Reverse-Time Event Sequence Prediction Using Summary Markov Models and Denoising Diffusion Processes

Jiao Wu
School of Computer Engineering, Shangqiu University, He Nan, Shang Qiu, 476000, China
E-mail：wujiao555888@126.com

Keywords: reverse time prediction, event sequences, stochastic systems, Markov Models

Recieved: March 19, 2025

Choosing a suitable subset of the event information is the first step in using the framework. We are unable to provide assistance with this endeavor since it is very customized and requires creativity. With any luck, the event data will be adequately represented by a probabilistic framework that is fitted using a learning algorithm once a dataset has been constructed. The algorithm's output is a representation suitable for use in prediction. This can only be achieved by feeding the framework event data from execution process instances. With the use of the probabilistic model, we can assess the chances of the procedure continuing using various event sequences given the current sequence of occurrences. Specifically, this feature may be used to foretell the process's continuation-inducing event type with the highest probability. Verifying that the prediction model does not go against common sense is something a framework user may want to undertake. The first step of this study is to use a transformation method to derive the conceptual framework from the probabilistic model. It is possible to see and understand this conceptual paradigm. A user may check how well the model matches his expectations. The outcome would depend on his expectations; he may find behavior that goes against them or decide that the probabilistic framework is sufficient. In the second scenario, a problem analysis might be conducted to determine whether the expectations were incorrect or if the probabilistic framework is insufficient. When the model's transformation algorithm produces a conceptual model that is too complicated for humans to understand, the framework offers algorithmic assistance to those who need it. To be more specific, one may utilize model query techniques to check for certain patterns in the model. The user's expectations on the presence or absence of model structure may be represented by the patterns. Any model, no matter how complicated, may be tested through comparing this expectation to the query results. The capacity of Markov Chain Models is used to handle sequential data—that is, to "remember" information from earlier events in the sequence as they go backwards through time—makes them ideal for reverse time prediction. Finally, we evaluate the order model (or memory) of time series related to electrocorticographic (ECG) data recorded epileptic episodes by making use of the latter attribute. Improved prediction accuracy and correlation efficiency are the results of a novel method that merges estimates from forward predictors and backward predictors. We prove that events may be informed by changes on Markov Chain Systems by analyzing dynamic graphs built from time-series data, i.e., time-series fluctuation. In a stochastic model known as a Markov chain, the previous state is irrelevant to determining the subsequent state; instead, the present state is used exclusively. This served as inspiration for the suggested encoding technique, which aims to provide accurate and interpretable predictions of time-series events. In a process that is fundamentally inverted from the conventional Markov chain prediction, the conditional likelihood of the prior state is computed given the present state. The experimental findings from five real-world datasets demonstrate that our method outperforms baselines and offers other explanations for the outcomes of event prediction.

Povzetek: Študija uvaja pristop obratnega napovedovanja zaporedij dogodkov z združevanjem povzetkovnih Markovih modelov in difuzijskih postopkov. Metoda izračuna verjetnosti preteklih stanj glede na trenutno stanje, izboljša napovedovanje časovnih serij in preseže primerjalne modele na petih realnih podatkovnih zbirkah.

1 Introduction

The goal of sequential event prediction is to forecast the subsequent event in a current event sequence by using a "sequence database" of previous event sequences as a learning resource [1], [2]. We concentrate on applications where the collection of historical occurrences, rather than the precise sequence of those occurrences, has predictive potential. These kinds of applications may be found in medical informatics, equipment maintenance [3], [4], recommender systems, and other fields. Forecasting

purchase timings, transaction patterns, and social media activity are just a few of the numerous real-world uses for sequence prediction. Since the challenge entails the difficult job of concurrently modeling two difficult data types—categorical data for event categories and strictly positive continuous data for inter-arrival times—it calls for a specialized model [5], [6]. Our method improves efficiency and performs noticeably better than current baselines for long-term forecasting. Our experimental research sheds light on how the model does this, showing that it is more effective at forecasting distant events and can capture more intricate association patterns. The objective is to guess the remaining (hidden) events in the sequence using the set of disclosed events—not necessarily their order. To create the forecasts, we may consult a "sequence database" of historical event sequences [7]. Every time a new event is announced, predictions for the next one are updated. Sequential prediction issues have numerous instances. Medical illnesses develop throughout time, and those that a patient has already encountered might be utilized to forecast future problems. The time-series prediction issues that one may solve using a Markov chain are not the same as the sequential event prediction problems that we examine here [8], [9]. For example, there is no inherent order in the online grocery shop recommender system dilemma about which items should be put to the basket, and the order of the addresses is probably not very important when it comes to email recipient recommendations [10], [11]. The remaining sequence can only be predicted with the set of previous elements. We approach the sequential event prediction process as a supervised ranking problem at each stage [12], [13]. Our algorithms rate all other potential events based on their probability of becoming a following event in the sequence, given a set of disclosed events from the present sequence. How far down the list we must search to locate the next item or items to be added determines how accurate our forecast is [14], [15]. A substantial body of literature has focused on solving these problems, which include complicated relationships (e.g., geographical, temporal, and semantic), heterogeneous multi-faceted outputs, and streaming data feeds. Since event prediction issues are highly multidisciplinary, the majority of current event prediction techniques were developed to handle particular application domains [16]. However, the concepts and evaluation processes used are often applicable to other domains as well. In the lack of an exhaustive literature review for event prediction, however, cross-referencing the methods across domains is both necessary and challenging [17]. Much effort has been put into developing and using event prediction techniques in order to tackle these problems in recent years. Though event prediction methods are often still in their early stages, there has been a recent upsurge of research that suggests and implements novel ways across several domains [18]. Although most current event prediction systems have been developed for particular use cases, their underlying principles are often generic enough to address issues in other use cases as well. Regrettably, it is challenging to use these methods in varied contexts for different groups. The unique characteristics of the subject matter necessitate complex and purpose-built evaluation strategies to ensure high-quality event prediction results. These features include, but are not limited to, the fact that the subject matter is multi-objective (e.g., focused on accuracy, resolution, efficiency, while lead time) and that the prediction results are heterogeneous (e.g., with multiple outputs). Unfortunately, there are currently no methods for systematically standardizing and summarizing the several suggested event prediction models.

1.1 Challenges

- **Data sparsity:** Precisely predicting the chain of events that leads to a given result may be challenging in the absence of sufficiently rich data.
- **Events association:** While the model is good at seeing patterns of association, it isn't perfect at capturing the linkages that really drive the events.

Within seconds, contemporary network settings produce a massive amount of network events. Consequently, there is a lot of labor involved in analyzing network events, especially when there are a lot of collected network variables. It is essential to convert the information into time series forms with equal intervals in order to examine data from time series network events. A huge number of streaming intrusion signals were previously analyzed using aggregation [6]. Nevertheless, due to the large number of events occurring at time *bit*i, the aggregation method used to time-stamped events can be prone to significant variance. To better predict when a network will be healthy or under assault, we provide a method that can transform several sequences of network activity into time series representations. In particular, our strategy applies statistical measures and computational approaches to transform network events into time series data over a desired time scale, allowing us to extract temporal properties. When representing data with several dimensions in a two-dimensional space, PCA is a common dimension reduction method used in visualization. All data instances are mapped into the x- and y-axis of the 2D space using the first and second main components that are calculated. By using the Fisher-Jenks algorithm, also known as Jenks' natural breaks categorization approach, to create clusters, the predicted network events may be more accurately represented. The optimal classification of values into several groups is found by it. Three separate groups were established from the various normal and assault occurrences via the use of a categorization procedure. Our suggested technique is shown to be successful in detecting attack patterns and projecting

probable future assaults using a publicly accessible network traffic dataset acquired in a honeypot system. We tested two-time scales for network attack event prediction and attacker temporal pattern understanding to find out how well they worked. To further test the efficacy of our suggested method, we conducted a battery of data analysis. Ultimately, this research adds the following to the existing body of knowledge:

- A state-of-the-art method for extracting time-related characteristics from network events is presented, which combines statistical measures with wavelet transform and permutation entropy. Additionally, we provide an innovative approach to retrieving time-related characteristics from categorical information pertaining to network traffic.
- The suggested method's efficacy is assessed by performance evaluations with varying time scales (Zs=60ts=60 and 120 s).
- The characteristics are evaluated and numerous outputs (such as network normal along with attack events) are forecasted using DL.
- The efficacy of evaluating network events using the retrieved temporal characteristics is determined by visual analysis using several visualization approaches.

In Section 2 of this publication, the relevant work is described. Section 3 follows with an explanation of our methodology and the dataset consisting of network traffic. Section 4 presents the outcomes of the experiments. Finally, Section 5 and Section 6 cover the study's and future research's implications.

2 Related works

The focus of the study conducted by the authors of [19] is on analyzing the performance of different machine learning algorithms using recorded EEG data. The goal of that research is to identify the optimal model for use in developing an ensemble model to enhance learning. Logistic regression, the Naive Bayes model, and the K-nearest neighbors Random Forest are examples of traditional machine learning approaches; artificial neural networks, convolutional neural networks, autoencoders, and long short-term memory are examples of deep learning techniques. When comparing the sensitivity and specificity of all the models in that investigation, Random Forest with Long Short-Term Memory stood out.

That study systematically examines the impact of varying noise levels on the accuracy of forecasting of different load components and compares four recurrent neural network frameworks [20]. When it comes to horizontal bending moment and torsional moment prediction, the GRU network excels, whereas the LSTM network excels in vertical bending moment prediction. Despite applying filters to the initial noise information, the prediction accuracy continued to decline with increasing noise levels. In comparison to the torsional moment, the vertical and horizontal bending moments are better at predicting future behavior.

Finding an efficient method for handling time series data with missing values and volatility is the goal of the authors of [21]. For accuracy evaluation, they used RMSE, and to estimate execution time, they used the Python Timeit module. According to the results, the LSTM model outperforms the ARIMA model with respect to accuracy in a dataset of 60 data points (RMSE 0.037618). The situation changes, however, when they look at a bigger dataset with 228 pieces of data and see that the ARIMA model outperforms the LSTM model in terms of accuracy (RMSE 0.006949 vs. 0.036025). The LSTM model often beats the ARIMA model in missing data settings, although both models' performance drops down as the total amount of missing values increases. In terms of speed, the ARIMA model is light years ahead of the LSTM model.

Using data from 299 individuals suffering from left ventricular systolic dysfunction and class III/IV heart failure, the authors of [22] want to do a comprehensive survival analysis with survival prediction. In a survival analysis, the number of participants who managed to stay alive after a mediation is tracked over time to determine the impact of that mediation. Survival time is defined as the amount of time it takes to get from a certain starting point to an endpoint, such as death, and survival analysis is the study of that time. Cox Potential Hazard regression and Kaplan-Meier estimations were used to conduct the study. KM plots displayed the estimated survival rates vs each clinical trait, as well as the impact of each feature at different levels on survival over time. The study's clinical characteristics were utilized to estimate the hazard of mortality event using Cox regression. Time, age, ejection fraction, and serum creatinine were shown to be substantial risk factors for advanced heart failure based on the research. The likelihood of survival decreases with age and increases in serum creatinine levels. The likelihood of a fatal incident decreasing by around 5.2% is directly correlated with the EF level, which has a positive impact on survival. Early postdiagnosis days have a higher death rate, but once a specific amount of time has passed, the risk begins to decline. Additional risk factors include hypertension and anemia.

Researchers in [23] conducted multi-step-ahead predictions of network bandwidth use using statistical techniques, such as auto-regressive integrated moving average, and machine learning algorithms, such as multi-layer perceptron as well as short-term memory. In order to anticipate the network's bandwidth, use one, two, and three hours in the future, they suggest comparing several methods, including recursive, regulate, multiple-in multiple-out, or variants on these themes. According to

the results, MIMO was superior for forecasts two and three hours in advance, whereas the direct method was superior for predictions one hour in advance. Additionally, the multi-stage methods were beneficial to all three prediction algorithms. They emphasize the significance of their findings for telecom network long-term forecasts.

The forecasting of time series of total everyday solar energy output has been investigated by the researchers of [24] using approaches based on machine learning. Prior to comparing its performance to two other common machine learning techniques, support vector machine, while artificial neural network, the time series is first modeled using the seasonal variant of the well-known classical approach auto regressive integrating moving average. The impressive results achieved by SVM showcase the promise of machine learning-based approaches in that field. Nevertheless, in order to have a better understanding, they need to investigate the causes of ANN's poor yield. Although SVM has shown some success in predicting solar production, there is still room for improvement in terms of overall accuracy. Future research should focus on finding ways to do that.

A technique for predicting when flights would take off using deep learning was suggested in [25]. The first section of that study is an analysis of the variables that affect flight departure time as well as what those factors are. Second, the article builds a GRU model, analyses how various hyperparameters affect network performance, and uses parameter tweaking to find the best hyperparameter combination. Lastly, ZSNJ's actual flight data is used for model verification and comparison analysis. The established model has the following evaluation values: an RMSE of0.42, a MAPE of6.07, and an MAE of 0.3. The model presented in that research reduces RMSE by 16%, MAPE by 34%, and MAE by 86% when compared to existing delay prediction models. Due to the model's excellent forecast accuracy, airport scheduling with collaborative decision-making may be implemented with confidence. Table 1 shows the cons and pros of the model,

Table 1: Comparison analysis models

Model	Cons	Pros
Reference [19]	Very attuned to extreme cases. Firm presumptions.	Competence in managing various time series elements and characteristics. Excellent comprehensibility.
Reference [20]	Very attuned to extreme cases. Compact intervals of certainty.	Aptitude for dealing with level, trend, along with seasonality components that are changeable. Optimization via automation.
Reference [21]	Further information is needed. Rigid guidelines and presumptions. Not easily automatable.	Excellent comprehensibility. Is confidence interval realistic? Objective predictions.
Reference [22]	The holdout error is higher. Extended periods of instruction and assessment.	Excellent comprehensibility. Clearer than competing models. Handles unknowns with ease. Manage the component variance.
Reference [23]	Not really easy to understand. Creating confidence intervals for the predictions is a challenging task. Further information is needed.	Minimal presumptions and constraints. Capability to manage intricate nonlinear situations. Extremely accurate forecasting capabilities. Is amenable to automation.
Reference [24]	- The weight initiation is a determining factor. -Problems with hardware reliance, overfitting, and generalizability - An issue with local minima	It is more generic and versatile, can handle numerous input variables, supports multivariate inputs, has excellent results for nonlinear time series, and can properly map input and output connections. It may not need a scaled or stationary time series as an input.
Reference [25]	- Strictly limited in its ability to handle real-world issues - I am unable to manage corrupt or missing files.	Makes use of lag and shift in previous data "Increase precision" with the use of a regression model and a moving average

3 Methodology

A. Event sequence dynamics

A knowledge graph that describes events, event classes, as well as event (class) interactions using "events" as its unit of knowledge. In addition, the event knowledge graph represents the norms and trends of event occurrence or progression via a series of scenario models. All of this information forms the basis of our suggested strategy for predicting future events. To ensure that this work is comprehensive, this part provides the necessary concepts of an occurrence knowledge graph and describes the event prediction.

Event knowledge graph model

The event knowledge graph depicts a scenario in which an event takes place at a given instant in time inside a certain setting, with several actors and displaying distinct action traits and state transitions. To define an event in the event knowledge network, we refer to the definition in.

$$E::=\langle \Lambda, P, O, L, T\rangle \quad (1)$$

The five components that make up an event are the building blocks of knowledge. The event's collection of activities is denoted by A. P stands for the set of people who took part in the event. O is the collection of things that are carriers of the event. The event takes place at venue L. Time, denoted as T, is the duration of the event. Among the elemental connections that may be found between events and their elements are has Participant, hasObject, hasAction, hasLocation, and hasTime.

With an event ontology and many event instances, the event knowledge graph is a knowledge base that is constructed around events and designed for various application domains. Here is a description of what it means.

$$EKG::=\langle EOs, EIs\rangle \quad (2)$$

EOs stand for the event ontology, whereas EIs stand for the event instance library. Concepts of event classes, connections between event classes, and rules for assertion and inference about event classes make up the event ontology. This is the best way to explain what the event ontology is.

$$|\ EOs::=\langle ECs, Event\ -\ Relations, Rules\rangle \quad (3)$$

ECs stand for the collection of event types. A taxonomic relation (is _ a) while logical relations (such as causal, isComposedOf, follow, while concurrence) are all part of the collection of event class relations that are represented by Event _ Relations. Rules are a set of inference rules that may be used to construct different types of reasoning for event classes. Among the models included in the event ontology are those representing event hierarchies and event scenarios. Connecting the various event types via logical relationships creates the event scenario model. Aside from that, it mirrors the pattern of the happening of a first occurrence and a series of following events that are put in motion by it. A taxonomy connection and event classes make up the event hierarchy model.

Numerous event instances, described as follows, are housed in the event instance library.

$$Els::=\langle Events, Event\ -\ Relations\rangle \quad (4)$$

in which the collection of event occurrences is represented by Events and the relations between those instances are represented by Event _ Relations. A particular event is represented by an event instance, which is a concrete implementation of the idea of an event class. It is usual practice to get event instances from databases or text. The inherent logic and evolutionary trends between events are expressed by the identical logical relations among event instances as among event classes.

It is our hope that the computer, given the present situation, can comprehend "what happened" and provide "what will happen ahead" when it comes to event prediction using event knowledge graphs. Allow me to paint a picture to help clarify. The boy's keen perception alerts them to certain happenings the moment he enters the door to his house. First, his instructor is chatting with his parents; second, his parents' faces reveal their frustration. Maybe the lad has been through this before and knows it's not a good sign; maybe a "storm" is on the horizon. We want to provide robots with this capacity to learn from experiences and respond appropriately. The first occurrence and any potential concurrent events are used to extract the characteristics of the occurrence scenario information, as previously stated. What follows is a more formalized version of the event scenario details:

$$ES = EI_e(A_e, O_e, L_e, T_e, P_e) + \sum_n EI_{ec}(A_{ec}, O_{ec}, L_{ec}, T_{ec}, P_{ec}) \quad (5)$$

EI_c signifies the first event e that contains the argument data, while EI_c stands for the data associated with the event ec that occurs concurrently with e. ES represents the scenario information's feature vector. We represented the goal of predicting the occurrence of an event using scene data as a multilabel classification problem. In order to determine the likelihood of each event type, we developed prediction algorithms to analyze the scenario data. To get the result vector, we first projected the scenario data into several result spaces using a linear function. The next step

was to create an analytical function that would take the result vector and return the occurrence probabilities of each event type.

$$W_R = \text{Linear}(ES_e) \quad (6)$$

$$P(\ell) = \text{Sigmoid}\big(\log\_\ \text{soft}_{\max}(W_R)\big) \quad (7)$$

The global fluctuations in the multivariate events sequence are conditionally homogenous given the history, and this may be expressed by a probabilistic parameterization of an ordered procedure over labels in L $\Theta = \{\Theta_X : X \in \mathcal{L}\}, \Theta_X = \{\theta_{x|h}\}$ s.t. $\sum_{X\in\mathcal{L}} \theta_{x|h} = 1$ for every distinct history h, where $\theta_{x|h}$ denotes the likelihood of occurrence label X happening at any point in the given history's sequence. With just a portion of the event labels' dynamics taken into account $\mathbf{X} \subseteq \mathcal{L}$, Assuming that the set L\X is not empty, we create a matching random variable X that has one state for every tag in X and one state for if the label belongs to L\X. If X is a complete subset of all labels, then ($\mathbf{X} \subset \mathcal{L}$), there are $|\mathbf{X}| + 1$ states of X; we denote these as x. We use $\tilde{\Theta}_{\mathbf{X}}$ to represent the total of all possible outcomes across all labels $\mathbf{X} \subseteq \mathcal{L}$, i.e. $\tilde{\Theta}_{\mathbf{X}} = \{\tilde{\theta}_{x|h}\}$ where $\tilde{\theta}_{x|h} = \sum_{X\in\mathbf{X}} \theta_{x|h}$, which is given in Fig 1.

(i). Missing data imputation

We provide an ensemble approach to the missing data issue in time series forecasting by combining two forecasting models with advantageous diversity—that is, models that function on various time series dynamics—into a single model. Time series values may be related to earlier time series values due to a certain dynamic that controls the forward flow of information. Time series values may also be stated in terms of future values thanks to the backward dynamic. Since the two dynamics are distinct, they will provide the ensemble with variety. Take the interval from t=M + 1 to t=M +J and assume it is absent from the time series.

We take into account the forward-looking model that estimates the missing variable at time t by predicting it using the lagged prior values.

$$t = M + 1 \quad (8)$$

we use xM−L−1, ..., x ˙ M as a parameter for our model. In addition, we create a "backcaster" model for backward forecasting that uses future values of the time series to predict missing values; for example, if we want to estimate the missing value at epoch t=M+J, we may utilize the following inputs: xM+J+1,..., x ˙ M+J+L. The next step is to put up an ensemble that includes both the forecaster as well as the back caster. To be sure, this is just the beginning. After the missing values have been estimated for the time series, we combine the initial training patterns with the new ones to form an augmented training set. To get stronger models, we retrain the forward and backcasting systems using this more comprehensive training data. This cycle of iterative improvement continues until the additional benefit is no longer noticeable.

B. Markov Models (MMs)

To identify the minimum influencing set, it is sometimes necessary to model the dynamics of a single or small set of labels that are important to the modeler. This is the case in many applications. In the context of knowledge discovery using events sequence datasets, this is very relevant. For every label set X∈L and every SMM function s(·), the unique minimum influencing set exists. Consequently, we provide a set of models that associates the frequency of labels from past events that belong to a subset U with the randomized variable X that corresponds to a label set X L. In order to make a confined history conditionally independent from all other histories given a SMM at any location, we want to allow Memorizing it with regard to these influential labels U for the random variable X. Thus, $P(x \mid h_i) = P\big(x \mid h_i^{\mathrm{U}}\big) = P(x \mid s_{\mathbf{U}})$ where $s_{\mathbf{U}}$ is the SMM state, and for every state x of the X random variable and every location i in an event sequence, which is given in Table 1.

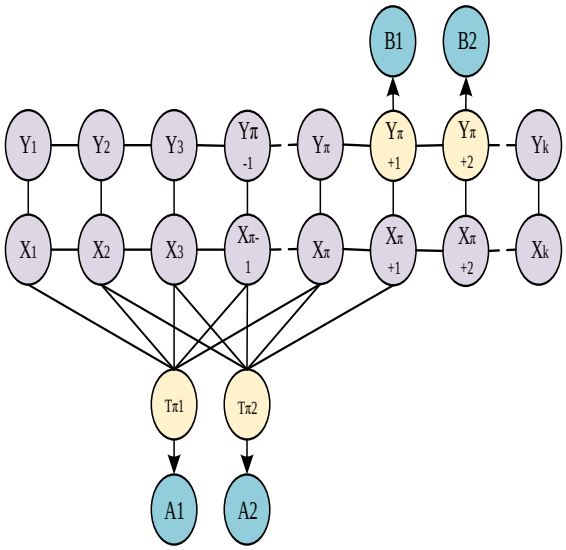

Figure1: Event Analysis using MMsThe time series forecasting technique can ly describe the framework of the variable's sequence without considering other related factors, while the causal prediction technique can only use the causal relationship among a variable along with additional variables.

Neither method can describe the framework of the variable's own time series. A string of variables that are not all known at the outset A set {xN} may be used to represent xN, which is referred to as a random sequence. It is also possible to create a random vector X in a multidimensional random space, with each component denoted by xi. Afterwards, the so-called time sequences are arranged chronologically; that is, the subscript in xi denotes the time variable t, wherein t is an integer representing the growth of the time interval, that is referred to as a random sequence. It is a time series as well. The standard way to express it is with {xt}. Because it is relative to the present instant, the time series' time variable t may take on either a positive or negative integer value. A negative number indicates that its generation occurred before to the present instant, whereas a positive number indicates that its generation occurred subsequent to the current moment. For this reason, in order to put the forward-backward algorithm discussed in this article into practice, one must first train the hidden Markov model's parameters using the definitions provided here. Only then can one determine and foresee the hidden state of the financial time series at different points in time.

Once the hidden state of the series at time 1..., T has been obtained and predicted for time T + 1, the next step is to choose an appropriate technique for forecasting the financial time series. Models based on least squares support vector machines and nonparametric kernel regression continue to form the backbone of this work. Two subsequences, the normal state sequence as well as the abnormal state sequence, were created from the initial financial time series. To get superior prediction results for these two subsequent sequences, it is necessary to pick prediction techniques based on their unique qualities.

Let us assume that the initial network sequence provided consists of a string of observations made at certain times $O = \{(T_i, X_i, Y_i)\}, i = 1,2,\cdots,N$ includes events related to network traffic (where n is the total events). In the initial network traffic sequences, X represents time, Y indicates network events, with nominal, real, and binary values. A regular interval time series data transformation is necessary for identifying the underlying patterns of network activities over time from the time sequence observation series (O). There are three stages to our method: Creating one-hot encoded variables, building time series using pre-defined time scale t_s, and creating forecasting models are the three main tasks. The suggested method was tested using two different time scales (t_s=60 and 120 s) in order to find out how well it worked.

A collection of network traffic series of data in t_s is transformed into time series data by extracting temporal characteristics and translating them to values across time. The first step is to use a pre-established time scale to divide the raw data t_s. The pre-established time scale determines the creation of a new time index t_s over time, $\nabla t_i = (t_s * c) - \nabla t_{i-1}, c = 2,\cdots,t_N, i = 2,\cdots,t_N, \nabla t_1 = t_s$ where $t_N = \frac{m_t}{t_s}$, m_t serves as a cap. This results in the creation of a fresh time index as $\nabla t_i = \{t'_1, t'_2, \cdots, t'_N\}$. Within each time index, ∇t_i comprises a series of tuples $\{(X_i, Y_i)\}$ creating M × J matrix $(X_i \in R^{M\times f})$ and M × D matrix $(Y_i \in R^{M\times D})$, where $M(M \geq 1)$ shows how many observations there were at time ∇t_i, where J is the total amount of variables and D(D≥1) is the size of the dependent variables. The size of M may change over time due to variations in the amount of network events. The number of network events per second for the dependent variables that are one-hot encoded across ∇t_i is computed as $C(Y_i^k) = \Sigma_1^{n_t} I_{Y_t}(\delta_i)$, If every network event is considered normal or an attack, it is indicated by δ_i. In addition, for the nominal variables, we quantify the frequency of every one-hot encoded variable across ∇t_i. For example, the count of utilized port numbers over ∇t_i is taken into consideration for the variables (source along with destination port numbers). In order to provide time-series information with similar intervals for the other factors, an appropriate value is measured for every M-dimensional vector across ∇t_i. We provide a technique in this work that converts values from the initial time sequences into time series elements by use of the wavelet transform with permutation entropy. Using both methods (DWT and PE) have many benefits, such as being able to spot unexpected changes in network events and showing how behaviors of these events have changed over time. Network traffic and other non-stationary data types are good candidates for DWT's

temporal and frequency domain analyses. It does this by repeatedly splitting the data into its component bands. That is, data is passed through highpass and lowpass filters in a preset order to create approximation and detail coefficients. At each level of decomposition, the coefficients reflect the time and frequency data by which the level is informed.

$$w_\varepsilon = \langle d(t), \phi_{(r,r)} \rangle = \int_{-\infty}^{\infty} d(t)\phi_{(r,r)}dt \quad (9)$$

where $d(t)$ shows data, $\phi_{(\tau,\gamma)}(t)$ stands for a mother wavelet function, whereas τ describes the frequency resolution (also known as scale) and γ stands for the shift parameters. Rounded estimate values $(a_{(\tau,\gamma)})$ display data at low frequencies, but the precise coefficients ($d_{(\tau,\gamma)}$) display the metrics that are very relevant to the data. The wavelet coefficients are analyzed using PE. It combines entropy with symbolic patterns to provide a measure of complexity. The coefficients were specifically used $a_{(\tau,\gamma)}, d_{(\tau,\gamma)}$, and $a_{(\tau,\gamma)} + d_{(\tau,\gamma)}$ characteristics to be extracted. The purpose of PE is to build subsequences (s_i) with an embedding dimension (e_d) that has already been established. After that, in order to get the order, every subsequence is transformed into a different permutation $\pi(i) = \{0,1,\cdots,e_d\}$. We can calculate the permutation's probability distribution as $p_{\pi(i)} = \frac{\delta_{\pi(i)}}{|x_i|}$, where $\delta_{\pi(i)}$ displays the pattern $\pi(i)$ as it occurs. In conclusion. To get the permutation entropy, we use Shannon's rule as $\Sigma_f - p_{\pi(f)} \times \log\left(p_{\pi(f)}\right)$.

In addition, statistical feature ($\mathcal{P}$) is removed as $\chi(w_{q_1}, o_i)$, where $\chi(\cdot)$ designates the ANOVA test, w_{q_1} signifies the thorough coefficients of o_i, also o_i symbolizes the i-th vector of the initial sequences O at ∇t_i. If the original sequences with the wavelet coefficients vary statistically in any way, this characteristic will indicate it as a p-value. In addition, we use the initial moment to calculate an extra feature $(\mathcal{E})$ as $\frac{1}{|a_i|}\sum o_i$ in ∇t_i. Method 1 introduces a pseudo-code that normalizes a time series of network data to a specified interval (t_s).

C. Feature selection

When working with a target variable, feature selection involves picking out the discriminative attributes that have the most impact. By eliminating features that do not contribute to the identification of the target variable, the goal of feature selection is to lower the computational cost of the model.

PCA and p-value-based filtering for event sequence prediction

The size of the feature space, which includes things like the number of event kinds or time intervals, may be reduced using principal component analysis in event sequence prediction. This may be particularly helpful when working with intricate event sequences including a wide variety of event kinds or when working with very fine-grained time intervals. When using p-value filtering, features are chosen according to how well they statistically predict the target variable, which is the event sequence in our example. Finding out which traits are most tightly linked to the event sequence while other ones are less important is aided by this. When using p-value filtering, features are chosen according to how well they statistically predict the target variable, which is the event sequence in our example. Finding out which traits are most tightly linked to the event sequence while other ones are less important is aided by this. The p-value of each attribute is found by running a statistical test on it, such as a t-test or a chi-square test. If the characteristic didn't significantly impact the event sequence, then the p-value indicates the likelihood of seeing the given outcome (or an even more extreme result). If the p-value is little, below a certain threshold (e.g., 0.05), then the characteristic is statistically significant along with can be valuable for making predictions. If you want to know what kinds of events or patterns of occurrences are most likely to happen in a given event or sequence, you may use p-value filtering in event sequence prediction. We suggest PCA as a viable option for reducing the temporal dimensionality of duplicated time series and identifying important characteristics. Particularly, PCA reduces the likelihood of overfitting in deep-learning algorithms by eliminating redundant and irrelevant variables and by decreasing correlations across various time steps. Additionally, PCA maps time series data onto principle components that reflect similar characteristics by examining the complete training dataset. This allows it to identify shared patterns. While reducing the length of the original series, this method maintains important statistical details. To reduce computing demands, time series may be preprocessed using PCA prior to being fed into deep-learning models. Feature space dimensionality reduction is possible using PCA. To extract the most useful information from the condensed set of characteristics, p-value filtering may be used. The accuracy of predictions and the interpretability of models may both be enhanced by combining these two factors. With its many benefits, principal component analysis (PCA) is a powerful tool for reducing time series data.

By removing characteristics with low variance and keeping those with large variance, it is a powerful technique for noise reduction. The essential statistical properties of the initial series are also preserved by PCA.

PCA is a powerful method for reducing noise in time series. Using a fresh collection of orthogonal components, PCA successfully removes the noise from the lower variance components while keeping the original historical series' main information. By re-creating the initial time series using the principal components, PCA-inverse transforms make this noise reduction visually apparent.

For this study, we analyzed the input-output connection using p-value and correlation metrics. As an indicator of how likely it is that an observation will be made, we have the p-value. The hypothesis is either accepted or rejected based on this likelihood. The following is the procedure for calculating the p-value using Equation (7):

$$z = \frac{\hat{P} - P0}{\sqrt{\frac{P0(1-P0)}{n}}} \quad (10)$$

Here, the sample size is denoted by n, the presumed population under the null hypothesis is denoted by P0, and the sample percentage is denoted by $\hat{P}$. Then, using the z-value as a starting point, we can calculate the p-value. Since the p-value did not provide the desired outcomes, we resorted to correlation metrics in our feature selection process.

A statistical tool for determining the degree of association between two variables by comparing their relative moments is the correlation coefficient. There is a range of +1 to -1 for its value. A very positive relationship among two variables is shown by a correlation coefficient value of +1, whereas an extremely negative association is represented by a value of -1. A correlation coefficient of 0 indicates that the two variables are unrelated.

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}} \quad (11)$$

In Equation (4), x_i the sample's X variable values, with $\bar{x}$ being the mean of the X variable; the sample's Y variable values, with $\bar{y}$ being the mean of the Y variable.

Algorithm 1: PCA	
a.	Locate the data set's midpoint.
b.	Do the square correlation calculation (c) in $N \times N$.
c.	The eigenvalues and eigenvectors of the covariance matrix can be calculated.
d.	Find the initial element by selecting the matrix's biggest eigenvector c
e.	Find the second part by selecting the eigenvector that is the next biggest in the matrix **c**.
f.	Create a new $N \times M$ matrices in the features the domain, where M is the desired number of elements.

Two key points are to be kept in mind when working with PCA, though: first, the second element of PCA should be orthogonal to the first component. PCA doesn't change the number of samples, but it can change the value of the samples to highlight the differences between datasets. For instance, let's say that a Markov chain describes the existence of certain irregularities in ℓ items coming in at a store, and that the first item in it undergoes an examination of these irregularities to determine its state (when that passes through the queue). By following the queue rule and counting the items as it exits the queue, we have information "with anticipation" on the state of the ℓ-th item. This example shows that observation with the excitement does not always require future occurrence prediction.

4. Results & discussion

We evaluate our learning technique and two proposed MMs in the following trials. Our research is based on actual occurrences and event sequences taken from actual texts, numerous event sequence datasets found in books and musical compositions. We test Python's learning capabilities using fake data in an experiment. In this study, we examine the dynamics of a basic event sequence MMs across five event labels, with the one label of interest having two additional labels as its minimum influencing set. As a function of the number of sequences (K) created, Table 1 illustrates the mean F1 scores comparing the estimated and ground truth influences over several generated sequence datasets. As the tendency continues to rise, it will inevitably converge.

A. Implementation details

All of the necessary libraries were utilized during the implementation in Python 3.9. These included PyTorch 1.11.0 for DGNN model building and training, NumPy 1.21.0 and pandas 1.3.3 for numerical operations and information handling, tqdm 4.62.3 for progress visualization, DyGLib for DGNN executions while link prediction components, and Matplotlib 3.4.3 for result visualization. The server used for all the trials had the following specs: Ubuntu 20.04 OS, 256 GB RAM, 2 TB NVMe SSD, 32 GB VRAM, Intel Xeon Gold 6230 CPU (2.10 GHz), and NVIDIA Tesla V100 GPU.

A 200-unit batch size, two attention heads, a learning rate of 0.0001, ten iterations, a 100-unit time

embedding dimension, a 0.1-unit dropout rate, and a 15% validation/test set ratio were all part of the conventional experimental configuration. To guarantee temporal relevance, historical neighbors were selected using the "recent" technique, whereas negative edge sampling applied a "random" strategy. The settings supplied by DyGLib were used to optimize parameters for each DGNN model, including the total amount of neighbors and layers.

Multivariate time series information with periodic patterns are the target of our model. When identifying periodic events is crucial, like in link forecasting event detection, as well as temporal connection analysis, it works very well. Having said that, the model is subject to the following restrictions and limitations. Input data for the model should have aspects of time, nodes, and edges. It works well with datasets characterized by a high degree of periodicity. Datasets that are neither periodic or irregular could lead to worse performance. The available computing resources determine the model's scalability. By applying an NVIDIA Tesla V100 GPU, we ran our tests on datasets that included up to 33 nodes with 175,360 edges. Memory and processing power may need to be increased for datasets that are larger. Issues with periodic patterns and temporal dynamics are well-suited to the approach. Different methods could work better with static data or graph issues that aren't time-dependent.

B. Dataset

Using the five real-world datasets, we do experiments. At the 0.8 mark on the time line, we divide the train/test set in half, with the segments to the left being utilized for training with the segments to the right for testing. We further divide 10% of the train set into a validation set to prevent overfitting. Starting using a rate of learning of 0.001 and decreasing it by roughly a factor of 10 every 20 iterations, we train the models we create for 100 iterations on a single GPU and a batch size of 1000. There are two sources of this enhancement. To begin, the Markov model is a powerful tool for decomposing time series into their component hidden states; second, the subsequence of the decomposed normal state mitigates the impact of outliers. Second, the model's prediction error is effectively reduced by the process of conversion hybrid prediction model.

B. Datasets

We take into account the following organized datasets, some of which are extracted from event datasets with time stamps that are disregarded (thought to be inaccurate or absent).

In our studies, we use five real-world datasets. Two of them are publicly available ones (DJIA30 as well as WebTraffic) from Kaggle1. The other three are obtained from China Telecom2 (NetFlow), State Grid3 (ClockErr), and Alibaba Cloud4 (AbServe). The statistics for the whole dataset are shown in Table 1.

- **Web Traffic Time Series Forecasting (WebTraffic) -** We find this dataset on Kaggle. It keeps track of the number of views for a particular Wikipedia page and includes around 3 million daily reads. The goal is to use the most current data from the last 12 months to forecast if there will be a significant increase (curve slope > 1.0) in the next 30 days. We find around 2 million false positives and 900,000 positives (rapid increase).
- **DJIA 30 Stock Time Series (DJIA30) -** We find this dataset on Kaggle. Every one of its fifteen thousand daily readings documents four observations made throughout a trading day: three types of trade prices and a trade number. The objective is to use the most up-to-date data from the last year (50 weeks) to forecast five trading days with abnormally volatile prices (variance more than 1.0). Approximately 12,000 typical examples and 3,000 outliers are found by us.
- **Information Networks Supervision (NetFlow) -** China Telecom is the source of this dataset. Each of the approximately 238K measurements documents the incoming and outgoing data from a network equipment on an hourly basis. The gadget will record an alert if it detects an anomalous flow via its ports. Based on information from the last 15 days, our objective is to forecast future anomalies (next day). We find around 200,000 typical examples and 20,000 outliers in total.
- **Abnormal Server Response (AbServe) -** Alibaba Cloud has made this dataset available. The system is comprised of over 12,000 server monitoring series, with each series recording the minute-to-minute readings of various metrics such as CPU, disk, memory, and more. An anomaly will be recorded in the log if a server doesn't reply. Our objective is to use the data from the last hour to forecast outliers within the next five minutes. We find 11.8K typical examples and 0.2K outliers overall.
- **Watt-hour Meter Clock Error (ClockErr) -** The Chinese State Grid was kind enough to provide this dataset. The data is comprised of around 6 million measurements taken weekly, with each reading documenting the watt-hour meters' deviation time and delay. The meter is considered to be malfunctioning if the deviation time is more than 120. We aim to forecast next month's anomalies using data from the previous 12 months. We find around 5 million typical examples and 1 million outliers altogether.

We emphasize that the selected event sequence dataset determines the learned MMs. Every single influencer in these case studies was trained using very little domain-specific data. In order to successfully implement the models in reality, this must be remembered at all times throughout the analysis. We verified the model's prediction accuracy, as illustrated in Figure 2, to assess its reliability. We calculated the model's overall prediction accuracy by averaging the final accuracies after finding the accuracy for each scenario independently. Each case had a sequence randomly eliminated during testing. In order to fit the model, the remaining sequences were used. Once the fitting was finished, the model used the deleted sequence's beginning sequence to forecast the next two steps. The model had a 90% success rate in predicting one of the phases, according to the findings. The model consistently properly predicted the following two steps in every single example. A whopping 96.7% of the time, the model gets the predictions right. Because it drastically reduces the amount of useable data, missing information in time series is a major hurdle to the effective operation of forecasting models. To estimate the missing values of a time series, we provide a paradigm similar to multiple imputation. The foundation of this framework is the construction of ensembles of predictions that are iterative and subsequent, with the goal of filling in the missing data. The algorithm's iterative nature enables the prediction accuracy to be improved over time. Furthermore, the ensemble benefits from the time series' diverse forward and backward dynamics. The created framework is versatile enough to use any underlying model for traditional or machine learning forecasting. Using both linear and nonlinear underlying forecasting algorithms, we successfully evaluated the suggested technique on large data sets.

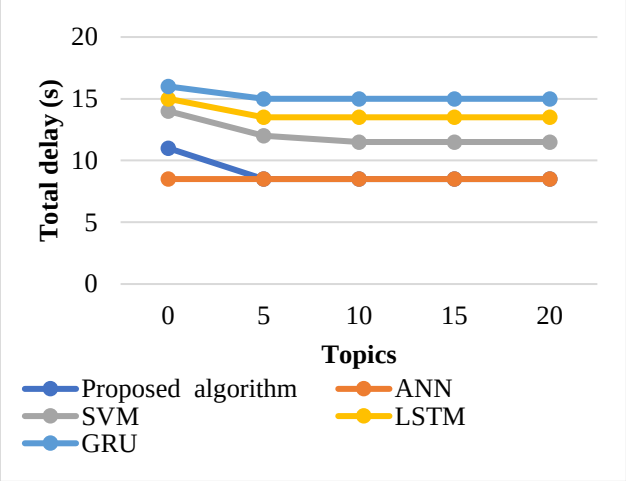

Figure 2: Total delay

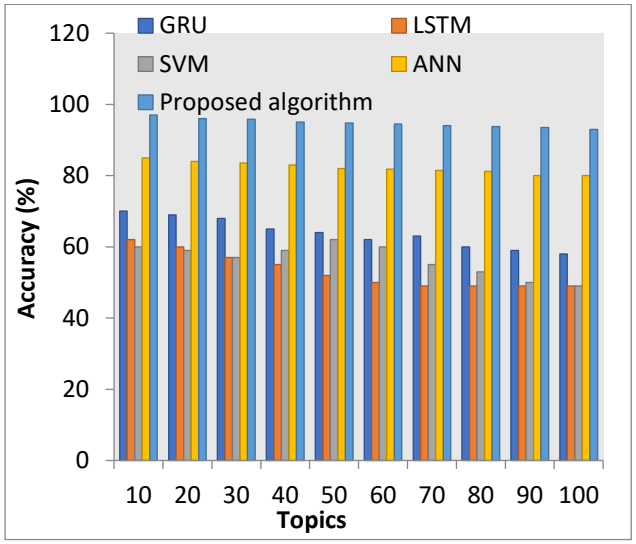

Figure 3: Accuracy analysis

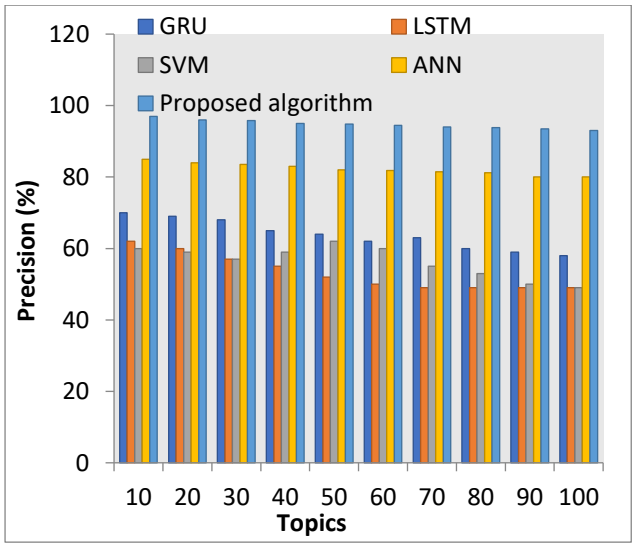

Figure 4: Precision analysis

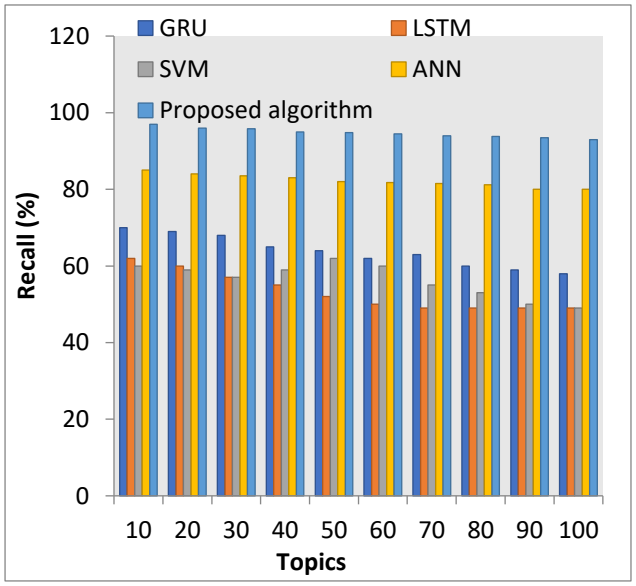

Figure 5: Recall analysis

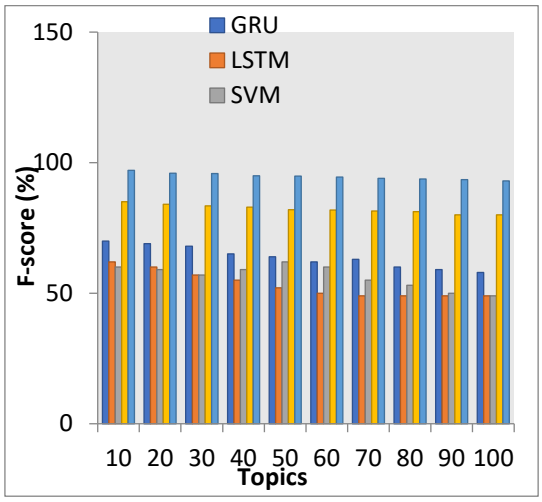

Figure 6: F-score analysis

Table 2: Prediction analysis

Scenario	Overall Prediction Accuracy
Topic 1	91.25%
Topic 2	92.15%
Topic 3	93.14%
Topic 4	94.14%
Topic 5	91.25%
Topic 6	93.14%
Topic 7	94.75%
Topic 8	94.15%

Fig 4, 5, and 6 shows the precision, recall and f-score for precision, recall and f-score, respectively. Table 2 illustrates the overall prediction accuracy for different topics using markov models.

Ablation Study

Here, we establish ablation experiments to examine how various event representations impact the outcomes of the predictions. Eliminating arguments and relations from the event graph was part of this process. Experiments with relation-level ablation preserved just the arguments while removing the edges of concurrent event relations from the graph. While keeping the graph structure and other characteristics, the argument-level ablation tests eliminated the events' spatial and temporal dimensions. You can see the outcomes of the ablation tests in Table 3.

Table 3: Comparisons with standards and ablation trials

Model	Accurate Matching	Recall	Precision	F1-Score
[19]	87.13	91.97	86.87	89.03
[22]	88.76	87.78	89.06	87.07
[20]	89.79	88.85	84.96	88.12
[23]	83.94	86.85	89.08	86.97
[24]	87.15	89.97	89.82	86.05
[25]	84.84	86.32	86.88	85.88
Proposed model	96.86	98	94	92.56

Various ablation tests showed that the model's performance dropped somewhat, as can be observed in the results of the experiment. Recall values increased more noticeably in the relation-level ablation trials, suggesting that although concurrent event inclusion did enhance accuracy in forecasting, it was also a major contributor to the model's more cautious forecasts. Reducing metrics in the argument-level ablation trials demonstrated that geographical and temporal information improved performance on the event prediction test. To test the efficacy of the event prediction model we presented in this paper—one that relies on event knowledge graphs—we built an event knowledge graph by hand in the transportation sector. We investigated the potential of a model-based graph attention neural network to validate the utility of a heterogeneous graph transformer for occurrence scenario characterisation, with the aim of better integrating the data contained in various arguments and concurrent events. The significance of spatial and temporal aspects in event representation was also confirmed by our tests.

More prediction approaches will be investigated and the possibility of building more complicated event linkages will be considered in future investigations. There are several sorts of entities represented by event arguments. To further enhance the forecast, it may be helpful to consider the impact of various entity kinds on the model. We will think about adding multimodal data to the event knowledge graph enabling real-time monitoring of unexpected occurrences and quick decision-making in the future. We are thinking about adding IoT to get additional data for decision-making, which will make the model better at making decisions.

5 Conclusion

For dynamics in a sequential event dataset, we have suggested summary Markov models. Two of these models use different summary mappings to determine which event labels are influential. We demonstrate the robustness of our models compared to previous techniques via experiments on structured datasets and event sequences derived from text. The suggested model's key benefit is that they find influencing sets and achieve prediction performance that is on par with baselines. By combining parameter sharing concepts from variable order Markov models with other methods for summary mapping, including counting-based models, and by adjusting the size of the summary range for expressive models, the scope of summary Markov models might be enlarged. An obstacle for future research is the management of noisy event sequence datasets including many event labels that are not meaningful. This work presents a one-step approach for predicting the subsequent hidden state of a time sequence when the sequence comprises two kinds of hidden states, based on the premise that the hidden Markov model's hidden state is a first-order Markov process. This research suggests using several models to forecast the concealed state's value based on varied amounts of time series sample information after obtaining the prediction result.

Acknowledgement

Ministry of Education Production and education cooperation project : The reform of the course "Probability theory and mathematical statistics" under the background of new engineering.
(230719114307227)

References

[1] Işık, Y.E., & Aydın, Z. (2023). Comparative analysis of machine learning approaches for predicting respiratory virus infection and symptom severity. PeerJ, 11. DOI:10.7717/peerj.15552

[2] Aalikhani, R., Fathian, M., & Reza Rasouli, M. (2024). Comparative Analysis of Classification-Based and Regression-Based Predictive Process Monitoring Models for Accurate and Time-Efficient Remaining Time Prediction. IEEE Access, 12, 67063-67093. DOI: 10.1109/CONIT51480.2021.9498451

[3] Qi, D.L., & Bure, V.M. (2024). Explanatory comparative analysis of time series forecasting algorithms for air quality prediction. Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes. https://doi.org/10.1007/s42979-020-00180-5

[4] Wahyuni, S.N., Sediono, E., Sembiring, I., & Nahar Khanom, N. (2022). Comparative analysis of time series prediction model for forecasting COVID-19 trend. Indonesian Journal of Electrical Engineering and Computer Science. http://doi.org/10.11591/ijeecs.v28.i1.pp600-610

[5] Widiputra, H., & Juwono, E. (2024). Parallel multivariate deep learning models for time-series prediction: A comparative analysis in Asian stock markets. IAES International Journal of Artificial Intelligence (IJ-AI). DOI: 10.11591/ijai.v13.i1.pp475-486

[6] Benevento, E., Aloini, D., & Squicciarini, N. (2021). Towards a real-time prediction of waiting times in emergency departments: A comparative analysis of machine learning techniques. International Journal of Forecasting.https://doi.org/10.1016/j.ijforecast.2021.10.006

[7] Pande, K., Divayana, D.G., & Indrawan, G. (2021). Comparative analysis of naïve bayes and knn on prediction of forex price movements for gbp/usd currency at time frame daily. Journal of Physics: Conference Series, 1810. doi:10.1088/1742-6596/1810/1/012012

[8] Pedraza, N.H., Villegas, C.M., Aqueveque, D.C., & Das, R. (2024). A Comparative Analysis of Time Series Data Augmentation Methods in the Identification of Diabetic Neuropathies Based on Deep Learning Algorithms. 2024 43rd International Conference of the Chilean Computer Science Society (SCCC), 1-8. DOI:10.1109/SCCC63879.2024.10767645.

[9] Boddu, M.S., Reddy, M.K., Sahitya, G.L., Afrin, S., & Sudha, K. (2024). Exploring Predictive Models for Airfare Forecasting: A Comparative Analysis of Time Series and Machine Learning Approaches. International Research Journal of Modernization in Engineering Technology and Science. DOI: 10.1109/ACCESS.2020.2980942.

[10] Anand, S., Sandhu, S.K., Biswas, B., & Bala, R. (2024). Comparative analysis of different Karnal bunt disease prediction models developed by machine learning techniques for Punjab conditions. International journal of biometeorology. DOI: 10.1007/s00484-024-02707-4

[11] Rosita, Y.D., & Moonlight, L.S. (2024). Comparative Analysis of LSTM Neural Network and SVM for USD Exchange Rate Prediction: A Study on Different Training Data Scenarios. Scientific Journal of Informatics. https://doi.org/10.15294/sji.v11i1.49975

[12] Riyadi, S., & Fahmi, F. (2024). Comparative Analysis of Cryptocurrency Prediction based on Deep Learning, Decision Tree, Gradient Boosted Tree, Random Tree, and k-NN Model. International Journal of Advances in Data and Information Systems. DOI:10.59395/ijadis.v5i2.1338

[13] Aryal, S., Nadarajah, D., Rupasinghe, P.L., Jayawardena, C., & Kasthurirathna, D. (2020). Comparative Analysis of Deep Learning Models for Multi-Step Prediction of Financial Time Series. Journal of Computer Science. DOI: 10.3844/jcssp.2020.1401.1416

[14] Othman, S.A., Jameel, H.H., & Abdulazeez, S.T. (2024). Comparative Analysis of Univariate SARIMA and Multivariate VAR Models for Time Series Forecasting: A Case Study of Climate

Variables in Ninahvah City, Iraq. Mathematical Problems of Computer Science. https://doi.org/10.51408/1963-0113

[15] Sun, H. (2024). Forecasting Washington State's Housing Market: A Comparative Analysis of Time Series Models and Economic Indicators. Emirati Journal of Business, Economics, & Social Studies. Doi:10.54878/zg7crb70

[16] Buragadda, S., Rao, G.M., Lavanya, M., & Gopi, A. (2024). Development and Impact Analysis of a Multi-Pandemic Real-Time Data Dashboard: A Comparative Analytical Approach. Journal of Electrical Systems. https://doi.org/10.52783/jes.3377

[17] Tripathy, N., Mishra, D., Hota, S., Priyadarshani Behera, M., Chandra Das, G., Sekhar Dalai, S., & Nayak, S.K. (2025). A comparative analysis of exponential smoothing method and deep learning models for bitcoin price prediction. IAES International Journal of Artificial Intelligence (IJ-AI). DOI:10.11591/ijai.v14.i2.pp1401-1409

[18] Li, X., Li, K., Shen, S., & Tian, Y. (2023). Exploring Time Series Models for Wind Speed Forecasting: A Comparative Analysis. Energies. https://doi.org/10.3390/en16237785

[19] Lekshmy, H.O., Panickar, D., & Harikumar, S. (2022). Comparative analysis of multiple machine learning algorithms for epileptic seizure prediction. Journal of Physics: Conference Series, 2161. DOI 10.1088/1742-6596/2161/1/012055

[20] Wang, Q., Wu, L., Li, C., Chang, X., & Zhang, B. (2024). Research on a Real-Time Prediction Method of Hull Girder Loads Based on Different Recurrent Neural Network Models. Journal of Marine Science and Engineering. https://doi.org/10.3390/jmse12050746

[21] Taslim, D.G., & Murwantara, I.M. (2024). Comparative analysis of ARIMA and LSTM for predicting fluctuating time series data. Bulletin of Electrical Engineering and Informatics. https://doi.org/10.11591/eei.v13i3.6034

[22] Mishra, S. (2022). A Comparative Study for Time-to-Event Analysis and Survival Prediction for Heart Failure Condition using Machine Learning Techniques. Journal of Electronics, Electromedical Engineering, and Medical Informatics. https://doi.org/10.35882/jeeemi.v4i3.225

[23] Sahoo, D., Sood, N., Rani, U., Abraham, G., Dutt, V., & A.D., D. (2020). Comparative Analysis of Multi-Step Time-Series Forecasting for Network Load Dataset. 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 1-7. DOI:10.1109/ICCCNT49239.2020.9225449

[24] Atique, S., Noureen, S., Roy, V., Bayne, S.B., & Macfie, J. (2020). Time series forecasting of total daily solar energy generation: A comparative analysis between ARIMA and machine learning techniques. 2020 IEEE Green Technologies Conference(GreenTech), 175-180. DOI:10.1109/GreenTech46478.2020.9289796

[25] Zhou, H., Li, W., Jiang, Z., Cai, F., & Xue, Y. (2022). Flight Departure Time Prediction Based on Deep Learning. Aerospace. https://doi.org/10.3390/aerospace9070394.