Journal of Information Systems Engineering and Management
2025, 10(2)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Enhancing Student Performance Management through Machine Learning and Deep Learning Models

1M. Kannan*,2Dr. K. R. Ananthapadmanaban
1Department of Computer Science, College of Science and Humanities,
SRM Institute of Science and Technology,Vadapalani Campus, Chennai, Tamilnadu
Corresponding Author km1200@srmist.edu.in
2Department of Computer Science, College of Science and Humanities,
SRM Institute of Science and Technology, Vadapalani Campus, Chennai, Tamilnadu
toananths@gmail.com

ARTICLE INFO

Received: 15 Nov 2024
Revised: 28 Dec 2024
Accepted: 14 Jan 2025

ABSTRACT

This work addresses the challenge of predicting student performance by investigating the sparsity of student action data and the imbalance between performance and completion rates. Initially, various machine learning (ML) and deep learning (DL) methods were compared using the Unitelma dataset, highlighting the superior performance of DL algorithms such as CNN and LSTM. Subsequently, an attention-based model combining convolutional and recursive layers (with LSTM units) was implemented and trained, although insufficient training data limited accurate model assessment. Lastly, the study focused on LSTM, TCN, and Kalman Filter models using the XuetangX dataset, leveraging oversampling (ADASYN) and data densification (PCA) techniques to address class imbalance and data sparsity issues. The Kalman Filter model demonstrated superior performance in terms of AUCPR, while LSTM and TCN models outperformed it in binary classification. TCN, in particular, showed increased efficiency over LSTM, especially for longer time sequences. Future work will involve applying these techniques to academic datasets, potentially retraining models with ADASYN and PCA algorithms to further improve performance. Additionally, emphasis will be placed on exploring the efficacy of TCN models in solving student performance prediction tasks.

Keywords: Student performance, Attention-based model, Kalman Filter, Convolutional and recursive layers.

INTRODUCTION

Student performance prediction is a pressing issue in education, with implications for intervention strategies and student success. Traditional statistical methods and newer ML and DL techniques offer promising avenues for addressing this challenge. However, the predictive accuracy of these models is influenced by factors such as the sparsity of student action data.

Various machine learning (ML) and deep learning (DL) methods, including CNN and LSTM networks, have been applied to predict student performance. These models leverage sequential student action data to make predictions about future performance behavior. However, challenges related to the sparsity of student action data, where students may have limited engagement with course materials, and imbalances across demographic, academic, and socioeconomic factors can bias predictive models assessing student performance. Varied representation in age, gender, ethnicity, and socioeconomic status, along with disparities in academic metrics, extracurricular activities, and course characteristics, can skew model perceptions.

Existing research has shown that DL algorithms, particularly CNN and LSTM, demonstrate superior performance in student performance prediction tasks compared to traditional statistical methods. However, these models may struggle with the sparsity of student action data and the imbalances across demographic, academic, and socioeconomic factors can bias predictive models assessing student performance. Varied representation in age, gender, ethnicity, and socioeconomic status, along with disparities in academic metrics, extracurricular activities, and course characteristics, can skew model perceptions, which can limit their predictive accuracy.

The objective of this work is to investigate the effectiveness of ML and DL methods in predicting student performance, considering the challenges posed by data sparsity and class imbalance. By comparing the performance of various methods using the Unitelma dataset and implementing an attention-based model, the study aims to provide insights into the most effective modeling approaches for addressing data sparsity and class imbalance in educational datasets. Furthermore, by evaluating the performance of LSTM, TCN, and the Kalman Filter model using the XuetangX dataset, the study aims to assess the efficacy of TCN models in addressing the challenges of student performance prediction, particularly in the context of longer time sequences and sparse data.

LITERATURE REVIEW

They were superior in that they could deal with high-dimensional data and were computationally cheap, relatively. Support vector machines (SVMs), however, concentrated on selecting the best hyperplane that maximally distinguished various classes of students. [25] [26] [27] It gave excellent generalization performance, even with the use of high-dimensional data. Lastly, non-parametric KNN classified the students according to their closeness to other students in the feature space. [25] Though these classical ML algorithms provided great insights and sufficient accuracy, they were usually unsatisfactory with larger and more complicated datasets in terms of tackling the complex relationship within them. [28] [29] The shortcomings of these less complicated models became all the more pronounced as the complexity and amount of educational data expanded.

Deep learning (DL) models, with their inbuilt ability to learn complex features automatically from raw data, have proven to be powerful tools for predicting student performance. [30] Recurrent neural networks (RNNs), specifically Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), are particularly well-suited to model sequential data, as they can capture the temporal dynamics of the student learning process. Convolutional neural networks (CNNs) have also been used successfully to learn spatial features from representations of learning interactions. The advent of CNNs was a major change in the strategy for predicting student performance, away from the need for manually designed features toward the automatic learning of rich representations directly from raw data. This method had the advantage of greater precision and less dependence on domain knowledge in feature creation.

RNNs, particularly LSTMs and GRUs, have been found quite effective in capturing the inherently sequential nature of learner interactions with educational platforms.[32] [33] LSTMs are particularly prominent with their capacity for addressing the vanishing gradient issue, a generic problem with simple RNNs that prevented long-range dependencies from being learned. This enables LSTMs to capture the long-range patterns and trends in the learning behaviors of students more effectively, offering a better picture of the learning process. Modeling the long-range dependencies is important to predict future performance accurately because it reflects the total impact of learning experiences in the past. [34] GRUs, an optimized version of LSTMs, provide similar performance with enhanced computational efficiency. [35] They are thus ideal for use with big data or real-time prediction. LSTMs and GRUs can naturally combine heterogeneous sources of data, such as assignment submissions, quiz grades, and online usage logs, to produce more precise and subtle predictions of future student performance. The application of RNNs is a major leap in the technology that enables dynamic and intricate learning processes to be modeled.

Current research is actively investigating hybrid deep learning models that combine the best of RNNs and CNNs to model both temporal and spatial dimensions of student learning. [34] [35] Hybrid models use the temporal modeling strength of RNNs to model the development of student knowledge over time and the spatial feature extraction strength of CNNs to model patterns in student interactions. [36] The synergistic use of these approaches offers a better and more subtle representation of student learning, thus making for more precise predictions of performance. RNNs and CNNs used together provide for a more enhanced and holistic description of student learning processes beyond what can be gained from using one or the other alone. This is an improvement in the science, as it offers a stronger and more versatile tool for student performance prediction.

3. DATA COLLECTION

Data collection for educational data mining (EDM) projects involves gathering various types of data from sources such as student records, course materials, online learning platforms, and questionnaires. This data can include numerical, categorical, and text data, reflecting factors (Table 1).

Table 1. Data Parameters

Data Parameter	
1. Age	2. Subject Area
3. Gender	4. Level of Difficulty
5. Ethnicity	6. Delivery Format
7. Socioeconomic Status	8. Online Provider

9. Grades from Previous Semesters	10. Enrollment
11. Standardized Test Scores	12. Relevant Start and End Dates
13. GPA	14. Extracurricular Activities
15. Attendance Records	16. Co-curricular Activities
17. Tardiness	18. Level of Involvement
19. Early Dismissals	20. Seminar and Assignment
21. Course Subject	22. NPTEL Course Participation
23. Course Level	24. Rural/Urban Classification
25. Teacher	26. Family Income
27. Type of Online Course	28. Caste
29. Grades	

3. EXPERIMENTS

In the initial experiment of the work project, the goal was to establish baselines for predicting student performance using ML models [25]. The process began with data preprocessing from the Moodle database (D2.1 and D2.2), which records all student actions (e.g., login, logout, view, comments, exercises) along with the timestamps.

First, student actions were mapped into a dictionary `d`, where each key is a student ID and the corresponding value is a chronologically ordered list of actions. Then, a join operation was performed between the Moodle dataset (D2.0) and the ESSE3 dataset (D1.0) to link each student in `d` with their ESSE3 registration ID. This mapping was stored in a dictionary, using ESSE3 IDs as keys and Moodle IDs as values, allowing the combination of personal data from ESSE3 with academic data from Moodle.

Next, to determine whether students in `d` had higher grade or lower grade, datasets D1.1 is utilized. D1.1 was checked first to identify students who had either higher grade or lower grade on their studies.

The experiment involved two approaches. The first approach created vector sequences representing student actions during their first year, aggregated by day. The second approach quantified each type of action performed by the student each day, within the first academic year. The dataset contained 101 different types of actions.

For the first approach, the data were processed to generate a matrix M. Each instance in M was a vector representing the actions of a student over the first year, with each vector having a length of 360, corresponding to the number of actions performed each day.

This matrix M was then used to train the designated ML algorithms, which are detailed in the following equations 1-2.

$M = [s_1, s_2, s_3, s_4, \ldots, s_k]$ with k = number of students in the dictionary d (1)

$s_i = [t_1, t, t_3, t_4, \ldots, t_N]$ is the the vector of the activities carried out by the student (2)

i, with N = 360 (days of the year), and t_i = sum of the actions carried out by the student on day i.

In the second approach, the training data is structured as a tensor T with dimensions k times N times W, where k is the number of students in the dictionary d, N is the time interval of 360 days, and W is the number of action types in the Unitelma dataset.

Specifically, the tensor $T = [s_1, s_2, s_3, s_4, \ldots, s_k]$ represents the actions of (k) students. Each student (s_i^t) has a matrix of actions over time: $s_i^t = [a_1, a, a_3, a_4, \ldots, a_W]$ is the vector of actions for student i on day t (for ($t = 1, 2, \ldots, N$)). Each element (a_j) within (s_i^t) denotes the number of times action j was performed by the student on that day.

The vector y contains the ground truth values used to evaluate model performance. It represents the career status of each student with binary values: 0 indicates a student who has higher grade, while 1 indicates a student who has lower garde.

Formally, $y = [c_1, c_2, c_3, c_4, \dots, c_k]$, where k is the number of students in dictionary d:

$$c_i = \begin{cases} 0 \text{ if the student has higher grade.} \\ 1 \text{ if the student has lower grade.} \end{cases} \qquad (3)$$

This y vector is used for both approaches.

3.1 Algorithms used

Using the M matrix and y vector, the data is split into 70% for training and 30% for testing (Fig 1). The training data is then used to train various ML and DL algorithms, including Linear Regression (LiR), Logistic Regression (LoR), Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), Naive Bayes (NB), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM).

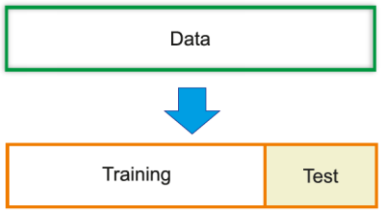


Figure 1. Train Set/Test Set Splitting

4.1.1.1 Linear Regression

LiR is a supervised learning method that models the relationship between variables by fitting a linear equation to observed data. It estimates the expected value using (k) independent variables $(x_1, x_2, x_3, x_4, \dots, x_k)$) to predict the dependent variable y.

The model learns ϑ parameters, which are estimated based on instances of the training set, to formulate the hypothesis $h_\theta(x)$ as close as possible to the dependent variable y in the learning examples of the training set, with $h_\theta(x) = \vartheta_0 + \vartheta_1 x_1 + \vartheta_2 x_2 + \cdots . + \vartheta_k x_k$.

4.1.1.2 Logistic Regression

LoR, despite its name, is a classification model used for binary classification tasks. It is effective for linear classification, where the decision boundary can perfectly separate the classes. Unlike LiR, LoR predicts a value between 0 and 1 using the hypothesis $h_\theta(x)$ (3), which applies a sigmoid (logistic) function $\sigma(z) = \frac{1}{1+e^{-z}}$ to a linear combination of the input variables.

$$h_\theta(x) = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + + \cdots \theta_k x_k) = \sigma(\theta^T x) = \frac{1}{1+e^{-\theta T_x}} \text{ with } 0 \le h_\theta(x) \le 1 \qquad (4)$$

3.1.1 Decision Tree

The DT is easy to interpret, handles both numeric and categorical data, and can learn non-linear relationships. The algorithm builds a tree by starting at the root and splitting the data based on the feature that maximizes information gain, calculated as the difference between the parent node's impurity and the sum of the child nodes' impurities. In this experiment, entropy was used as the splitting criterion due to its higher efficiency compared to the Gini index.

3.1.2 Random Forest

RF is an ensemble classifier that uses bagging with DTs to minimize overfitting. The algorithm generates multiple trees from the same training set, but each tree uses a random subset of features for splitting, ensuring diversity among the trees. RFs provide good classification performance, scalability, and ease of use by combining multiple weak learners to create a robust model with better generalization and less susceptibility to overfitting. For this experiment, the algorithm was trained with 100 trees to balance performance and computation cost.

3.1.3 Support Vector Machine

SVM is a supervised learning model used for classification and regression. The optimization goal of SVM is to maximize the margin, defined as the distance between the decision boundary (hyperplane) and the nearest training samples, known as support vectors (Fig 2).

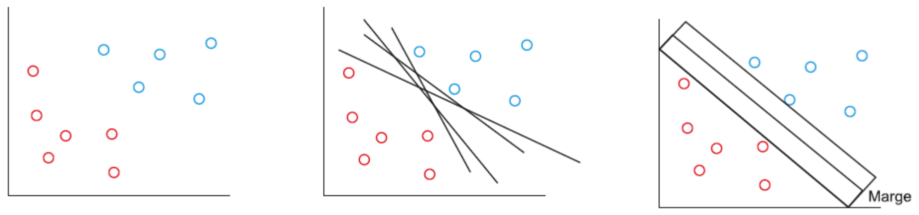


Figure 2. Representation of SVM margin

Large margins in decision boundaries are preferred because they typically result in lower generalization error, while smaller margins can lead to overfitting. In this experiment, the RBF (Radial Basis Function) kernel was employed as it was found to be more effective compared to linear, polynomial, and Gaussian kernels.

3.1.4 Naive Bayes

NB is a classification algorithm based on Bayes' theorem, applying a probabilistic approach to solve classification problems. It assumes independence between features given the class variable ☐. Formally, Bayes' theorem states the relationship between the class variable ☐ and the dependent feature vector x_1 through x_n as follows:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} = \frac{P(x_1,x_2,x_3,x_4,\dots,x_n)P(y)}{P(x_1,x_2,x_3,x_4,\dots,x_n)} = \frac{P(y)\prod_{i=1}^{n}P(x_i|y)}{P(x_1,x_2,x_3,x_4,\dots,x_n)} \quad (5)$$

As $P(x_1, x_2, x_3, x_4, \dots, x_n)$ is constant for the input, the model calculates predictions using the following formula:

$$\hat{y} = \arg\max_{y} P(y)\prod_{i=1}^{n} P(x_i|y) \quad (6)$$

Compared to Multinomial Naive Nayes, the use of a Gaussian NB was found to be more effective, because the latter fits better to continuous valued features as conforming to the Gaussian distribution as the data processed for the prediction of the performance.

3.1.5 Convolutional Neural Network

CNNs are widely used in computer vision and spatial data analysis. They transform complex inputs into simpler features through a series of steps, efficiently capturing spatial and temporal dependencies in images. CNN architectures excel with image data due to reduced parameters and reusable weights. They employ shared-weight neural networks and convolution operations to segment input into 2D feature maps. This experiment's CNN includes convolutional or Separable Convolutional layers (CLs), pooling layers (PLs), and fully connected layers (FCLs). The convolution operation convolves input domains with filter masks to generate output feature maps.

The output from the CL can be expressed with the following expression:

$$o(x,y) = \sum_{j=\frac{-(H-1)}{2}}^{\frac{(H-1)}{2}} \sum_{j=\frac{-(W-1)}{2}}^{\frac{(W-1)}{2}} i(x+i, y+j) * w(i+\frac{W+1}{2}, j+\frac{H+1}{2} \quad (7)$$

Where: $o(x, y)$ is the output from the CL (output obtained from running a kernel selection and convolution operation; $i(x+i, y+j)$ is a generic input $(x+i, y+j)$ of the CL; $\frac{W+1}{2}, \frac{H+1}{2}$ is a specific weight of the filter matrix defined in a horizontal range [1, W] and a vertical range [1, H];

• W and H are the width and height of both the 2D map and the filter mask.

Separable convolution focuses on the spatial dimensions of an image and kernel (width and height), dividing a kernel into two smaller ones. This approach speeds up network execution by reducing parameters and computational complexity compared to classic convolution. An example of separable convolution application is shown below.

The PL reduces data size by selecting small, usually non-overlapping, pooling masks within the image. It performs either max-pooling, selecting the maximum value in the mask, or average-pooling, averaging the values in the mask.

FCLs have neurons with connections to all activations in the previous layer, computed through matrix multiplication and a bias offset. This is distinct from sparsely connected layers, as shown in the image below (Fig 3).

The neural network chosen for this experiment follows the LeNet-5 architecture. This architecture is straightforward and well-suited for handwritten and machine-printed character recognition tasks. The LeNet-5 architecture is depicted in the following image.

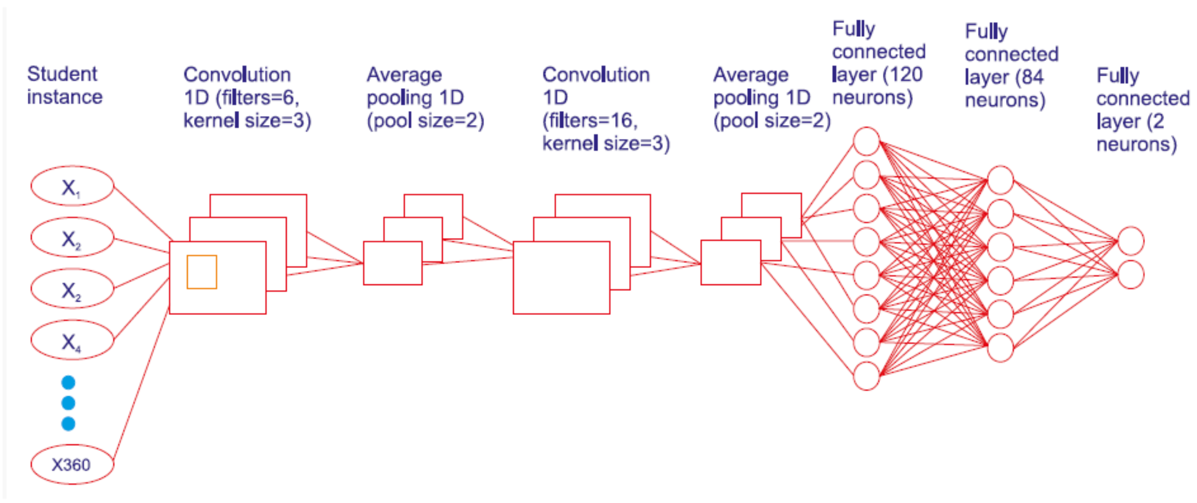

Student instance
Convolution 1D (filters=6, kernel size=3)
Average pooling 1D (pool size=2)
Convolution 1D (filters=16, kernel size=3)
Average pooling 1D (pool size=2)
Fully connected layer (120 neurons)
Fully connected layer (84 neurons)
Fully connected layer (2 neurons)
X1
X2
X2
X4
X360

Figure 3. Architecture of LeNet-5

The LeNet-5 architecture comprises two sets of convolutional and average PLs, followed by two FCLs and a binary classifier (BC). The first two CLs and FCLs use the Rectified Linear Unit (ReLU) activation function for faster training, testing times, and improved predictive performance. ReLU is defined as $\mathrm{ReLU}(x) = \max(0, x)$. The final layer, serving as the classification layer, employs the softmax function to generate the output vector.

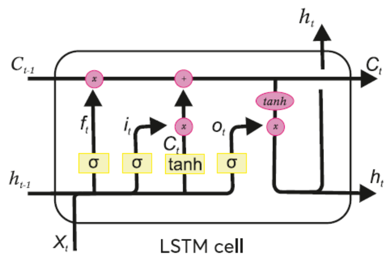
$$\mathrm{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (6)$$

The loss function used to train LeNet-5 is softmax cross entropy. It calculates the sum of cross entropies for each instance in the batch by comparing predicted and actual classes. This sum is then divided by the batch size and negated. In mathematical terms, if □x represents each instance in the batch, p(x)is the ground-truth distribution, and q(x) is the estimated distribution by the model, the loss function is defined as::

$$H(p.q) = -\frac{1}{|\mathrm{batch}|}\sum_{x\in \mathrm{batch}} p(x)\log(q(x)) \qquad (8)$$

3.1.6 Long-Short Term Memory

Recurrent networks, unlike other types, process inputs sequentially, considering their order and retaining memory of previous elements. They can propagate data forwards and backwards, effectively extracting and understanding elements within context. In this experiment, a recurrent network based on LSTM cells was implemented, known for their ability to remember and retain information from past inputs for extended periods (Fig 4).



$$i_t = \sigma(x_t U^i + h_{t-1} W^i)$$
$$f_t = \sigma(x_t U^f + h_{t-1} W^f)$$
$$O_t = \sigma(x_t U^o + h_{t-1} W^o)$$
$$C_t = \tanh(x_t U^g + h_{t-1} W^g)$$
$$C_t = \sigma(f_t * C_{t-1} + i_t * Ct)$$
$$h_t = \tanh(C_t) * O_t$$

Figure 4. Detailed representation of an LSTM unit

The architecture selected incorporates LSTM cells, a Flatten layer (FL), and a BC. LSTM cells, comprising 100 units, analyze input vectors derived from student instances. Post-LSTM processing, the FL compresses output dimensions into a 1-dimensional vector. Subsequently, a BC employs softmax activation to generate normalized output values. This architecture adeptly handles sequential data, facilitating the production of sequenced output (Fig 5).

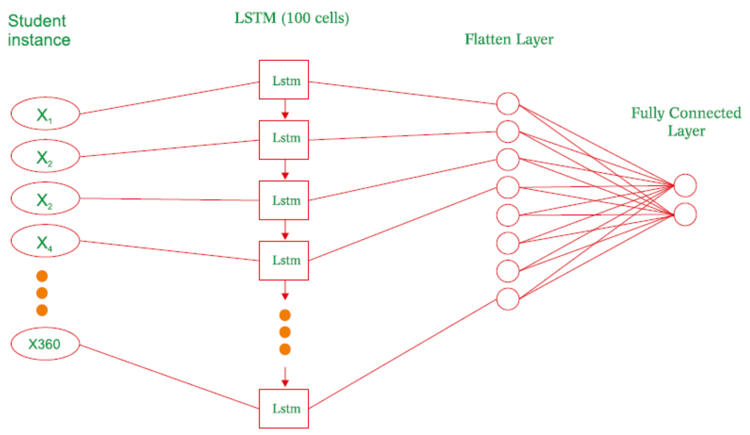


Figure 5. Architecture of the LSTM neural network

The chosen architecture for the recurrent network includes LSTM cells (100 cells) processing input vectors of student instances, followed by a FL to reduce output dimensions to a 1D vector. A BC uses the softmax activation function to return normalized output values. Similar to the CNN, the loss function for updating LSTM network weights during training is softmax cross entropy.

3.1.7 Choice of metrics

The metrics used to assess the performance of the models include precision, recall, F1 score, AUCPR, and Cohen's Kappa Score.

3.2. Achieved results

3.2.1 First approach

This section shows the training results of the algorithms mentioned in the previous paragraphs. The following performance are achieved by testing the trained models on the test set (table 2).

Table 2. Performance comparison of the ML and DL algorithms in terms of precision, recall, F1, AUCPR and Cohen's Kappa score

Algorithms	Precision	Recall	F1	AUCPR	Cohen's Kappa
LiR	0.637	0.621	0.573	0.559	0.248
LoR	0.636	0.626	0.631	0.565	0.257
DT	0.592	0.592	0.592	0.538	0.184
RF	0.654	0.648	0.651	**0.666**	0.292
SVM	0.649	0.567	0.605	0.606	0.114
NB	0.639	0.596	0.616	0.560	0.203
CNN	0.640	0.637	0.638	0.573	0.275
CNN with Separable Convolution	0.625	0.623	0.624	0.565	0.243
LSTM	**0.663**	**0.660**	**0.661**	0.591	**0.322**

The Precision-Recall graphs illustrate the tradeoff between precision and recall for various thresholds. Summary plots showcase the performance results achieved by the DL algorithms at the end of each training epoch (Fig 6-8).

Journal of Information Systems Engineering and Management
2025, 10(2)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

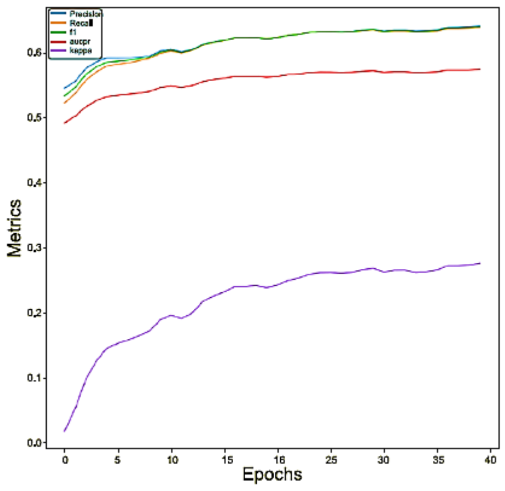


Figure 6. CNNresults

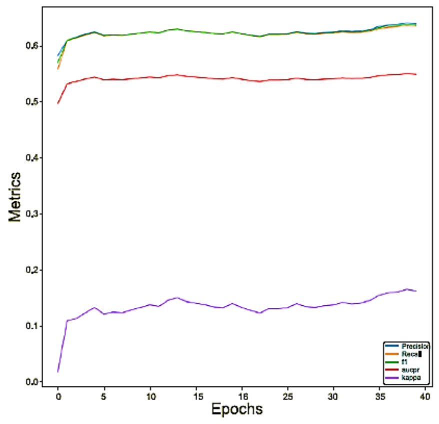


Figure 7. CNNwith Separable Convolution results

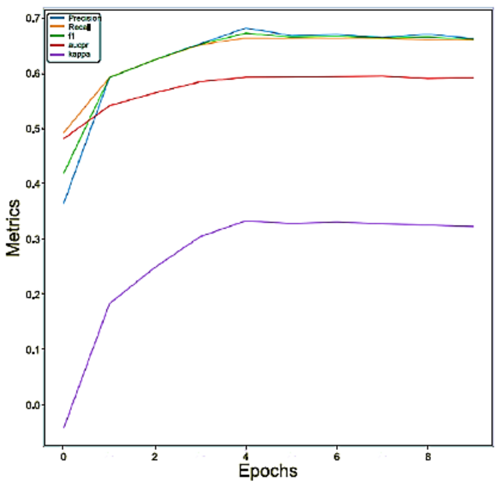


Figure 8. LSTM results

The LSTM model emerged as the most effective for this task, demonstrating higher reliability in Precision, Recall, F1 score, and Cohen's Kappa Score.

3.2.2 Second approach

This paragraph presents the results, Precision-Recall graphs, and learning outcomes of DL algorithms, obtained by executing the previously trained models on the test set (table 3 and Fig 9-23).

Table 3. Performance Comparison: ML vs. DL Algorithms

	Precision	**Recall**	**F1**	**AUCPR**	**Cohen's Kappa**
LiR	0.513	0.512	0.512	0.460	0.025
LoR	0.615	0.612	0.614	0.555	0.226
DT	0.641	0.641	0.641	0.579	0.280
RF	0.694	0.688	**0.691**	0.731	0.372
SVM	0.674	0.674	0.674	**0.745**	0.347
NB	0.642	0.625	0.634	0.565	0.257
CNN	0.691	**0.690**	**0.691**	0.622	**0.378**
CNNwith Separable Convolution	**0.700**	0.671	0.685	0.617	0.334
LSTM	0.597	0.598	0.597	0.544	0.193

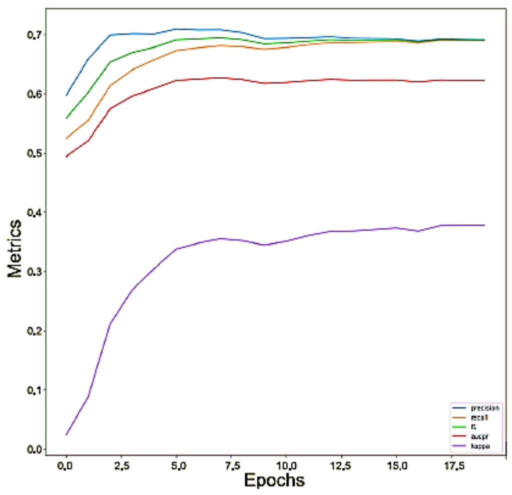


Figure 9. CNNresults

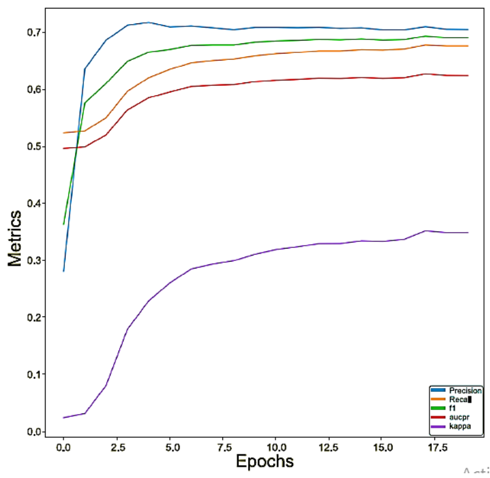


Figure 10. CNNwith Separable Convolution results

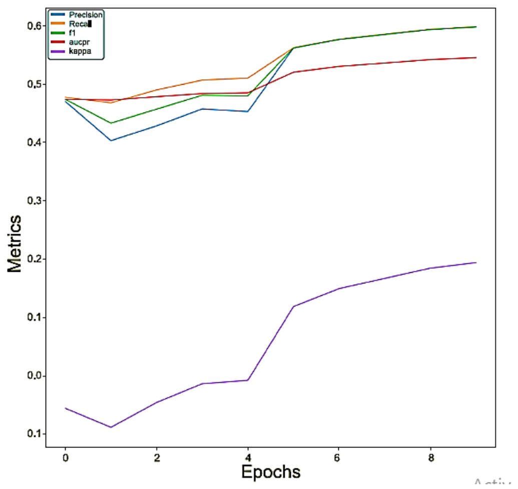


Figure 11. LSTM results

From what we have seen in the results, the most effective model for this task is CNN as more reliable in the Recall, F1 and Cohen's Kappa Score metrics.

3.3. Second experiment

The experiment focuses on studying student performance by analyzing the temporal sequence of actions performed by students in a specific degree course. Its objective is to evaluate performance phenomenon using timeseries data from a selected sample of students, focusing on the most challenging degree course each academic year. Degree course difficulty is determined by calculating the arithmetic mean of individual exam difficulties, representing the sum of exam difficulties divided by the total number of exams for that course in a given academic year.

Given that:

- $D^t = \{D_1^t, D_2^t, D_3^t, \ldots, D_m^t\}$ is a set of decimal values inherent to the difficulty of the single degree courses contained in the Unitelma dataset in the t-th academic year;
- n_i is the number of exams in the degree course i;
- δ_{ij}^t is the difficulty of the j-th exam belonging to the degree course i in the academic year t.

Then the degree course difficulty i is calculated as follows:

$$D_i^t = \frac{1}{n_i} \sum_{j=1}^{n} \delta_{ij}^t \qquad (9)$$

Where:

$$\delta_{ij}^t = \frac{\text{No.of students who passed the j exam in year t}}{\text{No.of students who attempted the j exam in year t}} * \frac{\bar{v}_{jt}}{31} \qquad (10)$$

And $\bar{v}_{jt}$ is the average grade of the exam j in the academic year t.

Therefore the most difficult degree course is established in the following way:

$$D^t_{min} = \min\{D^t_1, D^t_2, D^t_3, \dots, D^t_m\} \qquad (11)$$

The Unitelma dataset merges ESSE3 and Moodle data, categorized by academic year and degree course. Each academic year, data from the most challenging degree course is selected using a predefined formula. An attention-based neural model, CNN-LSTM, is trained on this data to analyze student timeseries and context features. These features encompass: p_1 = Number of attempts for all courses in the degree course in year t; p_2 = Student grade average up to year t; p_3 = Number of exams taken by the student in year t; p_4 = Student age; p_5 = Binary indicator (0 or 1) denoting if the student attempted the most challenging exam of their degree course in previous years; p_6 = The most difficult exam of the degree course in year t in which the student is enrolled.

$$P_6 = \delta^t_{min} \qquad (12)$$

And $\delta^t_{min} = \mathrm{mi}\{\delta^t_{i1}, \delta^t_{i2}, \delta^t_{i3}, \dots, \delta^t_{in_i}\}$ (13)

The experiment encountered challenges due to the sparsity of matrices representing student timeseries data. To address this, dimensionality reduction techniques were employed to reduce data dimensionality and alleviate the "curse of dimensionality" problem. Autoencoders, specifically feedforward networks trained with gradient descent and back-propagation, were implemented for this purpose. The autoencoder learns to compress input data while retaining relevant information, serving as an information compressor and effective representation identifier. After training the autoencoder, the embeddings generated from the input data were utilized to train the CNN-LSTM model, improving model performance.

3.3.1The CNN-LSTM model

The CNN-LSTM neural model architecture was designed by combining elements from the most effective algorithms in the previous experiment. It includes CLs with 3x3 filters, PLs with 2x2 average pooling, FCLs with 100 units, LSTM layer with 20 units, concatenation layer for context parameters, and an output layer with 1 unit. This architecture is visually represented in the following image.

ReLU activation function yielded the best results in terms of convergence speed and final accuracy for the two CLs and the subsequent FCL. The output layer utilizes the sigmoid activation function to limit output to a range between 0 and 1, beneficial for probability prediction.

$$\mathrm{Sigmoid}(x) = \frac{1}{1+e^{-x}} \qquad (14)$$

For what regards the loss function, sigmoid cross entropy was used to train the CNN-LSTM model. This function calculates the sum (for each instance in the batch) of the binary cross entropies comparing the predicted classes with the effective classes; then the final result will be divided by the batch size and changed sign.

Supposing that each instance of the batch is x, p(x) is the ground-truth distribution and q(x) is the estimated distribution by the model, the loss function is defined as following:

$$H(p.q) = \frac{1}{|batch|}\sum_{x \in batch} p(x)\log\big(q(x)\big) + \big(1 - p(x)\big)\log(1 - q(x)) \qquad (15)$$

The CNN-LSTM model was trained using the sigmoid cross-entropy loss function, which compares predicted classes with actual classes and calculates binary cross-entropies. The Adam optimizer, an extension of Stochastic Gradient Descent, was utilized for network estimation. A batch size of 10 was found to optimize estimation times.

3.3.2 Choice of metrics

The performance of the CNN-LSTM model is evaluated using the following metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-Squared (R^2).

Each degree course was split into 70% training data and 30% testing data. The CNN-LSTM model's (Fig 12) performance was evaluated on the test set, and the results are depicted in the following image.

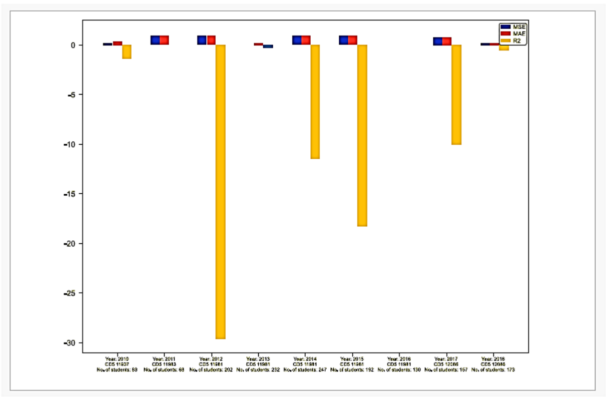


Figure 12. CNN-LSTM results

The graph above indicates a low margin of error (MSE and MAE) for performance prediction in each academic year. However, a significant issue observed in this experiment is class imbalance, where the number of withdrawn students far exceeds the number of graduates. This imbalance, coupled with a small number of instances, results in very low R-squared values (R2), particularly in the academic years 2012, 2014, 2015, and 2017. The limited number of instances hampers the accurate assessment of the CNN-LSTM model's performance.

3.4. Third experiment

The third experiment aims to identify optimal time windows for predicting student performance. It employs ML models to predict student performance within specific time periods. Two approaches are used: one based on pure data processing and the other incorporating oversampling and dimensionality reduction techniques for improved model performance. The dataset used is XuetangX, a MOOC platform, due to its larger data volume compared to Unitelma. Features extracted from student activities form instances represented as chronological action sequences. Models include the Kalman filter, LSTM-based recursive neural network, and TCN. TCN, being newer and more performant in sequence modeling tasks, may outperform traditional models like the Kalman filter. Detailed descriptions and architectures of both models are provided below.

3.4.1 Kalman filter

The Kalman filter is widely used for dynamic estimation problems involving stochastic noise affecting state and measurements. It minimizes the variance of the error between estimated and real states. It operates sequentially, predicting the current state and updating variables based on prediction errors to optimize future predictions. Notably, it evolves the error covariance matrix associated with forecast states over time. The filter comprises mathematical equations implementing a predictor-corrector estimator. It predicts the state of a discrete-time controlled process using the following equations:

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{16}$$

$$y_k = Hx_k + v_k \tag{17}$$

Where: $y \in R_n$ is the measurement value; w_k and v_k represent the process noise (p(w) _ N(0,Q)) and measurement noise (p(v) _ N(0,R)) respectively; A: n x n matrix relates previous state to current state; The B: $n \times l$ matrix relates the control input $u \in R^l$ to the state x; • The $H{:}\, m \times n$ matrix relates state to the measurement y_k.

3.4.2 Temporal Convolutional Network

TCNs are a specialized version of CNNs designed to excel at learning sequential patterns. They leverage dilated causal convolutions to capture dependencies between elements of a sequence and their preceding elements within a predefined time period. Unlike traditional convolutions, dilated convolutions connect neurons across layers at exponentially spaced intervals, enabling them to define larger receptive fields compared to recurrent neural networks without a significant increase in parameters. The structure of dilated convolutions is defined as follows:

$$F(i) *_l K = \sum_{n=-k}^{k} F(i + n.l) * K(n) \tag{25}$$

In TCNs, layers of neurons are stacked with increasing dilation factors, typically following powers of 2. This arrangement allows each neuron to have a wider receptive field, capturing information from multiple past time steps. In this experiment, a TCN with a kernel size of 2 and 4 layers of neurons was used, resulting in dilation factors of [1, 2, 4, 8]. This configuration enables each element in the sequence to consider information from the preceding 16 elements. Causal convolutions ensure that each output depends only on past inputs, making TCNs well-suited for sequence modeling tasks (Fig 13).

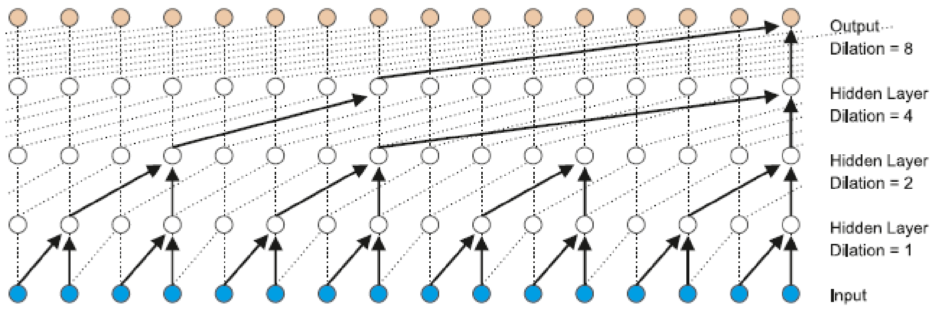


Figure 13. Stack of Causal Dilated Convolutions with l = [1, 2, 4, 8]

The TCN was equipped with performance, a technique that randomly removes a subset of neurons in the subsequent layer along with their connections from the current layer. This helps prevent overfitting by encouraging the network to generalize patterns instead of memorizing specific data during training.

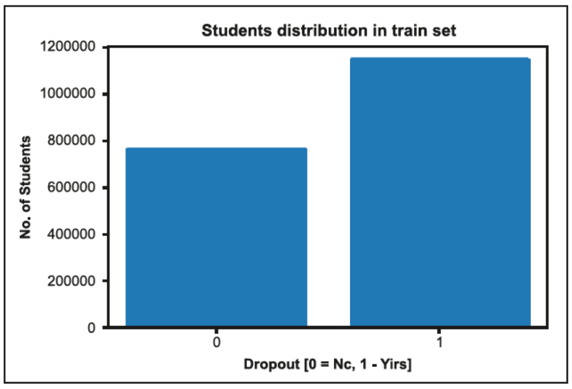
Residual connections are another key element of the TCN architecture. They address the issue of vanishing and exploding gradients, commonly encountered in deep networks. These connections involve comparing the output of a transformation, such as dilated convolutions, with the input to that transformation.

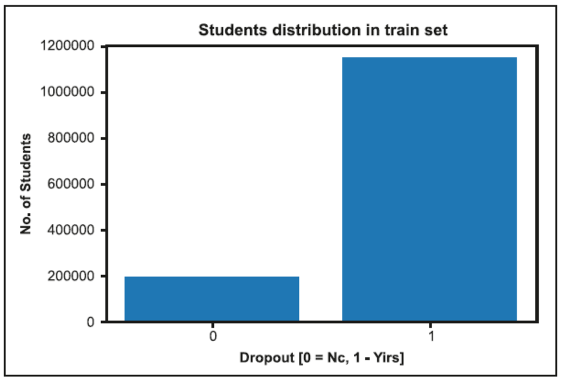
TCNs demonstrate superior efficiency compared to traditional recurrent networks like LSTMs, especially for long input sequences. LSTMs consume more memory to store partial results, while TCNs require less memory since they don't store intermediate results. TCNs also offer more control over the range of sequence elements through adjustments in kernel size or dilation factors. Additionally, TCNs allow parallel computation of convolutions, enabling faster processing of data during both training and testing phases. These advantages justify the preference for TCNs in addressing performance prediction. Moreover, ongoing advancements in TCN technology may further enhance their performance compared to LSTMs in the future.

3.4.3 Achieved results (first approach)

Imbalances across demographic, academic, and socioeconomic factors can bias predictive models assessing student performance. Varied representation in age, gender, ethnicity, and socioeconomic status, along with disparities in academic metrics, extracurricular activities, and course characteristics, can skew model perceptions is particularly pronounced in the test set. Below is a table 5 displaying the distribution of student classes in both the train set and the test set.

Table 5. Distribution of students in the train set and test set of XuetangX





Results from the models applied to classify performance in the XuetangX dataset are depicted in the following images (Fig 14-17). Performance metrics such as AUCPR, precision, recall, and F1 are used for evaluation.

Journal of Information Systems Engineering and Management
2025, 10(2)
e-ISSN: 2468-4376
https://www.jisem-journal.com/
Research Article

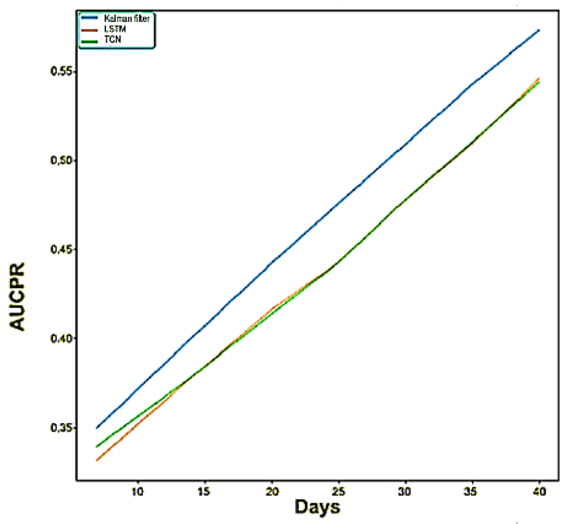


Figure 14. AUCPR plot

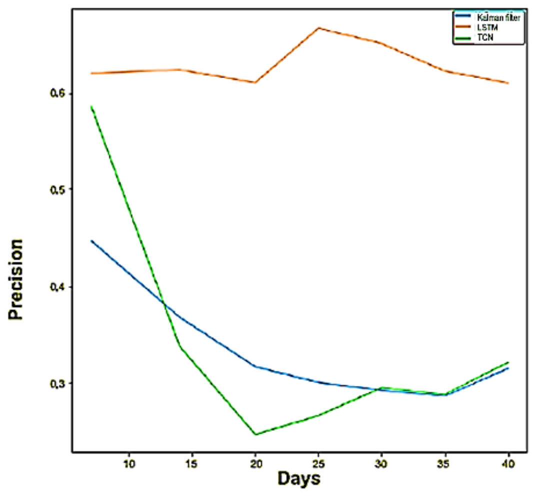


Figure 15. Precision plot

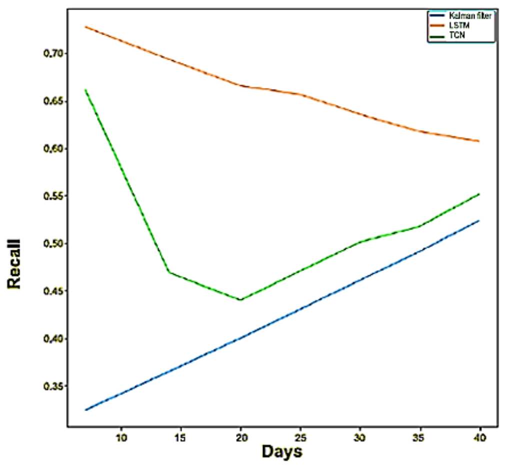


Figure 16. Recall plot

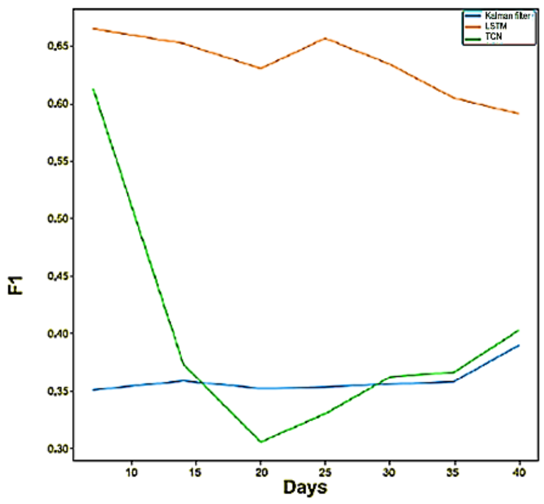


Figure 17. F1 plot

The Kalman Filter model exhibits increasing performance as the time window expands, yet it requires a longer time window to effectively differentiate between lower grade and higher grade. Conversely, LSTM demonstrates the most reliable performance across precision, recall, and F1 metrics, particularly with a 7-day time window. TCN performs adequately with a 7-day time window but exhibits fluctuating performance with longer time windows, falling short of LSTM's effectiveness.

3.4.4 Second approach

To address class imbalance and data sparsity issues, PCA was employed for data compression. PCA extracts the most relevant information describing data variability, enabling dimensionality reduction. This helps mitigate the effects of class imbalance and sparse data, enhancing model performance. Specifically, PCA was utilized to reduce the dataset's dimensionality by representing it in a space half its original size, thus improving model learning.

In detail, PCA performs a decomposition of independent variables into eigenvectors of the covariance matrix of X of dimension I × V , where I are the samples (observations), the latter defined by V independent variables. The covariance matrix is defined as follows:

$$□□□(□) = \frac{□^{□}□}{□-1} \quad (26)$$

The method divides a matrix X of rank R into a sum of R matrices Mr of rank 1, with r = 1, ...,R:

$$X = M_1 + M_2 + M_3 + \cdots + M_R \quad (27)$$

The generic Mr matrix can be represented with the outer product of two vectors tr and pr, score and loading. The X matrix can be rewritten as:

$$X = t_1 p_1^T + t_2 p_2^T + t_3 p_3^T + \cdots . t_R p_R^T \qquad (28)$$

PCA performs the algebraic approximation operation:

$$X = \sum_{a=1}^{k} t_a p_a^T + E = TP^T + E \qquad (29)$$

where k is the number of principal components, E is the residual matrix, T the score matrix (with shape I x k), with T = {t1, t2, t3, ..., tR} and P is the loading matrix (with shape k x V), with P = {p1, p2, p3, ..., pR}. The latter contains the main components sorted by row. The basic idea of the construction of P is to assign to each row pi an eigenvector of cov(X).

The results of the various analyzes carried out by the authors showed that ADASYN algorithm achieves competitive results (Fig 33-36), provides greater accuracy for both the minority class and the majority class, and not sacrifices one class to prefer another [13].

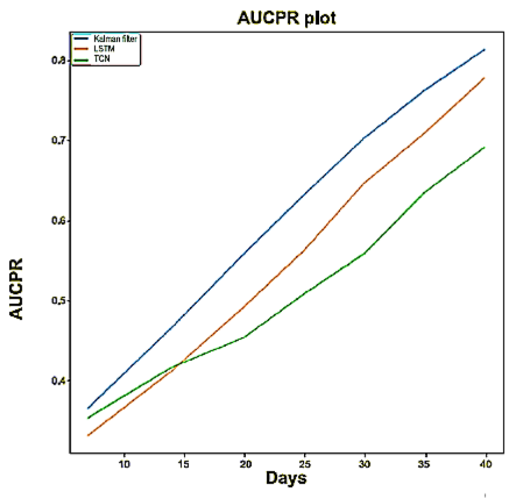


Figure 33. AUCPR plot

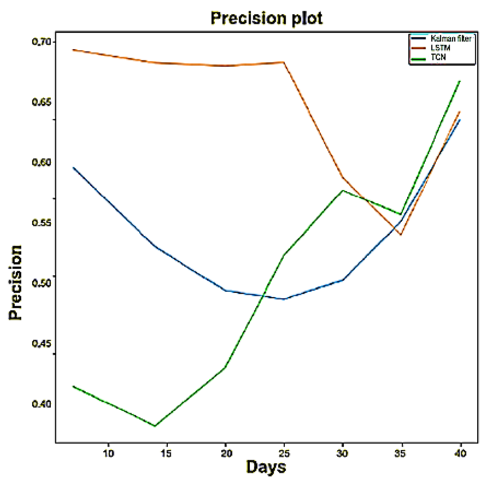


Figure 34. Precision plot

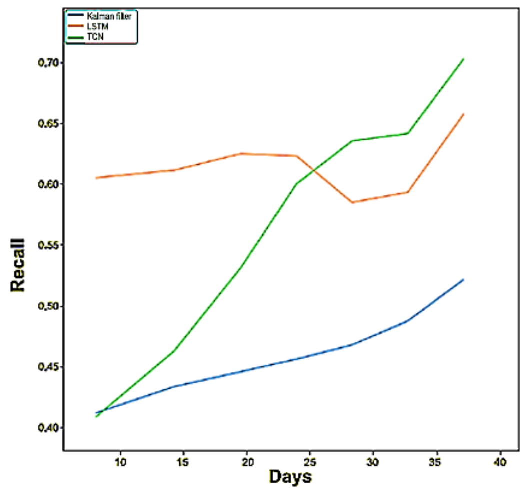


Figure 35. Recall plot

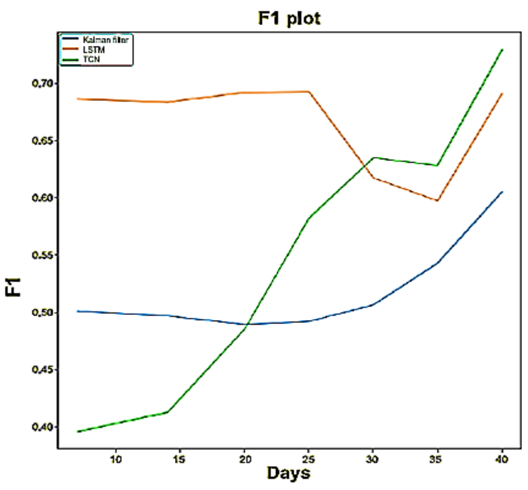


Figure 36. F1 plot

In the second experiment, Kalman Filter emerged as the most effective model based on AUCPR, with all models showing improved performance as the time window increased. TCN excelled in predictive abilities, especially in precision, recall, and F1 scores compared to LSTM. The application of oversampling and data densification strategies significantly enhanced TCN's performance, surpassing previous results.

4. CONCLUSIONS AND FUTURE WORK

This work delves into the challenging task of predicting student performance, addressing issues like data sparsity and class imbalance. Initially, ML and DL methods were compared using Unitelma dataset, with DL algorithms showing superior performance. The next experiment introduced an attention-based model combining CNN and LSTM, yielding promising results despite data limitations. Finally, the study explored

LSTM, TCN, and Kalman Filter models on XuetangX dataset, highlighting the effectiveness of oversampling and data densification techniques in improving model performance. Kalman Filter excelled in AUCPR, while LSTM and TCN showed superiority in binary classification, with TCN proving more efficient for longer sequences. Future work will focus on applying these techniques to the academic dataset, leveraging ADASYN and PCA algorithms, and further exploring TCN's potential as the best model for student performance prediction.

REFERENCES

[1] H. Waheed, S. U. Hassan, N. R. Aljohani, J. Hardman, S. Alelyani, and R. Nawaz, "Predicting academic performance of students from VLE big data using deep learning models," Comput. Human Behav., vol. 104, 2020, doi: 10.1016/j.chb.2019.106189.

[2] N. Tomasevic, N. Gvozdenovic, and S. Vranes, "An overview and comparison of supervised data mining techniques for student exam performance prediction," Comput. Educ., vol. 143, no. August 2019, p. 103676, 2020, doi: 10.1016/j.compedu.2019.103676.

[3] Z. Wu, T. He, C. Mao, and C. Huang, "Exam paper generation based on performance prediction of student group," Inf. Sci. (Ny)., vol. 532, pp. 72–90, 2020, doi: 10.1016/j.ins.2020.04.043.

[4] J. Yan, Z. Zhang, K. Lin, F. Yang, and X. Luo, "A hybrid scheme-based one-vs-all decision trees for multi-class classification tasks," Knowledge-Based Syst., vol. 198, p. 105922, 2020, doi: 10.1016/j.knosys.2020.105922.

[5] A. Khan and S. K. Ghosh, "Student performance analysis and prediction in classroom learning: A review of educational data mining studies", *Education and information technologies*, vol. 26, no. 1, pp. 205-240, 2021.

[6] S. Tangirala, "Evaluating the impact of GINI index and information gain on classification using decision tree classifier algorithm", *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 2, pp. 612-619, 2020.

[7] M. Tsiakmaki, G. Kostopoulos, S. Kotsiantis and O. Ragos, "Implementing AutoML in educational data mining for prediction tasks", *Appl. Sci.*, vol. 10, no. 1, pp. 90-117, Jan. 2020.

[8] R. G. Moises, D. P. P. R. Maria and O. Francisco, "Massive LMS log data analysis for the early prediction of course-agnostic student performance", *Comput. Educ.*, vol. 163, Apr. 2020.

[9] J. N. Walsh and A. Rísquez, "Using cluster analysis to explore the engagement with a flipped classroom of native and non-native English-speaking management students", *Int. J. Manage. Educ.*, vol. 18, no. 2, Jul. 2020.

[10] V. G. Karthikeyan, P. Thangaraj and S. Karthik, "Towards developing hybrid educational data mining model (HEDM) for efficient and accurate student performance evaluation", *Soft Comput.*, vol. 24, no. 24, pp. 18477-18487, Dec. 2020.

[11] L. M. Crivei, G. Czibula, G. Ciubotariu and M. Dindelegan, "Unsupervised learning based mining of academic data sets for students' performance analysis", *Proc. IEEE 14th Int. Symp. Appl. Comput. Intell. Informat. (SACI)*, pp. 11-16, May 2020.

[12] S. Delgado, F. Moran, J. C. S. Jose and D. Burgos, "Analysis of students' behavior through user clustering in online learning settings based on self organizing maps neural networks", *IEEE Access*, vol. 9, pp. 132592-132608, 2021.

[13] K. Okoye, A. Arrona-Palacios, C. Camacho-Zuñiga, J. A. G. Achem, J. Escamilla and S. Hosseini, "Towards teaching analytics: A contextual model for analysis of students' evaluation of teaching through text mining and machine learning classification", *Educ. Inf. Technol.*, vol. 1, pp. 1-43, Oct. 2021.

[14] E. S. V. Kumar, S. A. A. Balamurugan and S. Sasikala, "Multi-tier student performance evaluation model (MTSPEM) with integrated classification techniques for educational decision making", *Int. J. Comput. Intell. Syst.*, vol. 14, no. 1, pp. 1796-1808, Mar. 2021.

[15] A. Siddiqa, S. A. Z. Naqvi, M. Ahsan, A. Ditta, H. Alquhayz, M. A. Khan, et al., "An improved evolutionary algorithm for data mining and knowledge discovery", *Comput. Mater. Continua*, vol. 71, no. 1, pp. 1233-1247, 2022.

[16] Y. Wen, Y. Tian, B. Wen, Q. Zhou, G. Cai and S. Liu, "Consideration of the local correlation of learning behaviors to predict dropouts from MOOCs", *Tsinghua Sci. Technol.*, vol. 25, no. 3, pp. 336-347, Jun. 2020.

[17] S. Y. Lin, C. M. Wu, S. L. Chen and T. L. Lin, "Continuous facial emotion recognition method based on deep learning of academic emotions", *Sens. Mater.*, vol. 32, no. 10, pp. 3243-3259, Oct. 2020.

[18] A. Farissi, H. Mohamed Dahlan and Samsuryadi, "Genetic algorithm based feature selection with ensemble methods for student academic performance prediction", *J. Phys. Conf. Ser.*, vol. 1500, no. 1, Apr. 2020.

[19] H. Turabieh, S. A. Azwari, M. Rokaya, W. Alosaimi, A. Alharbi, W. Alhakami, et al., "Enhanced Harris hawks optimization as a feature selection for the prediction of student performance", *Computing*, vol. 7, pp. 1417-1438, Jan. 2021.

[20] H. B. Ma, S. Y. Yang, D. Z. Feng and L. C. Jiao, "Progressive mimic learning: A new perspective to train lightweight CNN models", *Neurocomputing*, vol. 456, pp. 220-231, Oct. 2021.

[21] A. Nabil, M. Seyam and A. Abou-Elfetouh, "Prediction of students' academic performance based on courses' grades using deep neural networks", *IEEE Access*, vol. 9, pp. 140731-140746, 2021.

[22] L. Gao, Z. Zhao, C. Li, J. Zhao and Q. Zeng, "Deep cognitive diagnosis model for predicting students' performance", *Future Gener. Comput. Syst.*, vol. 126, pp. 252-262, Jan. 2022.

[23] P. Mishra, A. Biancolillo, J. M. Roger, F. Marini and D. N. Rutledge, "New data preprocessing trends based on ensemble of multiple preprocessing techniques", *TrAC Trends Anal. Chem.*, vol. 132, Nov. 2020.

[24] Thamilarasi V., Roselin. R., "Application of Machine Learning in chest X-

a. Ray images", International book & River publisher series in computing and information science and Technology, pp: 52-1,2023. ISBN : 9788770228114, 9788770228107.

[25] Vijayalakshmi, V. and Venkatachalapathy, K.. 2019. "Comparison of Predicting Students Performance using Machine Learning Algorithms". None. https://doi.org/10.5815/ijisa.2019.12.04

[26] Vijayalakshmi, V. and Venkatachalapathy, K.. 2019. "Comparison of Predicting Students Performance using Machine Learning Algorithms". None. https://doi.org/10.5815/ijisa.2019.12.04

[27] Yan, Lijuan and Liu, Yanshen. 2020. "An Ensemble Prediction Model for Potential Student Recommendation Using Machine Learning". Multidisciplinary Digital Publishing Institute. https://doi.org/10.3390/sym12050728

[28] Babi, Ivana Urevi. 2017. "Machine learning methods in predicting the student academic motivation". Croatian Operational Research Society. https://doi.org/10.17535/crorr.2017.002

[29] Zhang, Yupei, Yun, Yue, An, Rui, Cui, Jiaqi, Dai, Huan, and Shang, Xuequn. 2021. "Educational Data Mining Techniques for Student Performance Prediction: Method Review and Comparison Analysis". Frontiers Media. https://doi.org/10.3389/fpsyg.2021.698490

[30] Hernndez-Blanco, Antonio, Herrera-Flores, Boris, Toms, David, and Navarro-Colorado, Borja. 2019. "A Systematic Review of Deep Learning Approaches to Educational Data Mining". Hindawi Publishing Corporation. https://doi.org/10.1155/2019/1306039

[31] in, Trn Thanh, Hoai, Sang, Nguyn, Hi Thanh, and Thai-Nghe, Nguyen. 2020. "Deep Learning with Data Transformation and Factor Analysis for Student Performance Prediction". Science and Information Organization. https://doi.org/10.14569/ijacsa.2020.0110886

[32] Li, Shuping and Liu, Taotang. 2021. "Performance Prediction for Higher Education Students Using Deep Learning". Hindawi Publishing Corporation. https://doi.org/10.1155/2021/9958203

[33] Kim, ByungHak, Vizitei, Ethan, and Ganapathi, Varun. 2018. "GritNet: Student Performance Prediction with Deep Learning". Cornell University. https://doi.org/10.48550/arxiv.1804.07405

[34] Yousafzai, Bashir Khan, Khan, Sher Afzal, Rahman, Taj, Khan, Inayat, Ullah, Inam, Rehman, Ateeq Ur, Baz, Mohammed, Hamam, Habib, and Cheikhrouhou, Omar. 2021. "Student-Performulator: Student Academic Performance Using Hybrid Deep Neural Network". Multidisciplinary Digital Publishing Institute. https://doi.org/10.3390/su13179775

[35] Manning, Christopher D.. 2015. "Computational Linguistics and Deep Learning". Association for Computational Linguistics. https://doi.org/10.1162/coli_a_00239

[36] A.Al-azazi, Fatima Ahmed and Ghurab, Mossa. 2023. "ANN-LSTM: A deep learning model for early student performance prediction in MOOC". Elsevier BV. https://doi.org/10.1016/j.heliyon.2023.e15382

[37] Chen, HsingChung, Prasetyo, Eko, Tseng, ShianShyong, Putra, Karisma Trinanda, Prayitno, Prayitno, Kusumawardani, Sri Suning, and Weng, ChienErh. 2022. "Week-Wise Student Performance Early Prediction in Virtual Learning Environment Using a Deep Explainable Artificial Intelligence". Multidisciplinary Digital Publishing Institute. https://doi.org/10.3390/app12041885

[38] Liu, Dong, Zhang, Yunping, Zhang, Jun, Li, Qinpeng, Zhang, Congpin, and Yin, Yu. 2020. "Multiple Features Fusion Attention Mechanism Enhanced Deep Knowledge Tracing for Student Performance Prediction". Institute of Electrical and Electronics Engineers. https://doi.org/10.1109/access.2020.3033200