Next Event Prediction in Business Process Logs Using Stacked Autoencoders with N-gram Encoding and Feature Hashing

Rongming Li
Computer Science and Technology, Zhoukou Vocational College of Arts and Sciences, Zhoukou, Henan, China
E-mail: leerongming@163.com

Keywords: business process prediction, stacked autoencoders, deep learning, hyperparameter optimization, process event prediction, N-gram encoding

Received: March 25, 2025

Proactive monitoring of business processes has become a key competitive advantage for firms, enabling timely interventions to prevent workflow deviations. Process-aware information systems generate extensive logs, which serve as valuable resources for predictive analytics. In this context, this study presents a deep learning-based approach for predicting the next event in an ongoing process by analyzing historical execution logs. The proposed method formulates event prediction as a classification task, leveraging n-gram encoding and feature hashing for effective feature preprocessing. The model consists of a multi-stage deep learning framework, incorporating stacked autoencoders for unsupervised pre-training, followed by a supervised fine-tuning phase to optimize classification accuracy. Experimental validation was conducted on six real-life event log datasets, including BPI Challenge 2012 (subsets A, O, W), BPI Challenge 2013 (Incidents, Problems), and Helpdesk logs. The proposed approach achieved up to 83.1% accuracy, 85.2% precision, and 92.3% AUC on the BPI 2012_A dataset, outperforming state-of-the-art classifiers such as LSTM-based models and Bayesian regularized PFAs. Notably, it demonstrated a 6–11% improvement in recall over existing methods on key datasets. The results highlight the model's ability to capture complex process dynamics and improve proactive situation awareness. Additionally, the study explores the impact of hyperparameter tuning and addresses data imbalance challenges using RBF-based synthetic data generation, contributing a robust framework for real-time decision support in business process management.

Povzetek: V prispevku so opisani zloženi samokodirniki z n-gram/hashing kodiranjem za napoved naslednjega dogodka, ki presegajo LSTM/PFA in z RBF uravnoteževanjem izboljšajo redke razrede.

1 Introduction

In practice, high-performance business processes remain as one of a handful of remaining sources of differentiation. Inclusion of predictive analytics into enterprise processes can bring a tremendous increment in business values. A process-aware Enterprise Information Systems (EIS) like workflow management systems (WMS), enterprise resource planning (ERP), customer relationship management (CRM), incident management (IM) generates log events when processes are being executed [1]. These logs are a rich source for predictive analytics, helping to make informed decisions due to their ability to predict future process behaviors. Well-designed and properly implemented predictive approaches enable business activities to proceed according to plans without predicted failures or deviations from the foreseen behavior of the processes. The data-driven predictive process analytics enable use cases such as real-time anomaly detection in processes, analysis of customer behavioral patterns for offering them individualized offers, risk management by prediction of compliance violations, and efficient resource allocation [2].

The current focus of EIS is to enhance the ability of an organization in performing high-performing business processes. However, most of the systems are not effective enough because advanced predictive analytics is missing. Most of the integrated business intelligence tools work on descriptive analytics that can only support addressing demographic trends and performance-related issues. It is also not adequate for a business just to optimize its operational efficiency in order to remain competitive. According to Duan and Da Xu, the struggle by companies is in converting the gigantic amounts of data generated into useful insights that will help them in providing quality products and services [3]. For that reason, in the future, EIS have to go beyond diagnostic examination of historical data by providing proactive decisions based on predictive analytics. The process needs predictive capabilities embedded in the business processes. EIS form the basis in their role as process orchestration tools. A significant and developing related trend of IS research involves integrating advanced analytics with EIS [4].

Business process prediction involves the forecast of a target variable by extracting meaningful features from business process log data analysis. When the target variable is continuous, for instance, estimated time

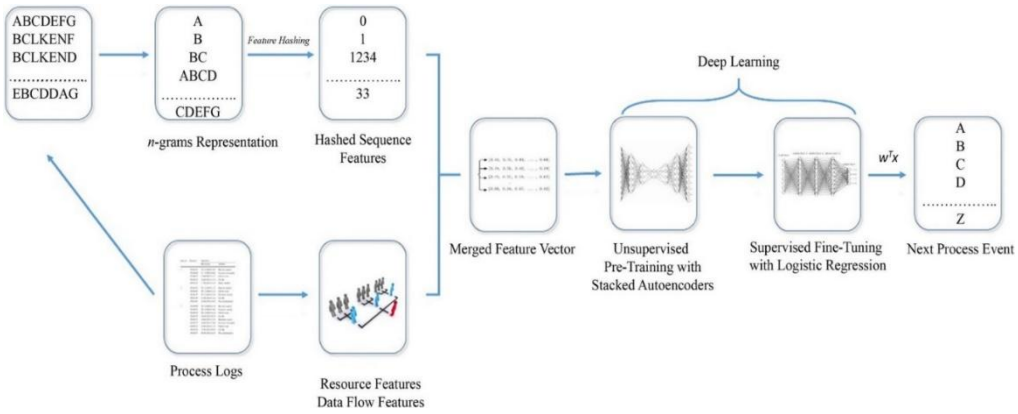

Figure 1: The stages of the proposed approach.

remaining for the completion of a process, then the task falls under the category of regression analysis. Otherwise, if the events involve discrete values, such as predicting the next activity in an ongoing process, predicting the outcome of a process instance, or the likelihood of violating a service level agreement, the problem is dealt with as classification. This study focuses on predicting future events in a business process based on the sequence of previous events extracted from the given instance using

Table 1: Summary of prior work on process prediction

Study	Method	Dataset	Metrics	Accuracy (%)
[2]	LSTM with embeddings	BPI 2012, 2013	Precision	~76–79
[12]	Bayesian PFA	BPI 2012, 2013	Accuracy, AUC	~71–76
[14]	LSTM with timestamps	BPI 2012, Helpdesk	Accuracy	~71–73
[13]	Evo. Rule-based	BPI 2013	F-score	~69–72
[10]	HMM + k-NN	Simulated logs	Accuracy	~70
Proposed Approach	Stacked Autoencoders + N-gram Hashing	BPI 2012, 2013, Helpdesk	Acc, Prec, AUC	Up to 83.1

execution log data from already finished instances. This problem is of high importance in process analytics since these predictive results enable proactive actions to minimize undesired outcomes. As an answer to this challenge, we propose a multi-stage deep learning framework.

This paper is the first to apply a deep learning framework in business process management, which has been characterized by both an unsupervised pre-training phase using stacked autoencoders and a supervised fine-tuning stage for multi-class classification tasks. The approach greatly enhances the state-of-the-art techniques on process prediction by initializing neural network parameters through employing greedy layer-wise pre-training and optimizing a global training objective with labeled data. In this work, a solid data pre-processing pipeline is also introduced, which, for the first time in this domain, transforms event log data into numerical feature vectors using n-gram representation and feature hashing. Such encoding considers the sequential nature of process data and, at the same time, reduces the dimensionality, enhancing the inference speed of deep neural networks. Furthermore, the research addresses some of the key challenges such as hyperparameter optimization and handling data imbalance to improve the accuracy and precision of prediction. Based on the "exaptation" design science research approach by Gregor [5], this study adopts proven techniques such as stacked autoencoders and feature hashing to develop novel predictive analytics models for process data in EIS.

To clarify the scope and objectives of this study, we define the outcome variable as the next activity label in a sequence of events from an ongoing business process instance. This transforms the task into a multi-class classification problem, where the input is a prefix of historical events and the target is the label of the next event in the sequence. The research is guided by the following key questions:

- RQ1: Can the proposed deep learning framework, which incorporates stacked autoencoders and n-gram encoding with feature hashing, outperform traditional classification methods on standard business process log datasets?
- RQ2: How does the predictive performance of the proposed approach compare to state-of-the-art deep learning models, such as LSTM-based architectures and Bayesian probabilistic finite automata?
- RQ3: Can data imbalance in business process logs be effectively addressed using synthetic data generation through RBF-based neural network resampling, thereby improving prediction for rare but significant events?

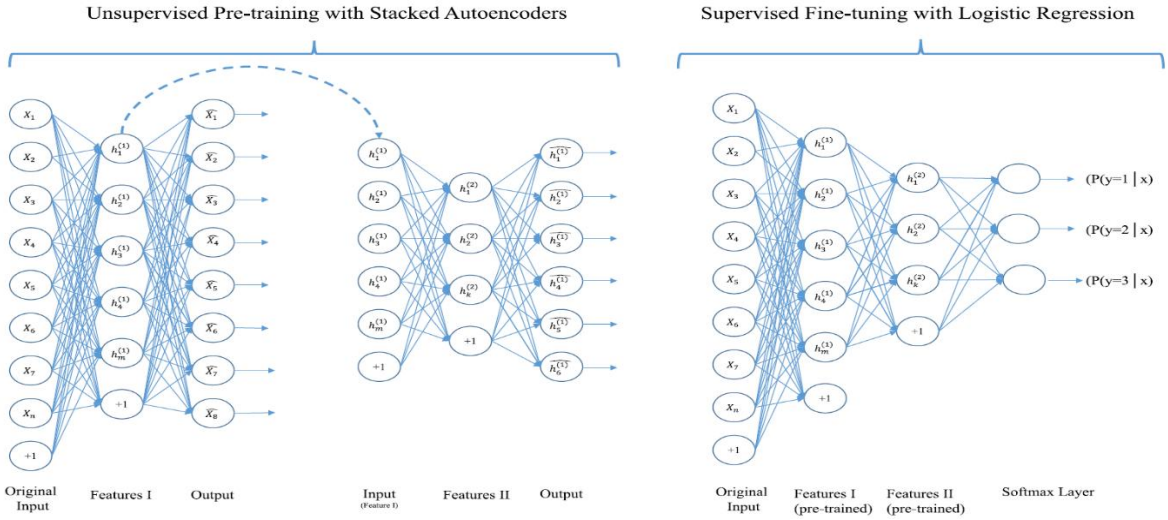

Figure 2: Deep learning with stacked autoencoders: unsupervised pre-training is shown on the left, while supervised fine-tuning is depicted on the right.

These questions directly shape our experimental design and analysis, as presented in the subsequent sections.

2 Related works

An increasing amount of research has focused on machine learning methods in the realm of business process management. We categorize these studies with respect to the type of the target variable they predict-whether it is discrete or continuous-and discuss the problem types falling into these categories.

Approaches belonging to the first category address regression problems; the most common is the prediction of the remaining processing time of incomplete cases. For example, van Dongen et al. [6] estimated the remaining cycle time using event log data based on nonparametric regression techniques. Rogge-Solti [7] have proposed a stochastic Petri net with generally distributed transitions with the aim of foretelling execution time as a function of the time elapsed since the last observed event. Folino et al. [8] introduced a hybrid model in which PCT was combined with performance-annotated FSM to produce time predictions. The approach proposed by Senderovich et al. [9] used linear regression, random forests, and XGBoost with features extracted from process instances and global models for time prediction.

The other group targets classification problems, which are process outcome prediction, service level agreement violations, nominal attribute prediction, and next event prediction. Only a few papers address the next event prediction problem presented here. Le et al. [10] took this a step further with their enhanced Markov model and sequential k-nearest neighbor classification, outperforming traditional Markov and HMM. The other works proposed include the following: Unuvar et al. [11] designed a decision tree model to forecast the next activity in a process that contains parallel execution paths. Breuker et al. [12] proposed a PFA with Bayesian regularization and assessed its performance using both simulated and real datasets. Márquez-Chamorro et al. [13] proposed an evolutionary rule-based approach for next event prediction, featuring window-based feature encoding, which was evaluated over datasets such as the BPI Challenge 2013 and health services logs.

More recent works on process prediction have abandoned explicit process models in favor of deep learning techniques. Among the very first proposals of RNNs with LSTM units, Evermann et al. [2] presented an approach that used word embeddings as a preprocessing step for the input features, adding to the case-specific and event-specific explanatory variables, greatly improving predictive performance. They evaluated their approach using the BPI Challenge 2012 and 2013 datasets. Similarly, Tax et al. [14] also used LSTMs for paying attention to the sequence of the activities and their corresponding timestamps. They represented input activities as a feature vector with one-hot encoding. Both investigated methods for the prediction of process activity durations.

This paper extends prior work in the following necessary areas: optimization of hyperparameter, handling imbalanced datasets, and enhancement of feature encoding. On the contrary to index-based encoding methods widely used in existing works, [9][13][15] we apply an n-gram-based encoding schema that captures interdependencies in sequential event data. We apply feature hashing to reduce the dimensionality of this high-dimensional feature space. Moreover, we perform hyperparameter optimization to ensure that better classification performance is achieved, which has not been taken into consideration yet in business process event prediction. To address the issue of imbalanced datasets-a challenge largely overlooked except in Márquez-Chamorro et al. [13], we balance the data by synthesizing new instances for the minority class using neural networks. This approach allows for better identification of rare events, which in most cases convey significant business implications.

3 Methodology

We formulate the prediction of the next process event as a classification problem. Our approach is outlined in Figure 1. Our approach applies deep learning algorithms on the feature matrix, which is directly extracted from data flow, control flow, resource, and organizational perspectives of the process via a preceding step of data preprocessing. Our approach extracts, in a sliding window fashion, the process events (or control flow) from event log data and encodes these into n-gram feature representations; see Figure 1. Afterwards, these n-grams are mapped to hash keys using feature hashing. Then, the hashed feature matrix is extended with data and resource features. This section elaborates on the proposed deep learning technique by predicting the upcoming process events after developing an extended feature matrix. It contains an unsupervised layer-wise pretraining module to generate higher-order features and a fine-tuning of the whole network in a supervised manner, accompanied by an output layer for multiclass classification at the top.

A) Terminology

An event log consists of process traces, where one trace is the execution of a single process instance or case. A trace is basically a sequence of events, and these events contain attributes describing their properties according to the XES Standard 2016. Common attributes of an event include the name of the executed activity, the timestamp of the event, the lifecycle transition (such as "start" or "complete"), and the organizational resources or roles associated with the event. Events in a stream are ordered sequentially by their timestamps, and other attributes may have case-specific information. The event prediction problem tackled in this work is that of predicting both the next activity to be executed and its corresponding lifecycle transition for an ongoing trace. The sequence for such predictions is drawn using a fixed length prefix of previous events present within the trace.

B) Data Pre-processing

Most previous studies, except a few, have not paid much attention to the pre-processing of data. The stages involved in data preparation, such as cleaning, encoding, reduction, and extraction, play a major role in the performance of classifiers.

N-gram encoding: First of all, our approach starts with the sequence encoding step. One relevant preprocessing step is the translation of character strings - such as the activity being executed in each event - to numerical input features. Leontjeva et al. [15] compared different techniques for encoding sequences with the goal of process outcome prediction and accentuated that choosing an appropriate encoding method is crucial since it strongly impacts the performance of a machine learning model. Event sequence data is intrinsically characterized by the presence of relations and dependencies among events. We employ n-gram encoding, which was underlined by Caragea et al. [16], a method that generates all continuous subsequences to capture relationships between adjacent items. By utilizing n-grams of different sizes, this model is able to capture both local and global patterns of event sequences.

Definition 1: *Given a sequence of events* $E = (e_1, e_2, \dots, e_{N+(n-1)})$*, where E is in an event universe* ϕ *and N and n are positive integers, an n-gram of E is any subsequence of n-consecutive events. There are N such n-grams in E. The total number of possible unique n-grams in the event universe is* $(|\phi|)^n$*, where* $|\phi|$ *denotes the total number of unique events in the process log data.*

As such, consider an event sequence such as $E = \{A, F, G, C, L, B, A, D, A, M\}$. Then, the 2gram or bigram features include all two-event combinations, that is $\{AF, FG, GC, \dots, AM\}$. Also, the trigram or 3-gram features will include $\{AFG, FGC, GCL, \dots, DAM\}$, and so on. In this representation, each input feature space combines multiple n-grams of different predefined sizes. That is, a 5-grams input feature space would include unigrams (1-gram), bigrams (2-grams), trigrams (3-grams), quadragrams (4-grams), and pentagrams (5-grams). Considering 15 unique events as the alphabet, the total size for features would be calculated as follows:

$$N_{\text{total\_features}} = 15 + 15^2 + 15^3 + 15^4 + 15^5 = 813{,}615$$

The n-grams approach represents a very comprehensive method, as the alphabet is known upfront, in this case, the set of unique activities executed in the process events. It applies to any domain, is rather efficient as one pass suffices, and at the same time is simple. Applications range from protein classification, information retrieval to process mining. Such an n-gram representation of event data automatically has the advantage that no further pre-processing, like aligning sequences of activities, has to be carried out. This method is especially effective because, in addition to encoding elements, it automatically captures the order. However, as can be seen from the example above, the size of the input feature set that will be generated for classification problems can get very large. The number of features increases exponentially with the length of the n-gram. It would be too costly to directly use all these features, and the sparsity of the input data could reduce the accuracy of the models. We employ feature hashing-a common technique for reducing dimensionality-to shrink the sizes of the n-gram feature vectors at limited expense to their utility.

Feature hashing: It is a powerful method for dimensionality reduction, whereby a high-dimensional input space can be projected to a lower dimensional space. -Weinberger et al., 2009. It has seen applications with great success in many fields, especially on NLP tasks related to news categorization, spam filtering, sentiment

analysis of social networks, and on many areas of bioinformatics as discussed in [14] [16], Da Silva et al. [17]. In feature hashing, the central idea relies on the usage of hash functions to map the n-grams of events into features. These compact feature vectors can then be used as input for training machine learning classifiers. To determine an appropriate dimensionality for the hashed feature space, we empirically evaluated several bit sizes, including 8, 10, 12, and 14. A 10-bit hash space (i.e., 2^{10} = 1024 dimensions) was selected as it provided a balanced trade-off between model performance and computational efficiency. Lower values (e.g., 8-bit) resulted in excessive collisions that degraded predictive accuracy, while higher values (e.g., 12- or 14-bit) offered only marginal improvements in accuracy but significantly increased memory consumption and training time. Moreover, under Zipfian distributions common in event log sequences, most n-grams occur infrequently, and collisions among these low-frequency features are less likely to affect model performance. Thus, a 10-bit hash was found to be optimal in minimizing information loss while ensuring fast inference and reduced memory usage.

Definition 2*: Consider a set of hashable features N, which represents the n -grams derived from the process event sequences. Let h be the first hash function, $h: N \rightarrow \{1, \dots, m\}$, and ξ be the second hash function, $\xi: N \rightarrow \{\pm 1\}$.*
The combined feature hashing function $\Phi^{(h,\xi)}$ maps a high-dimensional input vector of size d into a lower-dimensional feature vector $\boldsymbol{m}$, where $m < d$. The i-th element of $\Phi^{(h,\xi)}(x)$ is computed as:

$$\Phi_i^{(h,\xi)}(x) = \sum_{j:h(j)=i} \xi(j) x_j$$

where $j = 0, \dots, d$ and $i = 0, \dots, m$.

Feature hashing reduces computational cost and memory consumption by reducing the feature dimensionality. However, there is a risk of information loss arising from hash collisions where different n-grams may be mapped into the same hash key. Increase hash table size-that is, the bit size-and it minimizes the collisions. Optimal bit size depends on the vocabulary of the n-gram. By analyzing the n-grams of process sequences with Zipf's law, it can be found that most features are infrequent; thus, collisions usually happen between rare variables without causing significant information loss. To balance the dimensionality reduction and information retention, we use the 32-bit murmur Hash function by Langford et al. [18] and a binary hash function to preserve an unbiased hash kernel as suggested by Weinberger et al. [19].

B) Deep Learning Model

Artificial neural networks (ANNs) have many advantages compared to other machine learning methods for supervised learning tasks. They require less formal statistical modeling, can learn complex nonlinear relationships between variables, capture interdependencies between predictors, and offer a wide range of training algorithms. Their superior performance has been demonstrated by several comparative research studies and competitive analyses [16][20]. Historically, ANNs-especially deep neural networks-have been trained by directly optimizing the loss function using stochastic gradient descent with randomly initialized weights. However, this approach can lead to long training times and lower accuracy. In the mid-2000s, more effective training methods, such as deep belief networks (DBN) and autoencoder architectures, were introduced [21]. The methodologies in question comprise two distinct phases: (1) unsupervised layerwise pre-training, during which self-supervised learning acquires high-level abstractions via non-linear transformations, and (2) supervised fine-tuning, which refines the pre-trained weights to reduce classification errors through gradient-based optimization. Research indicates that neural networks benefiting from unsupervised pre-training demonstrate superior performance compared to their counterparts lacking this component. This technique yields improved initial weight distributions, identifies intrinsic dependencies, mitigates variance, and exceeds the efficacy of conventional regularization strategieS. We employ in our work stacked autoencoders for unsupervised layerwise pre-training to extract high-level features. Fine-tuning is then carried out with the addition of a logistic regression layer at the output to ensure accurate classification. The full process, including pre-training with stacked autoencoders and the addition of the output layer, is illustrated in Figure 2.

Unsupervised pre-training with stacked autoencoders: Autoencoders are a nonlinear version of PCA, as developed specifically for modeling nonlinear interdependencies between features, following Hinton and Salakhutdinov [22]. It consists of three layers: the input layer, the hidden layer, also known as the encoding layer, and the output layer, which is also the decoding layer. By employing the nonlinear activation function, f_θ, the high dimensional input vector, $x \in [0,1]^d$, is firstly mapped into a hidden layer through the encoder, which is consisting of: To promote sparse representation and reduce the risk of vanishing gradient problems, ReLU is used here as the activation function of encoding:

$$h = f_\theta(x) = \sigma(Wx + b)$$

Here, $\theta = \{W, b\}$ denotes the encoder's parameter set, where W is a weight matrix of size $d' \times d$, and b denotes the bias. The output, $h \in [0,1]^{d'}$, is the encoded representation in the hidden layer. Further, the decoder reconstructs the input by mapping the hidden layer representation back to a reconstructed vector $z \in [0,1]^d$ using the mapping function $g_{\theta'}$:

$$z = g_{\theta^r}(h) = \sigma(W'h + b')$$

The training objective is to optimize the parameter sets $\theta = \{W, b\}$ for the encoder and $\theta' = \{W', b'\}$ for the decoder to minimize the reconstruction loss. The reconstruction loss is defined as the squared error:

$$L(x, z) = \| x - z \|^2 = \|x - \sigma(W'(\sigma(Wx + b)) + b')\|^2$$

This is solved as an optimization problem using the mini-batch stochastic gradient descent method.

Stacked Autoencoders: They are a greedy, layer-wise procedure for multi-phase feature extraction. The features extracted by one autoencoder-the hidden layer representation-serve as the input to the next autoencoder as shown on the left side of Figure 2. Each autoencoder is trained in isolation to initialize weights for the subsequent stage of supervised fine-tuning. In this work, we use a form of undercomplete autoencoder-a network with progressively narrowing hidden layers-to approach the process prediction problem.

Supervised Fine-Tuning: The weights obtained from unsupervised reconstruction-based learning of the network are fine-tuned using logistic regression so as to map the output onto class labels (conceptually illustrated on the right side of Figure 2). Here, the decoding parts of the stacked auto-encoders are discarded, a logistic regression layer is added atop the trained encoding layers. The layer added for the multi-class classification task will contain Softmax units in order to compute an estimate of class probabilities.

$$P(y = j \mid x) = \frac{e^{\theta_j}}{\sum_{i=1}^{k} e^{\theta_i}}$$

The probability of the target class y being class j, given the input x, is computed using the input vector x and a set of weighting vectors w_j, where $h_j = w_j^T x$ represents the inner product of w_j and x. The combined network is trained as a standard multi-layer perceptron to minimize prediction error. It minimizes the cost function through stochastic gradient descent, using a lock-free scheme to parallelize the process of SGD; several cores are hence allowed to participate in contributing their share in gradient updates simultaneously, by [18] [22]. To enhance reproducibility, the architecture of the stacked autoencoders used in this study is summarized as follows. For each dataset, we used a consistent architecture comprising four hidden layers with progressively decreasing neurons: 425 → 400 → 390 → 300, followed by an output softmax layer for multi-class classification. The ReLU activation function was used for all hidden layers to promote sparsity and mitigate vanishing gradient issues, while the output layer used softmax for probability estimation. This architecture was selected based on extensive tuning on the BPI 2012_A dataset and applied across other datasets due to its stable performance. For transparency, a detailed per-dataset configuration table will be included in future versions.

4 Evaluation

Table 2: Features of the dataset.

Datasets	# of unique event types	# of events
BPI_2012_W_Completed	6	72,413
BPI_2012_O	7	31,244
BPI_2012_A	10	60,849
BPI_2013_Problems	7	9011
BPI_2013_Incidents	13	65,533
Helpdesk	9	13,710

The performance of the proposed deep learning approach was evaluated by conducting experiments with variations in datasets, configurations, and evaluation objectives. The study focused on three main research questions: (RQ1) whether the multi-stage deep learning approach outperforms existing classification methods across multiple evaluation metrics, (RQ2) how it compares to alternative methods, such as LSTM-based models by Evermann et al. [2] and Tax et al. [14] or the Bayesian-regularized probabilistic finite automaton (PFA) by Breuker et al. [12], and (RQ3) whether data balancing can improve predictions for rare events. Since most business processes include infrequent but important activities, leading to imbalanced event logs, we investigated data balancing techniques that could improve the predictive accuracy of such infrequent events. Traditional resampling methods tend to overfit and do not provide enough information for cost-sensitive learning. To address this challenge, we applied RBF neural networks, which are efficient in enhancing classification performance related to imbalanced datasets according to [21]. Experiments were done on an Intel i7-5500U 2.0 GHz processor with 16 GB RAM. Data preprocessing was done in R using the dplyr library. N-grams were generated by an in-house application implemented in Java while feature hashing was carried out on Microsoft Azure ML with the Vowpal Wabbit library. Both pre-trained stacked autoencoders and the supervised learning models were implemented using the H2O open source framework. Classical classification experiments were executed with the Weka toolkit.

A) Datasets

These experiments have been performed on three real-life datasets: BPI Challenge 2012, by van Dongen [6]; BPI Challenge 2013, by [14]. All these datasets are described in Table 1, which defines the number of unique event types for output classes of a multi-class classification problem. The BPI Challenge 2012 dataset contains 262,000 events originating from 13,087 cases of a loan application process of a Dutch financial institute. The process is divided into three sub-processes: application activities (A), work items (W), and offers (O). Following previous studies [2][12][14], only the completion events were used, resulting in three datasets: BPI_2012_A, BPI_2012_O, and BPI_2012_W_Completed.

The BPI Challenge 2013 dataset is based on Volvo IT's incident and problem management system. The incident subset includes 65,533 events (13 types) for 7554 cases, the open problems subset includes 2351 events (5 types), and the closed problems subset includes 6660 events (7 types). Merging the open and closed subsets resulted in 9011 events for consistency with other studies. The Helpdesk dataset includes 13,710 events originating from 3804 cases emanating from the ticketing system of an Italian software company. Resource IDs, loan amount, problem priority, functional divisions, and process owner details were some other organizational and case attributes that were added. It was added from BPI 2012 and BPI 2013, respectively. Feature vectors created through n-grams and feature hashing, to which this extra information was added so as to enrich the feature set.

B) Evaluation metrics

The following metrics were computed to compare the performance of our deep learning approach with other classification algorithms, and the respective average accuracy, precision, recall, F-measure, Matthews correlation coefficient (MCC), and the area under the ROC curve (AUC), all adapted for multi-class classification, are reported in Table 2. True positives-(tp_i),: gives the total number of events that belonged to class i correctly classified as class i, whereas; false positives-fp_i counts those events that do not belong to class i but were classified as class i, true negatives-tni are the number of events not of class i that were correctly classified as not from class i; finally, false negatives, fn_i, are events of class i which have been misclassified as not of class i. Accuracy was the proportion of correctly classified instances with respect to all instances. Precision was the proportion of correctly classified events of that particular class given all predictions for that class, whereas recall was the true positive rate for any class. The F-measure was the harmonic mean of precision and recall. MCC calculated the correlation between true values and predicted classifications, while AUC represented the area under the receiver operating characteristic curve. Each metric is computed for an individual class; overall scores are obtained by weighting class-specific metrics according to the true class size.

Table 2: Performance metrics used for multi-class classification evaluation

Metrics	Formula
Accuracy	$\frac{1}{n}\sum_{i=1}^{l} s_i \frac{tp_i + tn}{tp_i + fn_i + tn_i + fp_i}$
Precision	$\frac{1}{n}\sum_{i=1}^{l} s_i \frac{tpp_i}{tp_i + fp_i}$
Recall	$\frac{1}{n}\sum_{i=1}^{l} s_i \frac{tp_i}{tp_i + fn_i}$
F-measure	$\frac{1}{n}\sum_{i=1}^{l} s_i \frac{p_i \text{ recision } n_i \times \text{ recall}}{\text{precision}} +$
MCC	$\frac{1}{n}\sum_{i=1}^{l} s_i \frac{tp_i \times tl_i}{\sqrt{(tp_i + fp_i)(tp_i \times fp_i + fn_i)(t}$
AUC	$\frac{1}{n}\sum_{i=1}^{l} s_i \int_0^1 tpr_i d(fr_i)$

For the experiments, 80% of each dataset was used for training and 20% for testing. Unsupervised pre-training followed by supervised fine-tuning was performed on the deep learning model in the training phase. In the process, tenfold cross-validation was applied for training the model. The overall flow was based on splitting the entire training dataset into 10 subsets, where one subset was held out for validation, while the other nine subsets were used for training at a time. This was repeated 10 times to make the results robust and to yield the best hyperparameter configuration of Vincent et al. [21]. At the end, test results were used for comparison with other approaches.

C) Hyperparameter optimization

Deep neural networks can involve over 50 hyperparameters, and optimizing them is crucial for improving learning and prediction performance. Traditional manual search relies on expert intuition to define hyperparameter values (e.g., number of hidden layers, neurons, learning rate) and test combinations through multiple training sessions. However, this process is time-consuming and explores only a limited number of combinations, often failing to achieve optimal results in high-dimensional spaces. Grid search is a brute-force approach to train models for all possible combinations of hyper-parameters with a predefined range. It produces better results compared to the manual search in the same computational time [23]. However, it suffers from the "curse of dimensionality," as the number of combinations grows exponentially with the number of hyperparameters. For this, [23] have proposed random search: randomly

choose hyperparameter combinations and train models within a given computational budget. Random search has been empirically found to outperform grid search in several studies. Here, for hyperparameter optimization, we will use random search. The key settings' parameter ranges were defined: the number of hidden layers, 3-10; neurons per layer, 10-500; sparse data handling, True/False; initial weight

Table 3: Table of Optimal Hyperparameters for BPI Challenge 2012_A Dataset with Pre-Training and Whole Network Parameters.

Parameters	Pre-Training Values	Whole Network Values
Hidden Layers (Neurons)	425, 400, 390, 300	6 Layers (4 Hidden)
Weight Initialization	Normal Distribution	-
Sparsity	True	N/A
Learning Rate	0.005	Adaptive (Smoothing Factor: 1e-8)
Momentum Coefficient	0.9	Adaptive (Decay Factor: 0.99)
Annealing Parameter	10^4	-
Epochs	-	100
Activation Function	-	ReLU
Output Layer Activation	-	Softmax
Batch Size	-	20
Regularization Penalty (L2)	-	0
Loss Criterion	-	Cross-Entropy

distribution, uniform/normal; training epochs, 10-1000; and learning rates, 0.0001-1. For adaptive learning, parameters included time decay factor, 0.99, and smoothing factor, 1e −8. Annealing rates, if adaptive learning was turned off, were between 10 and 10^{-6}. Training was stopped early if the relative improvement in log-loss was below 0.001, or after 200 models were trained for a dataset. Table 3 provides a summary of the optimal hyperparameter configuration on BPI_2012_A data set. For all the remaining datasets, hyper-parameter optimisation is conducted but optimal values are not provided here due to space limitations.

5 Results

A) Comparative Analysis (RQ 1 and RQ 2)

We started by comparing our proposed approach to traditional classification algorithms such as support vector machines (SVM), random forests, naïve Bayes, k-nearest neighbors (kNN), and C4.5 decision trees, which are considered efficient and popular methods by Wu et al. [24]. Table 4: Next event prediction, with prefix length = 5, n-gram size = 3, and feature hashing bit size = 10. Our method consistently outperforms these traditional approaches in different metrics. Among them, the best performance was achieved by SVM and got closest to our approach. On the BPI 2013 dataset, most methods, except naïve Bayes, have similar performances, while on BPI 2012 and Helpdesk, our approach has much larger performance gaps. These results show that our deep learning model is much better in making predictions, especially for complex datasets. Therefore, we can see that our proposed approach performs the best among traditional classifiers, thus answering RQ1.

To answer RQ2, we compared our approach with three recent next-event prediction approaches. In Table 5, it is shown that on all the BPI 2012 datasets our approach performs better than its alternatives. For instance, in the case of the BPI_2012_W_Completed dataset, the accuracy obtained by our method is 0.831, while by Breuker et al. (2016), it is 0.719 and 0.760 [14]. Our method has performed much better in terms of recall compared to that of Breuker et al. (2016). Precision results also show the superiority of our approach, reaching 0.811 compared to 0.658 by Evermann et al. [2]. For other datasets like BPI_2012_A and BPI_2012_O, our approach is always better for all metrics. For the BPI_2013_Incident dataset, the results are not so straightforward. Breuker et al. [14] reported a slightly better accuracy, 0.714 versus 0.663, but our approach reaches a considerably higher recall, 0.664 versus 0.377. Similarly, Evermann et al. [2] have higher precision but, in general, our approach is better when tested on the BPI_2013_Problem's dataset. In the case of the Helpdesk

Table 4: The performance metrics of various models on different datasets. Each value is reported as **Accuracy / Precision / Recall / F-score / MCC / AUC**.

Metric	BPI 2012_A	BPI2013_Incidents	Helpdesk
SVM	0.817 / 0.856 / 0.822 / 0.817 / 0.748 / 0.895	0.652 / 0.599 / 0.653 / 0.622 / 0.350 / 0.730	0.715 / 0.605 / 0.716 / 0.652 / 0.389 / 0.725
RF	0.720 / 0.714 / 0.721 / 0.712 / 0.566 / 0.888	0.615 / 0.626 / 0.616 / 0.524 / 0.508 / 0.895	0.601 / 0.619 / 0.601 / 0.606 / 0.278 / 0.688
Naïve Bayes	0.612 / 0.633 / 0.612 / 0.555 / 0.485 / 0.772	0.576 / 0.618 / 0.577 / 0.590 / 0.519 / 0.879	0.631 / 0.634 / 0.631 / 0.622 / 0.323 / 0.733
C4.5	0.708 / 0.744 / 0.709 / 0.705 / 0.674 / 0.931	0.659 / 0.558 / 0.659 / 0.582 / 0.564 / 0.900	0.613 / 0.534 / 0.614 / 0.569 / 0.214 / 0.602
Deep Learning	0.824 / 0.852 / 0.824 / 0.817 / 0.751 / 0.923	0.663 / 0.648 / 0.664 / 0.647 / 0.583 / 0.862	0.782 / 0.632 / 0.781 / 0.711 / 0.412 / 0.762

Table 5: Comparison against benchmark approaches (higher numbers are better)

Dataset	Breuker et al. [14]	Evermann et al. [2]	Tax et al. [12]	Proposed Approach
BPI 2012_W	Accuracy: 0.719 Recall: 0.578	Precision: - Recall: -	Accuracy: 0.760 Recall: -	Accuracy: 0.831 Precision: 0.811 Recall: 0.832
BPI 2012_A	Accuracy: 0.801 Recall: 0.723	Precision: 0.832 Recall: -	-	Accuracy: 0.824 Precision: 0.852 Recall: 0.824
BPI 2012_O	Accuracy: 0.811 Recall: 0.647	Precision: 0.836 Recall: -	-	Accuracy: 0.821 Precision: 0.847 Recall: 0.822
BPI 2013_Incidents	Accuracy: 0.714 Recall: 0.377	Precision: 0.735 Recall: -	-	Accuracy: 0.663 Precision: 0.648 Recall: 0.664
BPI 2013_Problems	Accuracy: 0.690 Recall: 0.521	Precision: 0.628 Recall: -	-	Accuracy: 0.662 Precision: 0.641 Recall: 0.662
Helpdesk	-	-	Accuracy: 0.712	Accuracy: 0.782 Precision: 0.632 Recall: 0.781

dataset, our approach outperforms Tax et al. [12] by a fair margin in terms of accuracy: 0.782 versus 0.712. Furthermore, random hyperparameter optimization instead of a manual search shows dramatic improvements over our previous work. We also examine the impact of different sizes of n-grams and bit size of feature hashing on performance. As it turned out, on many datasets, including BPI 2012 and Helpdesk, increasing the size of n-grams above 5 does not bring about noticeable accuracy improvements while noticeably increases computational costs. For instance, in BPI_2012_A dataset with prefix length of 5, the accuracy is between 0.829 and 0.831 with n-gram size ranging from 2 to 5, which is almost no improvement. Also, increased feature hashing bit size above 10 does not contribute more considerably because hash collisions among less frequent n-grams take precedence. This will come according to Zipf's law, which states that only a few input features are high-frequency ones.

B) Imbalanced Classification (RQ 3) Class imbalance problem has always been one of the important challenges

faced by machine learning techniques, as the latter may fail to recognize the minority class examples. Wang and Yao, 2012 proposed several data-level techniques such as under/oversampling and algorithm-level approaches including cost-sensitive learning and boosting. Sun et al., [4] resampling techniques such as SMOTE by Huang et al., 2016 synthesize artificial samples to reduce class imbalance. However, resampling techniques might introduce noise. Cost-sensitive methods can be effective but usually require expert knowledge to determine the appropriate costs. In our study, we utilized the RBF neural

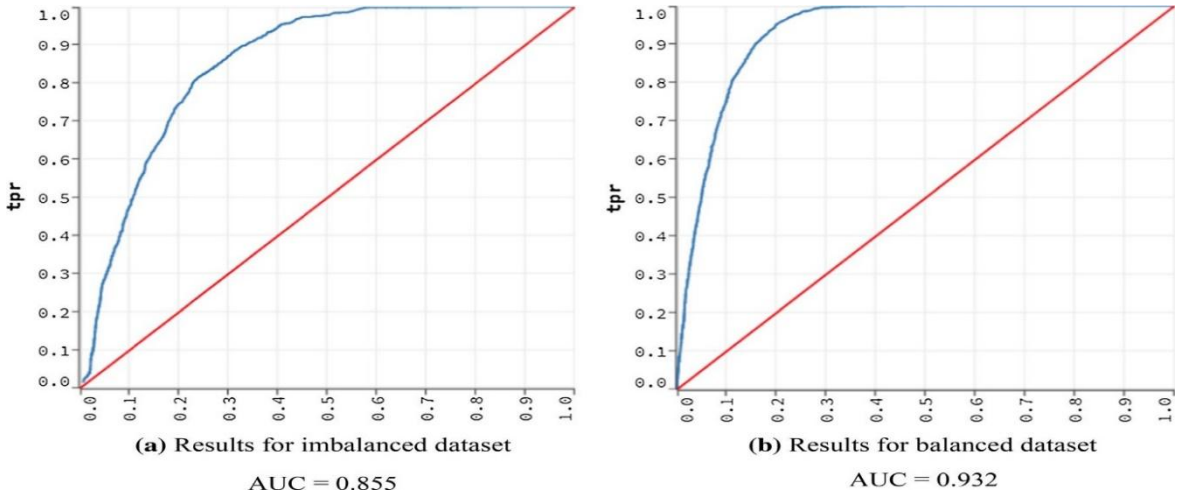

Figure 3: ROC curves for (a) imbalanced datasets and (b) balanced datasets. The ROC curve represents the relationship between the true positive rate (TPR) and the false positive rate (FPR).

network-based data generation method in order to balance the dataset. Unlike other methods, RBF maintains dependencies among input variables by extracting Gaussian kernels and generates data proportional to the minority class distribution. Hence, it would have better representation for the minority class without loss of information. The pseudo-code and details can be found in [8]. Due to rare but critical "Wait user" events, in case of the BPI Challenge 2013 Incidents data we have reformulated the problem into a binary classification problem. An RBF-based method was employed in rebalancing of the dataset prior to using our prediction model. We performed result comparisons on the imbalanced data-without rebalancing-and RBF-rebalanced data. Since accuracy is not suitable for imbalanced datasets, the AUC was used as the evaluation metric for this work [17]. Figure 3 shows the ROC curves of both cases, where the AUC increases from 0.855 in the case of imbalanced data to 0.932 in the case of RBF-rebalanced data.

6 Discussion

This paper, for the first time, investigates the effectiveness of a deep learning approach based on stacked autoencoders for future event prediction in ongoing process instances. Our approach is compared to three recent methods, including two LSTM-based models [2, 14] and a Bayesian PFA [12], as summarized in Table 1. The comparative analysis shows that our approach consistently outperforms prior methods on most datasets, with up to 83.1% accuracy, 85.2% precision, and 92.3% AUC. For instance, while Breuker et al. [12] achieved a maximum of 76.0% accuracy, our method improved this by over 7 percentage points on the BPI 2012_W dataset. Similar trends are observed across BPI 2012_A, 2012_O, and the Helpdesk dataset. This improvement is largely due to the multi-stage architecture of our model, which combines effective feature encoding (n-grams with feature hashing), unsupervised pre-training, and hyperparameter optimization. Our use of n-gram encoding captures both local and long-range dependencies in event sequences, unlike the traditional one-hot encoding used in earlier LSTM models. Furthermore, feature hashing reduces dimensionality while preserving semantic relevance, enabling real-time inference. The unsupervised pre-training using stacked autoencoders helps in extracting higher-order representations and better weight initialization for the supervised fine-tuning phase.

Despite these advantages, our method does show some limitations. For the BPI 2013_Incident's dataset, Breuker et al. [12] slightly outperformed us in accuracy (71.4% vs. our 66.3%). This performance drop may be attributed to the rarity of certain event types and the dynamic nature of incident management processes, which exhibit high variance in sequential patterns. In such scenarios, deeper models like LSTMs may generalize better when sequence lengths vary significantly. We also note that increasing n-gram size or feature hashing bit size beyond optimal thresholds (e.g., n=5, bit size=10) does not yield further performance gains and can even degrade accuracy due to increased sparsity and hash collisions. Thus, our model's performance is sensitive to hyperparameter tuning, although random search mitigates this challenge to a large extent.

In terms of generalizability, our framework demonstrates strong results across six diverse real-world datasets. However, our method assumes stationary process behavior, meaning it does not adapt to concept drift or evolving process dynamics. This limits its applicability in highly dynamic environments without retraining. Moreover, while the model performs well on numerical event log attributes, it does not yet incorporate textual or unstructured data, such as freeform user comments or emails, which may contain rich contextual information.

7 Concluions

This study introduced a novel deep learning framework using stacked autoencoders and n-gram-based feature hashing for predicting next events in business process logs. Extensive experiments on multiple real-world datasets confirmed the superior performance of the proposed approach over state-of-the-art methods. The integration of unsupervised pre-training, effective sequence encoding, and robust hyperparameter optimization contributes to its enhanced predictive capabilities.

Additionally, our approach addressed the issue of imbalanced datasets through RBF-based synthetic data generation, which significantly improved the AUC for rare class prediction. The methodological contributions, combined with empirical validations, underline the feasibility of applying advanced deep learning techniques in the domain of business process monitoring.

Future research will focus on addressing concept drift, incorporating unstructured data, and improving model interpretability. Utility-based evaluations and explainability modules will be explored to support deployment in real-world enterprise settings where decision accountability is critical.

References

[1] Yuan, Z., 2024. Consumer behavior prediction and enterprise precision marketing strategy based on deep learning. *Informatica*, *48*(15). DOI: 10.31449/inf. v48i15.6260

[2] Evermann, J., Rehse, J.R. and Fettke, P., 2017. Predicting process behaviour using deep learning. *Decision Support Systems*, *100*, pp.129-140. DOI: 10.1016/j.dss.2017.04.003

[3] Jing, L., 2024. Evolutionary deep learning for sequential data processing in music education. *Informatica*, *48*(8). DOI: 10.31449/inf. v48i8.5444

[4] Karna, H., Gotovac, S. and Vicković, L., 2020. Data mining approach to effort modeling on agile software projects. *Informatica*, *44*(2). DOI: 10.31449/inf. v44i2.2759

[5] Gregor, S. and Hevner, A.R., 2013. Positioning and presenting design science research for maximum impact. *MIS quarterly*, pp.337-355. DOI: 10.25300/MISQ/2013/37.2.01

[6] Van Dongen, B.F., 2015. Bpi challenge 2015. In *11th International Workshop on Business Process Intelligence (BPI 2015)*. DOI: 10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1

[7] Rogge-Solti, A. and Weske, M., 2013. Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In *Service-Oriented Computing: 11th International Conference, ICSOC 2013, Berlin, Germany, December 2-5, 2013, Proceedings 11* (pp. 389-403). Springer Berlin Heidelberg. OI: 10.1007/978-3-642-45005-1_27

[8] Forman, G. and Kirshenbaum, E., 2008, October. Extremely fast text feature extraction for classification and indexing. In *Proceedings of the 17th ACM conference on Information and knowledge management* (pp. 1221-1230). DOI: 10.1145/1458082.1458243

[9] Senderovich, A., Di Francescomarino, C., Ghidini, C., Jorbina, K. and Maggi, F.M., 2017. Intra and inter-case features in predictive process monitoring: A tale of two dimensions. In *Business Process Management: 15th International Conference, BPM 2017, Barcelona, Spain, September 10–15, 2017, Proceedings 15* (pp. 306-323). Springer International Publishing. DOI: 10.1007/978-3-319-65000-5_20

[10] Le, M., Gabrys, B. and Nauck, D., 2017. A hybrid model for business process event and outcome prediction. *Expert Systems*, *34*(5), p.e12079. DOI: 10.1111/exsy.12079

[11] Unuvar, M., Lakshmanan, G.T. and Doganata, Y.N., 2016. Leveraging path information to generate predictions for parallel business processes. *Knowledge and Information Systems*, *47*, pp.433-461. DOI: 10.1007/s10115-015-0896-2

[12] Breuker, D., Matzner, M., Delfmann, P. and Becker, J., 2016. Comprehensible predictive models for business processes. *Mis Quarterly*, *40*(4), pp.1009-1034. DOI: 10.25300/MISQ/2016/40.4.06

[13] Márquez-Chamorro, A.E., Resinas, M., Ruiz-Cortés, A. and Toro, M., 2017. Run-time prediction of business process indicators using evolutionary decision rules. *Expert Systems with Applications*, *87*, pp.1-14. DOI: 10.1016/j.eswa.2017.05.037

[14] Tax, N., Verenich, I., La Rosa, M. and Dumas, M., 2017. Predictive business process monitoring with LSTM neural networks. In *Advanced Information Systems Engineering: 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings 29* (pp. 477-492). Springer International Publishing. DOI: 10.1007/978-3-319-59536-8_30

[15] Leontjeva, Anna, and Fabrizio Maria Maggi. "Complex symbolic sequence encodings for predictive monitoring of business processes." In *Business Process Management: 13th International Conference, BPM 2015, Innsbruck, Austria, August 31--September 3, 2015, Proceedings 13*, pp. 297-313. Springer International Publishing, 2015. DOI: 10.1007/978-3-319-23063-4_19

[16] Caragea, C., Silvescu, A. and Mitra, P., 2011, November. Protein sequence classification using feature hashing. In *2011 IEEE International Conference on Bioinformatics and Biomedicine* (pp. 538-543). IEEE. DOI: 10.1109/BIBM.2011.61

[17] Da Silva, N.F., Hruschka, E.R. and Hruschka Jr, E.R., 2014. Tweet sentiment analysis with classifier ensembles. *Decision support systems*, *66*, pp.170-179. DOI: 10.1016/j.dss.2014.07.003

[18] Langford, J., Li, L. and Strehl, A., 2007. Vowpal wabbit online learning project. *Vowpal Wabbit online learning project*.

[19] Weinberger, K., Dasgupta, A., Langford, J., Smola, A. and Attenberg, J., 2009, June. Feature hashing for large scale multitask learning. In *Proceedings of the 26th annual international conference on machine learning* (pp. 1113-1120). DOI: 10.1145/1553374.1553516

[20] Caruana, R. and Niculescu-Mizil, A., 2006, June. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning* (pp. 161-168). DOI: 10.1145/1143844.1143865

[21] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A. and Bottou, L., 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, *11*(12). DOI: 10.1162/neco.2006.18.7.1527

[22] Hinton, G.E., 2006. A Fast-Learning Algorithm for Deep Belief Nets. *Neural Computation*.

[23] Bergstra, J. and Bengio, Y., 2012. Random search for hyper-parameter optimization. *Journal of machine learning research*, *13*(2). DOI: 10.1162/neco.2006.18.7.1527

[24] Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Yu, P.S. and Zhou, Z.H., 2008. Top 10 algorithms in data mining. *Knowledge and information systems*, *14*, pp.1-37.