Trading Desk Behavior Modeling via LSTM for Rogue Trading Fraud Detection

Marine Neyret[1], Jaouad Ouaggag[2] and Cédric Allain[1]

[1]*DataLab, Institut Louis Bachelier, Paris, France*

[2]*DataLab, Inspection Générale et Audit, Société Générale, Paris, France*

Keywords: Behavior Modeling, LSTM, Recurrent Neural Networks, Outliers Detection, Trading Activity, Rogue Trading Detection, Unexpected Behavior Detection.

Abstract: Rogue trading is a term used to designate a fraudulent trading activity and rogue traders refer to operators who take unauthorised positions with regard to the mandate of the desk to which they belong and to the regulations in force. Through this fraudulent behavior, a rogue trader exposes his group to operational and market risks that can lead to heavy financial losses and to financial and criminal sanctions. We present a two-step methodology to detect rogue trading activity among the deals of a desk. Using a dataset of transactions booked by operators, we first build time series behavioral features that describe their activity in order to predict these features' future values using a Long Short-Term Memory (LSTM) network. The detection step is then performed by comparing the predictions made by the LSTM to real values assuming that unexpected values in our trading behavioral features predictions reflect potential rogue trading activity. In order to detect anomalies, we define a prediction error that is used to compute an anomaly score based on the Mahalanobis distance.

1 INTRODUCTION

Our work described in this paper intends to detect rogue trading activities using a dataset consisting of transactions booked by operators of a specific desk that deals interest rates and foreign exchange products. This dataset keeps records of all the events happening to a transaction (such as creation, amendment and cancellation) as well as several features describing each of these events (for example, the quantity of a financial product exchanged, its nominal value when issued or the counterparty name).

Currently, the classic process for detecting rogue trading is based on the application of deterministic rules. For example, an alert is generated when the amount of a transaction exceeds a given threshold or when a transaction is recorded with a fictitious counterparty. For each trading activity we find several controls that are mostly based on the fraud schemes detected in the past. This approach presents methodological and practical limitations. Indeed, since it is based on the analysis of past fraudulent behavior, it is hard with this approach to detect or predict new fraud schemes that were not previously listed. In addition, the increasing sophistication of fraudulent behavior, the difficulty in identifying fraud and the constraints associated with massive data processing generate operational and practical complications which materialize through a large number of false positive alerts.This means that a considerable proportion of the alerts issued by the monitoring systems do not actually reveal any suspicious activity. Establishing effective systems to supervise traders is therefore a requirement that all financial institutions must comply with. In addition to meet the obligations imposed by regulators, major banks are seeking to go further by creating new tools to detect abnormal trading behavior.

In this paper, we detect deviations in traders' behavior using a Long Short-Term Memory (LSTM) model. Our anomaly detection method consists of two parts. The first part models traders' behavior by choosing time dependent variables that describe the traders' daily activity and by building a LSTM prediction model to learn the normal patterns of these time series in order to predict their future values. In the second part of this paper, we propose an anomaly score to detect deviations in traders' behavior based on the prediction errors of the LSTM model. Our methodology relies on the hypothesis that our prediction model is trained on normal data (as opposed to fraudulent data) meaning that the training set should not contain records of suspicious transactions related to rogue trading activities. The aim is to learn the time series patterns that characterize a normal traders' behavior in such a way that higher prediction errors identify deviations in traders behavior revealing that

Neyret, M., Ouaggag, J. and Allain, C.
Trading Desk Behavior Modeling via LSTM for Rogue Trading Fraud Detection.
DOI: 10.5220/0009680701430150
In *Proceedings of the 9th International Conference on Data Science, Technology and Applications (DATA 2020)*, pages 143-150
ISBN: 978-989-758-440-4

they are taking unauthorized positions, synonymous of rogue trading. Our work is a first step in the field of deep learning based behavioral analysis for rogue trading fraud detection.

2 RELATED WORK

Anomaly detection in time series data has been widely studied in statistics and machine learning (Chandola et al., 2009) and has many applications in various domains such as fraud detection and cyber security. In the literature we can find problems of semi-supervised anomaly detection for time series data that consist in detecting abnormal points in a test time series with respect to a normal time series. A large number of semi-supervised anomaly detection techniques for time series data have been proposed, such as kernel based techniques (Yankov et al., 2008) and segmentation techniques (Chan and Mahoney, 2005) (Salvador et al., 2004).

The LSTM models introduced by Hochreiter and Schmidhuber in 1997 (Hochreiter and Schmidhuber, 1997) have become the reference model to deal with temporal sequences using recurrent neural networks. Indeed, by avoiding the problem of the vanishing gradients and by introducing a system of memory gates in their architecture, LSTM models make the relevant information contained along the temporal sequences accessible. Moreover, LSTM models have been widely used for sequence learning tasks other than time series such as speech recognition or handwritting recognition.

In some recent published works (Thi et al., 2018) (Malhotra et al., 2015) (Marchi et al., 2015), LSTM networks have been used for anomaly detection in time series data using a pretty similar methodology. LSTM networks are used to model the time series normal patterns by training them on an anomaly-free dataset and to predict their future values, then the prediction errors are used to identify abnormal points. The probability distribution of the errors made while predicting on normal data is then used to obtain the likelihood of normal behavior on the test data. The resulting prediction errors are modeled according to a given distribution (usually a multivariate Gaussian distribution as in (Malhotra et al., 2015)), which is used to assess the likelihood of abnormal points. This methodology has been tested on multiple domains and its efficiency has been demonstrated on various datasets such as ECG data or multi-sensor engine data.

Several LSTM network based applications for the financial industry have been proposed. For instance, LSTM networks have been used to perform stock market predictions both with intraday data (Borovkova and Tsiamas, 2019) and day-over-day data (Lanbouri and Achchab, 2019). Furthermore, some studies have demonstrated the potential of using LSTM networks in the field of trading investment decisions. Market behavior and logic can be learned in order to develop new optimized trading strategies (Troiano et al., 2018), (Sang and Pierro, 2019).

To the best of our knowledge, our paper is the first one to propose a methodology to use LSTM networks to model human behavioral features for deviation detection in the financial industry. Our results thus can not be confronted to similar approaches or benchmark sudies at this point. Our modeling relies on real transactional data generated by the traders and on the definition of accurate time dependant variables that fit best their daily activity. Since traders' behavior deviations can be caused by other things than rogue trading, such as operational errors and unusual or poor trading activity, our approach can point out behavior anomalies both in its global and weak signals based characteristics. Therefore, thanks to its methodology and to the data used, our approach can be seen as a real-world traders' behavior anomaly detection method.

3 METHODOLOGY

A trading desk activity may be represented by time series describing the traders' activities. Our approach is to use LSTM networks to predict trading behavioral features in order to detect unexpected patterns that are possible frauds. These are highlighted by comparing the predictions made by the LSTM to real values. In this part, we present in detail and chronologically our two-step methodology developed for fraud detection.

3.1 Background on Recurrent Neural Networks (RNN)

RNNs tackle the time series modeling problem in the neural networks area since their popularization by Elman in 1990 (Elman, 1990). This type of neural network is able to capture short time dependencies in sequences of data as it takes into account both new time steps and information of previous states. This means that, at time t, a RNN computes its output depending on the current data sample x_t (corresponding to l time steps of the time series) and what it knows from previous time steps h_{t-1}.

The Vanilla RNN is driven by the following equations:

$$\begin{aligned} h_t &= \sigma_h(W\ x_t + U\ h_{t-1} + b_h) \\ y_t &= \sigma_y(V\ h_t + b_y) \end{aligned} \tag{1}$$

where:

- $x_t \in \mathbb{R}^l$, $h_t \in \mathbb{R}^k$ and $y_t \in \mathbb{R}^p$ are denoted as input, hidden state (memory of the model) and output;
- $W \in \mathbb{R}^{k\times l}$, $U \in \mathbb{R}^{k\times k}$ et $V \in \mathbb{R}^{p\times k}$ are weights matrices;
- $b_h \in \mathbb{R}^k$ et $b_y \in \mathbb{R}^p$ are bias vectors;
- σ_h et σ_y are common activation functions.

This shows that a single RNN is working on fixed-size sequences that are processed recursively in order to complete a constant task as its weights and bias do not evolve according to the processed time step.

In order to update weight matrices and bias vectors, the most common approach is to use a gradient descent method, generalized to work on time dependent data: the back-propagation through time (BPTT). This concept is explained in detail in (Goodfellow et al., 2016) and a survey on different optimizers based on gradient descent is given in (Salehinejad et al., 2018).

3.2 Forecasting Behavior using Long Short-term Memory

As mentioned in (Hochreiter and Schmidhuber, 1997), (Salehinejad et al., 2018) and (Shertinsky, 2018), the issue of exploding and vanishing gradients is encountered while training a RNN with BPTT on long sequences. This means that the RNN cannot learn long-term dependencies.

To overcome this issue, LSTM networks were introduced by S. Hochreiter and J. Schmidhuber in 1997 (Hochreiter and Schmidhuber, 1997). They are a type of recurrent network designed to work on different time scales, on both long and short term dependencies. The repeated unit is a combination of four elements:

- a memory cell, $C_t \in \mathbb{R}^k$, responsible for information storage and updated at each time step;
- a forget gate, $f_t \in [0;1]^k$, that defines what past information stored in the memory cell will be forgotten;
- an input gate, $i_t \in [0;1]^k$, that controls what new information to add to the memory cell;
- an output gate, $o_t \in [0;1]^k$, that drives the impact of the memory cell on the hidden state $h_t \in [-1;1]^k$.

Each element has its own parameters (weight matrices and bias vectors) but they are all applied on the same inputs: the current data sample $x_t \in \mathbb{R}^l$ and what it knows from previous time steps $h_{t-1} \in \mathbb{R}^k$.

The LSTM follows the equations below, using $\circ$ the element-wise product:

$$\begin{aligned} f_t &= \sigma(W_f\ x_t + U_f\ h_{t-1} + b_f) \\ i_t &= \sigma(W_i\ x_t + U_i\ h_{t-1} + b_i) \\ C_t &= C_{t-1} \circ f_t + \tanh(W_C\ x_t + U_C\ h_{t-1} + b_C) \circ i_t \\ o_t &= \sigma(W_o\ x_t + U_o\ h_{t-1} + b_o) \\ h_t &= \tanh(C_t) \circ o_t \end{aligned} \tag{2}$$

This model is able to predict future time steps of time series and we will use it to forecast some variables that represent the behavior of a trading desk. These will be introduced later in the paper.

3.3 Detection of Unexpected Behavior

In this section, we present the chosen methodology to discover unexpected patterns in our predicted behavioral variables. From now on, the following notations will be used:

- $t \in [0;n]$ is a fixed time step;
- $x^{(t)} = \left(x_1^{(t)}, x_2^{(t)}, \ldots, x_m^{(t)}\right)^T \in \mathbb{R}^m$ corresponds to input data at time t;
- $y_{pred}^{(t)} = \left(y_{pred,1}^{(t)}, y_{pred,2}^{(t)}, \ldots, y_{pred,d}^{(t)}\right)^T \in \mathbb{R}^d$ is the LSTM based prediction at time t (computed with information available at $t-1$);
- $Y_{pred} = \left(y_{pred}^{(1)}, y_{pred}^{(2)}, \ldots, y_{pred}^{(n)}\right) \in \mathbb{R}^{d\times n}$ is the vector that gathers all the carried out predictions;
- $y_{true}^{(t)} = \left(y_{true,1}^{(t)}, y_{true,2}^{(t)}, \ldots, y_{true,d}^{(t)}\right)^T \in \mathbb{R}^d$ are the observed values of outputs at time t;
- $Y_{true} = \left(y_{true}^{(1)}, y_{true}^{(2)}, \ldots, y_{true}^{(n)}\right) \in \mathbb{R}^{d\times n}$ is the vector that gathers all the observed values.

These notations show that we use the LSTM to predict a d-dimensional output vector from m input variables. Some features may be both in the input and output vectors. Regarding the temporal aspect, l time steps are given to the LSTM for the m input features in order to compute the d predictions on the next time step. At the end, the goal is to identify the set of times t such that $y_{true}^{(t)}$ deviates from what we predicted.

First, we start by comparing, at each time step $t \in [0;n]$, the prediction $y_{pred}^{(t)}$ to the observed value $y_{true}^{(t)}$ in order to compute the prediction error that is defined for all $t \in [0;n]$ by:

$$e^{(t)} = \left\| y_{true}^{(t)} - y_{pred}^{(t)} \right\|_2^2 = \begin{pmatrix} e_1^{(t)} \\ \vdots \\ e_d^{(t)} \end{pmatrix} \in \mathbb{R}_+^d, \tag{3}$$

where,

$$e_i^{(t)} = \left(y_{true,i}^{(t)} - y_{pred,i}^{(t)}\right)^2, \; i = 1,...,d.$$

Then, this temporal prediction error is used to define an anomaly score for all $t \in [0;n]$, $a^{(t)}$, based on the Mahalanobis distance. Introduced by P.C. Mahalanobis in 1936 (Mahalanobis, 1936), this distance has the advantage of taking correlations into account. Here, we are interested in defining the Mahalanobis distance between one point P and a set of points. To that end, we use the Mahalanobis distance between P and the mean of the set of points.

The Mahalanobis distance between a p-dimensional vector $x = (x_1, x_2, ..., x_p)^T$ and a set of points with mean $\mu = (\mu_1, \mu_2, ..., \mu_p)^T \in \mathbb{R}^p$ and variance-covariance matrix $\Sigma \in \mathbb{R}^{p \times p}$ is:

$$D_M(x, \mu, \Sigma) = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)}. \tag{4}$$

For all $t \in [w+1;n]$, the idea here is to compute the Mahalanobis distance of $e^{(t)}$ from the set of w prediction errors corresponding to a time window preceding t. For the rest of this section, let $w \in \mathbb{N}$ and $t \in [w+1;n]$ be fixed. We define the prediction errors on w as:

$$E_w^{(t)} = \left(e^{(t-w)}, e^{(t-w+1)}, ..., e^{(t-1)}\right) = \begin{pmatrix} E_{w,1}^{(t)} \\ \vdots \\ E_{w,d}^{(t)} \end{pmatrix}, \tag{5}$$

where,

$$E_{w,i}^{(t)} = \left(e_i^{(t-w)}, e_i^{(t-w+1)}, ..., e_i^{(t-1)}\right), \; i = 1,...,d.$$

We deduce the empirical mean and the covariance from this set as:

$$\widehat{\mu}_w^{(t)} = \frac{1}{w} \sum_{i=1}^{w} e^{(t-i)} \tag{6}$$

$$\widehat{\Sigma}_w^{(t)} = \frac{1}{w-1} \sum_{i=1}^{w} \left(e^{(t-i)} - \widehat{\mu}_w^{(t)}\right) \left(e^{(t-i)} - \widehat{\mu}_w^{(t)}\right)^T \tag{7}$$

$$\tag{8}$$

and the anomaly score of $x^{(t)}, t \in [w+1;n]$, is computed as:

$$\begin{aligned} a^{(t)} &= D_M\left(e^{(t)}, \widehat{\mu}_w^{(t)}, \widehat{\Sigma}_w^{(t)}\right) \\ &= \sqrt{\left(e^{(t)} - \widehat{\mu}_w^{(t)}\right)^T \left(\widehat{\Sigma}_w^{(t)}\right)^{-1} \left(e^{(t)} - \widehat{\mu}_w^{(t)}\right)} \in \mathbb{R}_+. \end{aligned} \tag{9}$$

As a history of w prediction errors is needed to compute an anomaly score, we only get $n-w$ anomaly scores. We denote by $A = \left(a^{(w+1)}, a^{(w+2)}, ..., a^{(n)}\right) \in \mathbb{R}_+^{(n-w)}$ the vector of anomaly scores.

Finaly, the last step is to get a rule to assess the abnormality of a time step t. As seen in the litterature (Cabana et al., 2019) (Filzmoser, 2004) (Malhotra et al., 2015), we assume that the prediction error vectors follow a multivariate Gaussian distribution. Given the definition of the anomaly score and the normality assumption, the first idea is to use the following property between Gaussian and Chi-squared distributions: if $X \in \mathbb{R}^p$ is a multivariate random variable following a Gaussian law of parameters μ and Σ, $\mathcal{N}_p(\mu, \Sigma)$, then $D_M(x, \mu, \Sigma)^2$ is well approximated by a Chi-squared distribution with p degrees of freedom, $\chi(p)$. This means that, for all $t \in [w+1;n], D_M\left(e^{(t)}, \widehat{\mu}_w^{(t)}, \widehat{\Sigma}_w^{(t)}\right) \sim \chi(d)$ which allows us to define a threshold thanks to the $(1-\alpha)$-quantile of the previous law as:

$$\tau_{chi} = \chi_{1-\alpha}(d), \alpha \in [0;1]. \tag{10}$$

The second idea is to get a fixed proportion of anomalies. We sort in ascending order the vector A of all anomaly scores such as $\overline{A} = (a_0, a_1, ..., a_{n-w-1}) \in \mathbb{R}_+^{n-w}$. To highlight the top $\alpha \in [0;1]$ of highest anomaly scores, we fix:

$$\tau = a_{\lfloor \bar{n} \rfloor} + \left(a_{\lceil \bar{n} \rceil} - a_{\lfloor \bar{n} \rfloor}\right) * \{\bar{n}\} \tag{11}$$

where $\bar{n} = (n-w-1)(1-\alpha)$ and $\{x\}$ denotes the fractional part of $x \in \mathbb{R}$.

In order to take into account the most recent context, we add temporality by computing one threshold at each time step t based on the anomaly scores of the preceding window of length $w' \leq w$. This means that the computation of the threshold is restricted on the vector $A_{w'}^{(t)} = \left(a^{(t-w')}, a^{(t-w'+1)}, ..., a^{(t-1)}\right) \in \mathbb{R}_+^{w'}$, or $\overline{A_{w'}^{(t)}} = \left(a_{t,0}, a_{t,1}, ..., a_{t,w'-1}\right)$ for its sorted version. For all $t \in [w+1+w';n]$, the time-dependent threshold is:

$$\tau_{prop}^{(t)} = a_{t,\lfloor \bar{n}_{w'} \rfloor} + \left(a_{t,\lceil \bar{n}_{w'} \rceil} - a_{t,\lfloor \bar{n}_{w'} \rfloor}\right) * \{\bar{n}_{w'}\}, \tag{12}$$

where $\bar{n}_{w'} = (w'-1)(1-\alpha)$.

We chose to use a combination of both thresholds to get a dynamic one that can detect more complex anomalies. Therefore, the final rule of detection is to consider $y_{true}^{(t)}$ as abnormal if:

$$\begin{aligned} a^{(t)} > \tau_{chi} \;\; &\text{when } t \in [w+1; w+1+w'], \\ a^{(t)} > \min\left(\tau_{chi}, \tau_{prop}^{(t)}\right) \;\; &\text{when } t > w+1+w'. \end{aligned} \tag{13}$$

4 EXPERIMENT

The purpose of this section is to test the proposed methodology. Our main focus will be to evaluate our ability to predict trading behavioral features. To provide a comprehensive analysis of the detector performances, a complete and exhaustive supervised dataset or discussions with business experts would be required to challenge detection of abnormal patterns. We point out that, at this point of the project, mainly transactional data is used to create the trading behavioral features. One transaction is the result of the buying or selling of a financial product. In the following, a deal and a transaction refer to the same idea.

4.1 Transactional Data

Our work is focused on a desk dealing interest rates and foreign exchange products, mainly on emerging currencies from Latin America. The studied desk is working worldwide on different types of products related to foreign exchange rates and currencies such as spot, forward, futures... The pool of traders composing a trading desk may evolve over a considered period. A trading desk can experience some turnover as traders change position or take holidays.

Four and a half years of transactions made by this trading desk have been gathered. In this dataset, one transaction corresponds to several rows if the deal has been modified or cancelled after its creation, as these actions generate new versions of the transaction. A few of the many characteristics of the transactions are detailed in Table 1. We assume with enough certainty that the data available corresponds to normal activity, excluding any rogue trading records. This is key for the training task as it is a fundamental hypothesis for our proposed methodology. Moreover, this characteristic does not undermine the prediction performances as the underlying data is not biased by fraud.

In the first place, the key point is to define features from this transactional database that represent a trading behavior as time series. We choose to aggregate the information by hour in order to have enough depth in the data without being at a scale hiding all the signal. This means that a tradeoff was made between having enough depth in our time series data to train the model and maintaining a sufficient level of detail when aggregating. The behavioral features have evolved a lot throughout the project. For the results in 4.3, we keep the set of behavioral features in Table 2.

Besides, we decide to add some simple market data to contextualize our trading behavioral features. Indeed, the daily Volatility Index [1] (VIX) and Treasury-EuroDollar rate spread [2] (TED Spread) are given as input data to the prediction model without being features to predict.

Finally, we get 5 time series with a one-hour time step: 3 of them are trading behavioral features and the 2 others are contextual ones. The goal is to predict the behavioral features from their past and contextual ones. Therefore, our LSTM input is a vector of 5 time series: $x^{(t)} \in \mathbb{R}^m$ with $m = 5$ and our LSTM output is a vector of 3 time series: $y^{(t)}_{pred} \in \mathbb{R}^d$ with $d = 3$.

4.2 Experimental Settings

The first step of the methodology is to use LSTMs to carry-out a single-step ahead prediction task knowing the l previous hours, using the features framework presented in 4.1. All the time series, except the proportion ones, are scaled with a min max scaler between 0 and 1, before being shaped into a 3 dimension dataset (observations, features, time).

The model is implemented with Keras. Our model is optimized and trained with four years of transactional data, keeping the last six month seperate for testing purposes; in terms of set sizes, it represents 35064 observations for training and 4440 for testing. The last 20 % of the train set are used as a validation set. We choose the MAE to optimize and train the prediction model as the detector already penalizes large errors.

To choose the different parameters of the model, we generate sets of parameters based on the possible values of each parameters. A randomized search is then performed to find the best suiting parameters: a random subsample of the sets of parameters is picked, for each of these sets of parameters an LSTM is trained on training data (validation set excluded) and the performance of the trained LSTM is evaluated on the validation data. The best set of parameters is chosen as the one that minimizes the MAE on the validation set. The optimization was performed with a maximum training duration of 100 epochs, minibatches of size 32 and an early stopping patience of 10. We choose to tune the following parameters: number of LSTM layers, number of neurons and dropout values for each LSTM layer, type of optimizer and learning rate. Other experiments were conducted regarding input features, the length of input series and output features.

The best performances in terms of MAE on validation set were reached using an Adam optimizer with a 0.001 learning rate and input series of 48 time steps. The results in the next section were obtained using a neural network with the following structure:

[1] https://www.investopedia.com/terms/v/vix.asp

[2] https://www.investopedia.com/terms/t/tedspread.asp

Table 1: Characteristics of one transaction.

Characteristic	Description
Nominal	Value given to the financial product when issued
Quantity	Number of financial product units that are bought or sold
Way	Specifies if the financial product is bought or sold
Product	Type of the financial product exchanged
Currency 1 and 2	Foreign currencies involved in the transaction
Version number	Number that increments by 1 at each new action done on the transaction
Version date and hour	Date and hour on which an action is done on the transaction
Version type	Specifies if the action corresponds to the creation, to a modification or to the cancellation of the transaction
Counterparty name	Full name of the entity on the other side of the transaction
Counterparty type	Specifies if the counterparty is inside the company network

Table 2: Designed behavioral features.

Behavioral feature name	Description
Prop_versiontype_M	Proportion of deals that have been modified at least once during the hour
Mean_versiontype_M	For the modified deals, the average number of modifications in the hour
Prop_versiontype_D	Proportion of deals that have been canceled during the hour

- an input layer with 5 features and 48 time steps;
- two LSTM layers with respectively 64 and 32 hidden neurons, a dropout value of 0.2 on both layers and a hyperbolic tangent activation function;
- a dense layer as output layer with 3 neurons and a linear activation function.

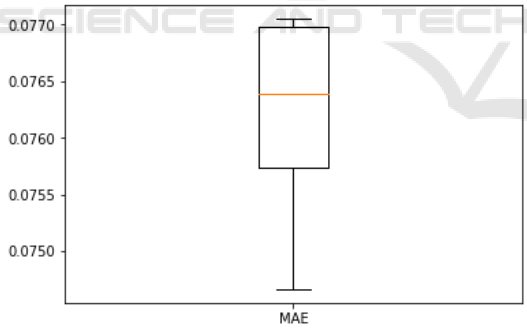

Figure 1: Assessment of LSTM stability.

To ensure model stability, this LSTM topology is then trained 15 times and we report the MAE computed on the validation set for each training in Figure 1. It shows an average performance of 0.0762 and a standard deviation of 0.00078. Considering that the standard deviation represents 1 % of the average performance, we can assume that this LSTM topology is stable.

4.3 Results

4.3.1 Behavioral Features Prediction

In Table 3, to demonstrate the reliability of the proposed methodology, we compare MAE performances on the scaled test set of our trained LSTM with:

- two naives models that copy either previous day values or previous week values as these lags correspond to the strongest seasonalities,
- one SARIMA (Seasonal AutoRegressive Integrated Moving Average), a classic time series model that appears to better fit our data than other tested models such as AR, MA or ARIMA because of the seasonality component of the time series. The parameters are optimized independently for each time series and no other feature is given to the model as context. This means that, to predict one feature, the SARIMA only uses its past. Considering our time series, we chose to set the seasonality to 24 hours as it is the smallest seasonality appearing in the analysis. The second seasonality, being one week, is a multiple of the chosen seasonality.

The proposed methodology is trained 15 times to get a performance interval defined by a mean MAE and standard deviation.

We observe that the MAE of one week naive model is better than the one of one day naive model. This is mainly caused by the existence of a strong weekly seasonality in our data as there is less activity

Table 3: Models performances.

Model	MAE
Naive D-1	0.1020
Naive D-7	0.0832
SARIMA	0.0926
Proposed LSTM	**0.0713 ± 0.0009**

on week-end days which breaks the daily seasonality.

Besides, we see that the optimized SARIMA with a 24 hour seasonality improves the performance of the one day naive model. This suggests that it captures the signal better. However, it does not reach the same performance as the LSTM. This may be due to three main reasons:

- each feature is predicted separately from the others,
- the contextual information is not included,
- the weekly seasonality is not enough taken into account.

Our proposed prediction model seems to have better results on trading behavioral features than more classic models. It seems to be a good starting point on which we can apply a deviation detector in order to get some abnormal patterns.

Regarding running time, predictions with the one week naive model are immediate while the proposed LSTM requires a certain optimization and training time before making predictions. Improved results justify the near half hour LSTM training and prediction time. However, it must be kept in mind that the optimization process is quite long as many sets of parameters have to be tested.

4.3.2 Detection of Behavior Deviation

For our study, we chose a desk on which we had confirmed presence of abnormal behavior over a period close to the four and a half years of data used in the previous section. We highlight the fact that the number of abnormal transactions is marginal regarding the desk activity. This should thus be taken into account for the interpretation of the following results: our model objective being also to be able to seek very weak signals (in number or volume of transactions, which are negligible compared to the activity).

Over a two-month period, not included in the data used to train and test the LSTM in 4.3.1, there was a total of 42 established abnormal transactions on different hours. Using the three behavioral features defined in 4.1, an α of 0.05 for the chi threshold and an α of 0.1 for the percentile one, the methodology emphasizes 59 abnormal hours out of more than 1400. These hours include 5 of the 42 abnormal transactions to be found. Even if the rate of identification of true anomalies is low, the analysis of these methodology results would have probably raised alerts among analysts on this abnormal pattern if such a model was running at the time of these abnormal deals. Moreover, business experts have not yet analyzed whether the false alerts were actually accurate.

Before this set of behavioral features, we tested many other sets, including one with 3 different behavioral features to predict: sum of nominals, number of transactions and number of modified or cancelled transactions. In this set up, the prediction abilities of the model were worse than the ones shown in 4.3.1 but the we found 16 of the 42 abnormal deals among 73 abnormal hours emphasized by the detector. Discussions with experts showed that:

- the feature sum of nominals has no business meaning because the definition of nominal depends on the product,
- using mixed types of features (nominal vs count of transactions) might be tricky for the prediction step.

This is the reason why we worked again on behavioral features definition ending up with the three behavioral features defined in 4.1.

The difference between the previous detection results proves that the key to detect real anomalies is to define features that are consistent and global enough to detect new abnormal patterns. The three features used here are a starting point but they need to integrate other aspects of transactions characteristics. For example, the same features may be computed on data subset depending on the type of counterparty and the type of product. This will also let us use the nominal feature as it will make more sense to compare them within a type of product. Moreover, using a larger panel of prediction outputs could lead to a more accurate deviation detector to identify abnormal patterns.

5 CONCLUSION

In this paper, we propose a whole methodology in order to detect deviations in trading behavior. A LSTM based model is used to predict simple behavioral features linked to the type of transactions and, by comparing these predictions to their real values, a detector is defined using the Mahalanobis distance. The goal was to demonstrate the potential of such an approach using behavioral features for fraud detection in a financial use case. We do not claim to achieve state of the art performances in the prediction task: further fine tuning and more sophisticated models could lead to improved results. However, we prove that using a LSTM network to predict trading behavioral features makes sense as it improves prediction performances. Detector performances are not really outstanding for the moment but many prospects have already been raised in the team to improve detecting performances.

First, as emphasized in the last part of the paper, the definition of behavioral features is key. Focusing on transactional data, as we already said, chosen features must involve more aspects of transactions structure. Furthermore, contextual data could be enhanced with more market data, information about traders' communications or data linked to economic news announcements.

Then, setting up the first prospect above will multiply the number of series both for the input and output vectors. For this reason, a more sophisticated prediction model would probably be used such as DeepAR (Salinas et al., 2017) or attention-based LSTM network (Qin et al., 2017).

Finally, an optimization of the parameters of the detector would be necessary before any production phase of the methodology. For that, either experts are ready to invest time to help us analyze the detected transactions for different parameters of the detector in an iterative process or we design a generator of abnormal transactions that have to be identified in order to perform calibration. This parametrization could imply the use of a more robust distance to define the anomaly score as seen in (Cabana et al., 2019).

REFERENCES

Borovkova, S. and Tsiamas, I. (2019). An ensemble of LSTM neural networks for high-frequency stock market classification. *Journal of Forecasting*.

Cabana, E., Lillo, R., and Laniado, H. (2019). Multivariate outlier detection based on a robust Mahalanobis distance with shrinkage estimators. *Stat Papers*.

Chan, P. and Mahoney, M. (2005). Modeling multiple time series for anomaly detection. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*.

Chandola, V., Cheboli, D., and Kumar, V. (2009). Detecting anomalies in a time series database. *Computer Science Department TR 09-004, University of Minnesota*.

Elman, J. (1990). Finding structure in time. *Cognitive science*.

Filzmoser, P. (2004). A multivariate outlier detection method.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation 9*.

Lanbouri, Z. and Achchab, S. (2019). A new approach for trading based on Long-Short Term memory technique. *International Journal of Computer Science Issues*.

Mahalanobis, P. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*.

Malhotra, P., Vig, L., Shroff, G., and Agarwal, P. (2015). Long Short Term Memory networks for anomaly detection in time series. In *ESANN*.

Marchi, E., Vesperini, F., Weninger, F., Eyben, F., Squartini, S., and Schuller, B. (2015). Non-linear prediction with LSTM recurrent neural networks for acoustic novelty detection. In *2015 International Joint Conference on Neural Networks (IJCNN)*.

Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., and Cottrell, G. (2017). A dual-stage attention-based recurrent neural network for time series prediction. *CoRR*.

Salehinejad, H., Sankar, S., Barfett, J., Colak, E., and Valaee, S. (2018). Recent advances in Recurrent Neural Network (RNN). *ArXiv*, 1801.01078.

Salinas, D., Flunkert, V., and Gasthaus, J. (2017). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *ArXiv*, 1704.04110.

Salvador, C., Chan, P., and Brodie, J. (2004). Learning states and rules for detecting anomalies in time series. In *FLAIRS Conference*.

Sang, C. and Pierro, M. D. (2019). Improving trading technical analysis with TensorFlow Long Short-Term Memory (LSTM) neural network. *The Journal of Finance and Data Science*.

Shertinsky, A. (2018). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *ArXiv*, 1808.03314.

Thi, N., Cao, V., and Le-Khac, N. (2018). One-class collective anomaly detection based on Long Short-Term Memory recurrent neural networks. *ArXiv*, 1802.00324.

Troiano, L., Villa, E. M., and Loia, V. (2018). Replicating a trading strategy by means of LSTM for financial industry applications. *IEEE Transactions on Industrial Informatics*, 14.

Yankov, D., Keogh, E., and Rebbapragada, U. (2008). Disk aware discord discovery: Finding unusual time series in terabyte sized datasets. *Knowl Inf Syst*, 17.