adam [18] to optimize parameters of the model. Note that all the representations learned by node transformation and GAT are dependent on the optimization of the objective function Γ_{master}. In other words, there are three sets of parameters to be optimized through minimizing Γ_{master}: 1) the parameters for node transformation; 2) the parameters of GAT on company correlation graph; and 3) the parameters for slave-model generation.

IV. Experiments

A. Environment and settings

Experiments are conducted on a GPU-CPU platform. The GPU is Tesla P40, and the CPU is Intel Xeon Gold 5117 with 206 GB memory. All the program and baselines are implemented in Python 3.6. The deep learning models are implemented in PaddlePaddle[3].

B. Baselines

We compared AMS with several baselines which can be classified into three groups including: 1) XGBoost, three linear model variants (Lasso, Ridge, and Elasticnet) which have good interpretability; 2) Multilayer Perceptron (MLP) and neural sequence models (LSTM and GRU) which have high capacity and can capture temporal dependency; 3) statistical model ARIMA which uses the historical revenue for prediction, and QoQ/YoY which uses alternative dataset. The details are as follows:

- **XGBoost** [19] is a scalable implementation of Gradient Boosting Decision Tree algorithm. We set the parameter of "objective" in XGBoost as "reg:linear" to build a regression model for unexpected revenue prediction.
- **MLP** (Multilayer Perceptron) is a multilayer feedforward neural network. In our experiment, each node is a neuron that uses a nonlinear activation function $ReLu(\cdot)$. This model has a greater capacity than linear regression but is uninterpretable.
- **Lasso** (Least Absolute Shrinkage and Selection Operator) is a variant of linear regression model with L1 regularization.
- **Ridge** is a variant of linear regression model with L2 regularization.
- **Elasticnet** is a variant linear regression model with L1 and L2 regularization.
- **LSTM** [20] (Long Short-Term Memory) is a sequence deep learning model designed to efficiently capture long-term and short-term dependencies through gating mechanism.
- **GRU** [21] (Gated Recurrent Unit) is a popular variant of LSTM which has less parameters.
- **ARIMA** [22] (Autoregressive Integrated Moving Average model) is a popular statistical model for time series forecasting.
- **QoQ (Quarter to Quarter)** uses the relative change amount of the alternative feature from quarter to quarter to predict the unexpected revenue, which is is defined as $\frac{A_i^t}{A_i^{t-1}}R_i^{t-1} - E_i^t$.
- **YoY (Year to Year)** uses the relative change amount of the alternative feature from year to year to predict the unexpected revenue, which is defined as $\frac{A_i^t}{A_i^{t-4}}R_i^{t-4} - E_i^t$.

C. Training Method

The random search strategy [23] is adopted on validation data to determine the optimal hyperparameters. Models are optimized with Adam [18]. We repeat 10 times of experiments for training AMS model, and the average running time for optimizing AMS is 771 seconds. Dropout [24] is applied to models who have stacked fully connected layers like AMS and

[3]http://www.paddlepaddle.org/

MLP. Same as AMS, we use L2 regularization technique to other models.

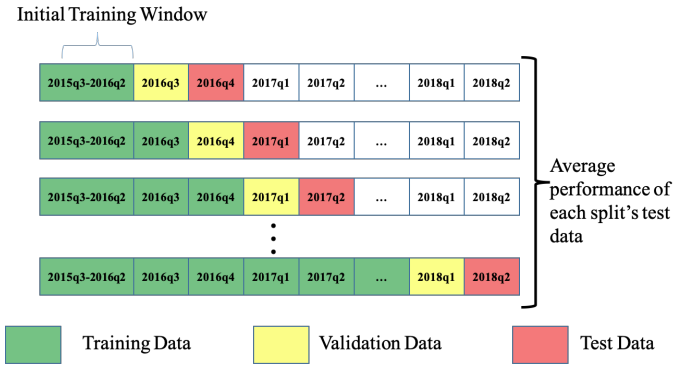


Fig. 5. Time series cross-validation at online transaction amount dataset.

We use time series cross-validation (CV) to evaluate the model's performance. Figure 5 is an example on online transaction dataset. For online transaction amount dataset, we drop data from 2014q3 to 2015q2 due to the absence of historical information of one year. At the initial step, we hold one's year historical data from 2015q3 to 2016q2 for training due to sufficient data for rolling. We use data in 2016q3 for validation and data in 2016q4 for testing. Next step, we use data before 2016q4 for training, data in 2016q4 for validation, and data in 2017q1 for testing. In this way, we can evaluate models' performance in the last seven quarters from 2016q4 to 2018q2. For map query dataset, we also drop data from 2016q2 to 2017q1 due to the absence of historical information of one year. Then we use data in 2017q2, 2017q3 as initial training data, data in 2017q4 for validation, and data in 2018q1 for testing. Next step, we use data in 2017q2, 2017q3, 2017q4 for training, data in 2018q1 for validation, and data in 2018q2 for testing. Then we can evaluate models' performance in the last two quarters. To sum up, we evaluate models on online transaction amount dataset from 2016q4 to 2018q2 and map query dataset from 2018q1 to 2018q2.

D. Model Performance

The experiments on BA and SR are summarized in Table I and Table II. Since companies almost have no overlap between the transaction amount and map query dataset, there is no experiment on companies with both map query and transaction amount data.

We conducted a significance test – pairwise t-test – between AMS and baselines with regard to BA on transaction amount data. The P-values are shown in Table I. We find that AMS significantly outperforms all the baselines. Due to the limit time length of map query dataset, we only have CV results on 2018q1 and 2018q2. Thus we cannot conduct the significance test on map query data. We directly show the model's results on 2018q1 and 2018q2. On transaction amount dataset, the BA of AMS is larger than 58%, but none of the baselines is larger than 53%. On map query dataset, AMS also achieves highest BA 57.258%.

TABLE I
PERFORMANCE COMPARISON ON BA (BOUNDED ACCURACY). THE FIRST COLUMN OF BA ON EACH DATASET IS THE AVERAGE OF CROSS VALIDATION RESULTS. THE FIRST (SECOND) LINE OF YOY/QOQ ON MAP QUERY DATASET IS THE RESULT OF YOY/QOQ WITH MAP QUERY TO STORE (PARKING LOT).

Dataset	transaction amount		map query		
Model	BA	P-value	BA	BA(18q1)	BA(18q2)
AMS	**58.551**	-	**57.258**	**54.838**	**59.677**
XGBoost	50.503	0.0179	45.967	45.161	46.774
MLP	51.307	0.0134	47.580	48.387	46.774
Lasso	48.088	0.0454	38.709	41.935	35.483
Ridge	52.515	0.0009	45.967	48.387	43.548
Elasticnet	52.917	0.0396	45.161	43.548	46.774
Lstm	51.710	0.0121	54.032	50.000	58.064
GRU	50.905	0.0233	50.806	51.612	50.000
ARIMA	12.273	<1e-4	15.322	17.741	12.903
YoY	28.772	<1e-4	11.290	11.290	11.290
			15.322	9.677	20.967
QoQ	27.692	<1e-4	20.161	22.580	17.741
			15.322	19.354	11.290

Table II illustrates the comparison results under SR metric. For transaction amount dataset, in order to measure whether models are significantly better than analysts' consensus, we apply the pairwise t-test between models and analysts' consensus with regard to SR on CV results. Except QoQ, YoY and ARIMA are significantly worse than analysts' consensus (P-value is small but SR is much larger than 1), AMS is the only one that obtains a substantially small p-value toward significance but other models' P-values are larger than 0.48. Besides, AMS achieves the best performance (with lowest SR) among all baselines. As we can see from Table II, AMS' SR is substantially smaller than 1. This means that AMS using transaction amount and map query data can beat professional analysts.

From the experimental results, we can find that ARIMA gets poor performance. The reason is that though ARIMA can incorporate both autoregressive and moving average features with removing trend of the data. However, a company's revenue is usually affected by complex factors, which cannot be modeled by ARIMA. What is more, simply using the QoQ and YoY of transaction amount features are useful but still far worse than other models. It means that predicting unexpected revenue with transaction amount data still requires nontrivial models. Different from the results of transaction amount dataset, QoQ and YoY with map query features get much worse results. This shows that the usability of map query data is not as straightforward as the transaction amount data. Furthermore, models, such as XGBoost, MLP, LSTM and GRU, also get worse results compared with our model. This demonstrates that our AMS can better capture the information of transaction amount and map query than baselines. The results show that AMS enhanced by the graph neural network model can fully utilize these features' information to improve the prediction performance.

E. Effectiveness of Alternative Data

We also conducted an experiment to demonstrate the effectiveness of the alternative data used in our model. For

TABLE II
PERFORMANCE COMPARISON ON SR (SURPRISE RATIO). THE FIRST COLUMN OF SR ON EACH DATASET IS THE AVERAGE OF CROSS VALIDATION RESULTS. THE FIRST (SECOND) LINE OF YOY/QOQ ON MAP QUERY DATASET IS THE RESULT OF YOY/QOQ WITH MAP QUERY TO STORE (PARKING LOT).

Dataset	transaction amount		map query		
Model	SR	P-value	SR	SR(18q1)	SR(18q2)
AMS	**0.9603**	**0.0657**	**0.9626**	1.0071	**0.9180**
XGBoost	1.0177	0.5940	0.9989	1.0049	0.9929
MLP	0.9888	0.7009	1.0165	1.0205	1.0125
Lasso	1.0045	0.7264	1.0114	1.0111	1.0117
Ridge	0.9925	0.8200	1.0393	1.0625	1.0161
Elasticnet	0.9921	0.5613	1.0077	1.0185	0.9968
Lstm	0.9827	0.4892	0.9978	0.9987	0.9970
GRU	0.9913	0.7263	1.0178	1.0088	1.0267
ARIMA	5.8950	<1e-4	4.3795	3.9150	4.8439
YoY	2.4771	<1e-4	5.2282	5.6360	4.8203
			5.3544	6.2700	4.4389
QoQ	4.3493	0.0076	3.4215	2.8067	4.0363
			3.9063	3.1906	4.6220

this purpose, we removed all alternative features from the data and re-trained models to run the experiments in previous sections. We add the suffix **-na** to represent the model without alternative data. In order to compare the model performance with and without alternative data, we use the SR-m and BA-m to denote the result of the SR and BA without alternative data minus the SR and BA with alternative data, e.g. SR-m of AMS-na is just got by SR of AMS-na minus SR of AMS. Note that the larger the value of SR-m and the smaller the value of BA-m, the more useful of the alternative features.

TABLE III
FEATURES EFFECTIVENESS AMONG VARIOUS MODELS. THE SUFFIX **-NA** IS USED TO REPRESENT THE MODEL WITHOUT ALTERNATIVE DATA. THE SR-M AND BA-M DENOTE THE RESULT OF THE SR AND BA WITHOUT ALTERNATIVE DATA MINUS THE SR AND BA WITH ALTERNATIVE DATA.

Dataset	transaction amount		map query	
Model	SR-m	BA-m(%)	SR-m	BA-m(%)
AMS-na	0.0269	-5.633	0.0377	-6.451
XGBoost-na	-0.0013	-6.438	0.0506	-4.032
MLP-na	0.0125	-2.414	-0.0127	3.225
Lasso-na	0.0000	0.000	0.0000	0.000
Ridge-na	0.0143	-2.816	-0.0167	0.806
Elasticnet-na	0.0031	-0.804	0.0000	0.000
Lstm-na	0.0092	-0.804	0.0026	-4.838
GRU-na	-0.0070	-1.408	-0.0119	-3.225

For transaction amount dataset in Table III, all models get worse on BA and most of the models get worse on SR without transaction amount features. XGBoost-na and GRU-na get better results on SR but the improvement is too small to show a significantly better performance than Table II. Overall, models get worse performance which shows the effectiveness of the transaction amount data. It is not unexpected that AMS's performance has dropped dramatically without transaction amount features. This not only shows that transaction amount is effective, but also shows that our model can make full use of it. Although XGBoost-na also gets much worse results. But in Table I XGBoost gets bad performance as well, which shows XGBoost is not a good model for this problem.

For map query dataset in Table III, AMS's performance also has dropped dramatically without map query features, which proves that AMS can fully exploit the value of alternative data. From the results on map query, MLP-na and Ridge-na get better performance. One possible reason is that both of the fully connected models (Ridge and MLP) cannot exploit the value of map query data (Ridge can be viewed as one layer L2 regularized MLP without activation function). In this case, map query data is noisy for Ridge and MLP.

It shows that all baselines are not able to fully utilize the information from alternative data. Without the information from alternative features, it is hard to predict a useful unexpected revenue because baselines can only fit a rule of experience from historical data. We also can observe that Lasso and Elasticnet in some cases get the same results without alternative features. An explanation of this phenomenon is that Lasso with L1 regularization encourages the model to select fewer features for prediction. The alternative features' relationship with current revenue is not as strong as analysts' consensus and historical revenue. Thus, Lasso chooses to discard alternative features. Elasticnet shares the same case with Lasso.

F. An Application of the Model – Backtest

In this section, we conduct several backtests to showcase the usability of our proposed model for investment, and demonstrate the superiority of our AMS model compared with baselines enhanced with the alternative dataset.

The stock price reflects the current expectations of the company's revenue. Based on whether the predicted unexpected revenue showing a positive surprise, we can judge whether the company's current revenue is underestimated by the market. In this case, the company's stock price may be undervalued. Thus after the revenue report is announced, the stock price will rise to a normal value. If the revenue is underestimated, our strategy is to buy the stock on long position, i.e. to buy stocks at end of the company's fiscal quarter (which is before revenue announcement) and sell them a month later. If the revenue is overestimated, our strategy is to short sell the stock, i.e. to sell the stock at the end of the company's fiscal quarter and buy it back a month later. In reality, the fundings a company's stock attracts is positively correlated with its market capitalization. In this paper, we simply classify companies into three categories based on their market capitalization with two boundaries of 1 billion and 10 billion and set the ratio of money for investing as 1:2:3 to different categories accordingly.

In order to compare their performance, we calculate the earning, Max Drawdown (MDD), Sharpe Ratio and Excess Return (ER) during the whole trading period. MDD measures the maximum loss of assets in the whole quarter from a peak to a through of a given portfolio strategy defined as:

$$MDD = \max_{l \in (0,T)} [\max_{t \in (0,l)} [S_t - S_l]]$$

where S_t is the assets and T is end of the date. The smaller the MDD means the strategy performs better. Sharpe Ratio is

a risk-adjusted profit measure which measures the return per unit of deviation relative to our AMS model defined as:

$$SharpeRatio = \frac{AVG(R_B - R_{AMS})}{STD(R_B - R_{AMS})}$$

where R_{AMS} is time series vector of daily return given by AMS, R_B is time series vector of daily return given by baseline model B, AVG is average of the vector and STD is the standard deviation of the vector. If Sharpe Ratio is smaller than 0, it means the strategy on method B cannot obtain excess return than the one on AMS.

In order to judge whether AMS is better than other baselines at every end of the quarter, we calculate Average Excess Return (AER) in this paper. Excess Return is the excess earning baselines earns over AMS at the end of the quarter. AER is average of the Excess Return at every end of the quarter.

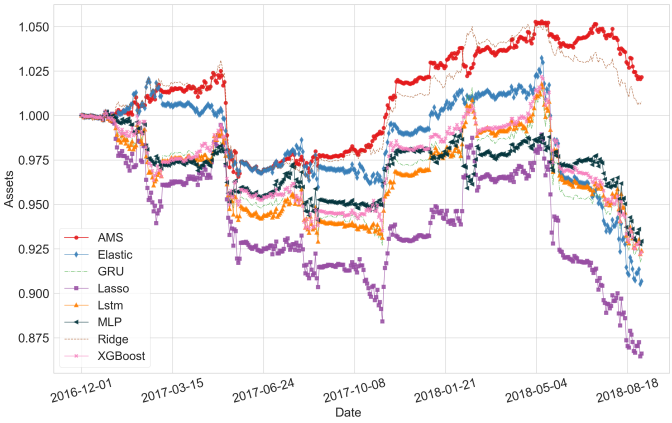


Fig. 6. Strategy performance on transaction amount dataset from 2016q4 to 2018q2.

TABLE IV
BACKTEST RESULTS FROM 2016Q4 TO 2018Q2 ON TRANSACTION AMOUNT DATASET.

Model	Earning(%)	MDD(%)	Sharpe Ratio	AER(%)
AMS	**2.1707**	**5.8010**	-	-
XGBoost	-7.6668	9.7786	-0.0706	-4.8013
MLP	-7.0645	7.4250	-0.0737	-4.8435
Lasso	-13.3881	13.7267	-0.0770	-8.4646
Ridge	0.7887	6.5520	-0.0176	-0.2632
Elasticnet	-9.3145	12.3236	-0.0835	-3.6364
Lstm	-7.5981	9.3912	-0.0639	-5.3055
GRU	-8.0377	9.7075	-0.0629	-5.3068

As from Table IV, AMS can beat all of the baselines with highest earning 2.1707% and lowest MDD 5.8010%. In terms of Sharpe Ratio, what the best baseline can achieve is -0.0176, still lower than 0. Therefore, the strategies based on all baseline models do not obtain excess returns over the one on AMS. Furthermore, the result of AER shows that our model's average performance at different quarters during the whole trading period can beat all baselines. Figure 6 shows that asset curve given by AMS exceeds other baselines at most of the time. This is also consistent with the result in Section IV-D where AMS achieves highest BA and lowest SR.

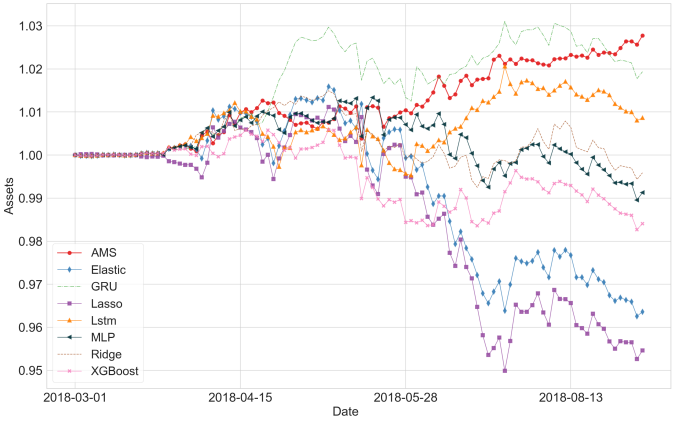


Fig. 7. Strategy performance on map query dataset from 2018q1 to 2018q2.

TABLE V
BACKTEST RESULTS FROM 2018Q1 TO 2018Q2 ON MAP QUERY DATASET.

Model	Earning(%)	MDD(%)	Sharpe Ratio	AER(%)
AMS	**2.7772**	**0.8687**	-	-
XGBoost	-1.5919	2.3056	-0.2185	-3.4804
MLP	-0.8722	2.3462	-0.2319	-2.0195
Lasso	-4.5401	6.0639	-0.1461	-4.4512
Ridge	-0.4117	2.2115	-0.1632	-2.1341
Elasticnet	-3.6408	5.2510	-0.1395	-3.7569
Lstm	0.8608	1.6638	-0.0751	-1.7009
GRU	1.9363	1.6707	-0.0326	-0.2810

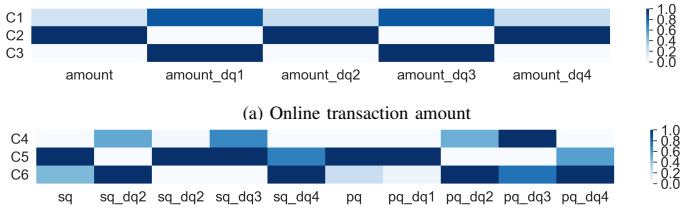
The backtest results on map query dataset are shown in Table V and Figure 7. It is not unexpected that AMS reaches the highest earning 2.7772% and lowest MDD 0.8687%. Besides, the best sharpe ratio of baselines is -0.0730. Therefore, with map query data the strategy on all baseline models cannot obtain excess returns over the one on AMS. Figure 7 shows that stable asset curve given by AMS. This result is also consistent with the result in Table I that AMS achieves high BA at both of the quarters.

Since different dataset's companies and trading period are different, it is meaningless to compare each model's performance on Earning, MDD, Sharpe Ratio and AER between transaction amount dataset and map query dataset. Note that the earning rankings of baselines are not strictly consistent with the BA rankings of baselines in Table I. In a real-world scenario, the market does not always change in the direction we expect, due to company management, company's expectation for next quarter, news, market sentiment, etc. But the more accurately our model predicts the unexpected revenue, the higher the expectation of profit we can get.

G. Interpretability of AMS

Different from traditional deep learning models, a key advantage of our AMS model is the interpretability of the final linear regression's adaptive weight produced by the master model. Here we output the companies' weight in the slave-LR model on the test dataset to illustrate the interpretability of AMS in Figure 8. In order to highlight the difference between different companies' weight in the same feature and for the convenience of visualization, we linearly scale the value along

with the feature to $[0, 1]$ in selected companies. We randomly selected three companies (C) on each dataset.



(a) Online transaction amount

(b) Map query to store (sq) and to parking lot (pq)

Fig. 8. Visualization of features' weight produced by AMS's master model on different dataset. Features with suffix dqk represent features in k quarters ago.

The different weights of alternative features among companies are shown in Figure 8. We can find that, different from the traditional linear regression with fixed weight for the same feature dimension, features' weight produced by AMS is not the same among companies. AMS utilizes the information in graph and company's features together to output a unique weight for each company. At the same time, the original features are kept for the final linear regression model, so the weight measures the outcome changes if increasing a small delta unit of each feature. From this point of view, the slave-model of AMS is interpretable because the impact of fluctuations in each unit of the feature can be interpretable on the final result.

V. Related Work

A. Alternative data in financial applications

Alternative data can be classified into three categories: data produced by individuals (such as social media posts), data generated through business processes (such as credit card or online payment apps transactions data), and data generated by sensors (such as GPS) [7]. How to utilize such alternative data for investment has attracted great research attention in recent years. However, most of research efforts are spent on how to extract stock market's alpha signals from the first category of data, i.e. data produced by individuals such as social media data [9], [10], web search query data [11], [12] and news data [13], [14]. Though there are some case studies about how to utilize the second and third category data (i.e. generated through business processes and generated by sensors) for investment [7], all of these methods simply aggregate the information from the data, and use the aggregated value as a reference indicator to help portfolio managers make decisions.

Some authors in their pioneering papers also studied how to use alternative data to forecast values of macro economic indicators like unemployment claims from web query data [25], population from map search query [26], poverty and wealth from mobile phone data [27] and satellite imagery data [28]. Such methods cannot be easily extended to support the analysis of company activities. Different from such methods, our model focuses on the micro economic data, i.e. unexpected revenue of a company. As far as we know, our model is the first attempt to utilize the business process data and map query data for unexpected revenue prediction via machine learning.

B. Adaptive prediction model

Another research topic related to our model is the adaptive prediction model. Most of existing adaptive prediction models are the models which can be adjusted after observing the "truth" of each individual. We name these models as "passive adaptive model" since it only updates the model after the behavior of the individual has changed. One kind of the machine learning algorithm for the passive adaptive models is based on online gradient descent, which simultaneously processes an observation and learns to perform better [29], [30]. Such solutions usually adaptively tune the model parameters as the data sequence is progressively revealed [31]. The passive adaptive prediction model is also related to transfer learning since the passive adaptive prediction algorithm adaptively modifies the model when different types of pattern drifts are detected [32], or adaptively updates the model when the environments and population are gradually changed. However, a key point of the passive adaptive model is that such models can only be adaptive for an individual when the ground truth is revealed and the algorithm suffers a loss. Another similar methods related with our master-slave model is the semi-lazy learning approach [33]–[35]. Instead of generating slave models by a master model, semi-lazy learning approach tries to optimize an individual model upon a small set of data points generated by any nearest neighbor search method like [36]. The weakness of such previous adaptive or semi-lazy learning models in our application is that it cannot have enough information to adaptively update the model since the financial data is usually very sparse.

Our AMS model tries to adaptively generate a slave-model for each individual, and the slave-model can consider the special characteristic for the individual. In this paper, we generate a linear regression model for each individual company. From this point of view, we can consider our adaptive model as "aggressive adaptive model" who adjusts the model even without seeing the ground truth. As far as we know, we are the first to study such aggressively adaptive model with the master-slave model architecture for the financial machine learning applications.

C. Graph neural network

In this paper, we use a graph neural network as the main component of the master model. The graph neural network has arisen at the intersection of deep learning and graphs recently. According to the functions for relational reasoning over the graph, the graph neural networks can be classified as full connect graph network [37], independent recurrent network [38], message-passing neural network [39], non-local neural network [17], relation network [40] and deep sets [41]. Since our main objective of graph neural network is to compact the graph structure information into the node (i.e. company) features, we use the non-local neural network to model the

relationship between companies. The graph attention mechanism is also used to handle the pairwise-interaction between companies. We refer readers to a comprehensive survey [42] for a detailed discussion about the graph neural network.

VI. Conclusion

In this paper, we propose AMS, an adaptive master-slave regularized model for unexpected revenue prediction using alternative data. The key idea of AMS is to employ a master model to generate a slave-LR model to predict the unexpected revenue of a company with alternative data. We devise several novel techniques in our model, including the GNN on company correlation graph, supervised slave-LR generation and model assembly. Besides, our AMS is more intepretable than other black-box deep learning approaches. The experiments conducted on transaction data and map query data demonstrate the effectiveness of our model.

VII. Acknowledgements

The research is supported in part by the National Natural Science Foundation of China Grant 91746301, 61822203, 71531001 and the Zhongguancun Haihua Institute for Frontier Information Technology and Turing AI Institute of Nanjing.

References

[1] K.-P. Lin, P.-F. Pai, Y.-M. Lu, and P.-T. Chang, "Revenue forecasting using a least-squares support vector regression model in a fuzzy environment," *Information Sciences*, vol. 220, pp. 196–209, 2013.

[2] Y. Ertimur, J. Livnat, and M. Martikainen, "Differential market reactions to revenue and expense surprises," *Review of Accounting Studies*, vol. 8, no. 2-3, pp. 185–211, 2003.

[3] N. Jegadeesh and J. Livnat, "Revenue surprises and stock returns," *Journal of Accounting and Economics*, vol. 41, no. 1-2, pp. 147–171, 2006.

[4] U. Chandra and B. T. Ro, "The role of revenue in firm valuation," *Accounting Horizons*, vol. 22, no. 2, pp. 199–222, 2008.

[5] I. Kama, "On the market reaction to revenue and earnings surprises," *Journal of Business Finance & Accounting*, vol. 36, no. 1-2, pp. 31–50, 2009.

[6] A. S. Manikas and P. C. Patel, "Managing sales surprise: The role of operational slack and volume flexibility," *International Journal of Production Economics*, vol. 179, pp. 101–116, 2016.

[7] M. Kolanovic and R. Krishnamachari, "Big data and ai strategies: Machine learning and alternative data approach to investing," *JP Morgan Global Quantitative & Derivatives Strategy Report*, 2017.

[8] A. H. Monk, M. Prins, and D. Rook, "Rethinking alternative data in institutional investment," *Available at SSRN: https://ssrn.com/abstract=3193805*, 2018.

[9] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of computational science*, vol. 2, no. 1, pp. 1–8, 2011.

[10] J. Si, A. Mukherjee, B. Liu, Q. Li, H. Li, and X. Deng, "Exploiting topic based twitter sentiment for stock prediction," in *ACL*, vol. 2, 2013, pp. 24–29.

[11] C. Curme, T. Preis, H. E. Stanley, and H. S. Moat, "Quantifying the semantics of search behavior before stock market moves," *PNAS*, vol. 111, no. 32, pp. 11 600–11 605, 2014.

[12] T. Dimpfl and S. Jank, "Can internet search queries help to predict stock market volatility?" *European Financial Management*, vol. 22, no. 2, pp. 171–192, 2016.

[13] X. Ding, Y. Zhang, T. Liu, and J. Duan, "Deep learning for event-driven stock prediction." in *IJCAI*, 2015, pp. 2327–2333.

[14] Z. Hu, W. Liu, J. Bian, X. Liu, and T.-Y. Liu, "Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction," in *WSDM*. ACM, 2018, pp. 261–269.

[15] V. Sokolov, "Discussion of 'deep learning for finance: deep portfolios'," *Applied Stochastic Models in Business and Industry*, vol. 33, no. 1, pp. 16–18, 2017.

[16] J. Heaton, N. Polson, and J. H. Witte, "Deep learning for finance: deep portfolios," *Applied Stochastic Models in Business and Industry*, vol. 33, no. 1, pp. 3–12, 2017.

[17] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *KDD*. ACM, 2016, pp. 785–794.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[22] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[23] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *JMLR*, vol. 13, no. Feb, pp. 281–305, 2012.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *JMLR*, vol. 15, no. 1, pp. 1929–1958, 2014.

[25] H. Choi and H. Varian, "Predicting the present with google trends," *Economic Record*, vol. 88, pp. 2–9, 2012.

[26] J. Zhou, H. Pei, and H. Wu, "Early warning of human crowds based on query data from baidu maps: Analysis based on shanghai stampede," in *Big Data Support of Urban Planning and Management*. Springer, 2018, pp. 19–41.

[27] J. Blumenstock, G. Cadamuro, and R. On, "Predicting poverty and wealth from mobile phone metadata," *Science*, vol. 350, no. 6264, pp. 1073–1076, 2015.

[28] N. Jean, M. Burke, M. Xie, W. M. Davis, D. B. Lobell, and S. Ermon, "Combining satellite imagery and machine learning to predict poverty," *Science*, vol. 353, no. 6301, pp. 790–794, 2016.

[29] L. Bottou, "Online learning and stochastic approximations," *On-line learning in neural networks*, vol. 17, no. 9, p. 142, 1998.

[30] E. Hazan, A. Rakhlin, and P. L. Bartlett, "Adaptive online gradient descent," in *NIPS*, 2008, pp. 65–72.

[31] P. Auer, N. Cesa-Bianchi, and C. Gentile, "Adaptive and self-confident on-line learning algorithms," *Journal of Computer and System Sciences*, vol. 64, no. 1, pp. 48–75, 2002.

[32] J. Liu and E. Zio, "An adaptive online learning approach for support vector regression: Online-svr-fid," *Mechanical Systems and Signal Processing*, vol. 76, pp. 796–809, 2016.

[33] J. Zhou and A. K. Tung, "Smiler: A semi-lazy time series prediction system for sensors," in *SIGMOD*. ACM, 2015, pp. 1871–1886.

[34] J. Zhou, A. K. Tung, W. Wu, and W. S. Ng, "A semi-lazy approach to probabilistic path prediction in dynamic environments," in *KDD*. ACM, 2013, pp. 748–756.

[35] ——, "R2-d2: a system to support probabilistic path prediction in dynamic environments via semi-lazy learning," *PVLDB*, vol. 6, no. 12, pp. 1366–1369, 2013.

[36] J. Zhou, Q. Guo, H. Jagadish, L. Krcal, S. Liu, W. Luan, A. K. Tung, Y. Yang, and Y. Zheng, "A generic inverted index framework for similarity search on the gpu," in *ICDE*. IEEE, 2018, pp. 893–904.

[37] J. B. Hamrick, K. R. Allen, V. Bapst, T. Zhu, K. R. McKee, J. B. Tenenbaum, and P. W. Battaglia, "Relational inductive bias for physical construction in humans and machines," in *CogSci*, 2018.

[38] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia, "Graph networks as learnable physics engines for inference and control," in *ICLR*, 2018.

[39] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," *arXiv preprint arXiv:1704.01212*, 2017.

[40] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, "A simple neural network module for relational reasoning," in *NIPS*, 2017, pp. 4967–4976.

[41] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *NIPS*, 2017, pp. 3391–3401.

[42] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.