International Journal of Basic and Applied Sciences, 14 (2) (2025) 296-310

International Journal of Basic and Applied Sciences

Website: www.sciencepubco.com/index.php/IJBAS
https://doi.org/10.14419/88yz9489
Research paper

Enhancing Student Performance Prediction Through Machine Learning and Deep Learning Models

M. Kannan [1]*, S. Albert Antony Raj [2]

[1] Department of Computer Science, Faculty of Science and Humanities, SRM Institute of Science and Technology, Kattankulathur Campus, Chennai, Tamilnadu, India
[2] Department of Computer Applications, Faculty of Science and Humanities, SRM Institute of Science and Technology, Kattankulathur, Chennai, Tamilnadu, India. alberts@srmist.edu.in
**Corresponding author E-mail: km1200@srmist.edu.in*

Received: April 15, 2025, Accepted: June 9, 2025, Published: June 22, 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 Swayam 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 NPTEL 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 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 layer; Recursive layer*

1. 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 (Waheed et al., 2020; Tomasevic et al., 2020; Wu et al., 2020)

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 (Yan et al., 2020; Khan & Ghosh, 2021). Varied representation in age, gender, ethnicity, and socioeconomic status, along with disparities in academic metrics, extracurricular activities, and course characteristics, can skew model perceptions (Tangirala, 2020; Tsiakmaki et al., 2020).

Existing research has shown that DL algorithms, particularly CNN and LSTM, demonstrate superior performance in student performance prediction tasks compared to traditional statistical methods (Moises et al., 2020; Walsh & Rísquez, 2020). 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 (Karthikeyan et al., 2020; Crivei et al., 2020).

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 Swayam 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 NPTEL 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 (Delgado et al., 2021; Okoye et al., 2021).

2. Literature Review

Kumar et al. (2021) develops a Model using single and ensemble classifiers. Data preprocessing removes duplicates and redundant data, improving accuracy. The primary classification phase uses Naive Bayes, and the secondary phase uses ensemble classifiers like Boosting, Stacking, and Random Forest. Performance analysis shows the model's classification accuracy and efficiency. Another study by Siddiqa et al. (2022) presents an improved evolutionary algorithm for feature subset selection with neuro-fuzzy classification for educational data mining. The Chaotic Whale Optimization Algorithm (CWOA) selects relevant feature subsets for better classification, followed by Neuro-Fuzzy Classification (NFC).

Based on MOOC learning characteristics, a study by Wen et al. (2020) proposes a feature matrix for local learning behavior correlations and a CNN model for performance prediction, showing improved accuracy with early predictions. Another method by Lin et al. (2020) combines CNN and LSTM networks for continuous facial emotional pattern recognition to analyze academic emotions, enhancing e-learning systems' ability to understand and motivate students. Experimental results show a 12% accuracy improvement over using CNN alone.

Farissi et al. (2020) proposes using Genetic Algorithm (GA) for feature selection combined with ensemble classification to predict academic performance. GA achieves optimal feature selection for better accuracy, evaluated using AUC on Kaggle data, showing impressive prediction results. Another approach by Turabieh et al. (2021) employs a dynamic controller and KNN algorithm for feature selection and clustering, using various classifiers, including LRNN, achieving 92% accuracy with the enhanced HHO and LRNN combination. Progressive Mimic Learning (PML) by Ma et al. (2021) trains lightweight CNN models by mimicking a teacher model's (TM) learning trajectory. Landmarks from TM stages guide the student model (SM), improving prediction accuracy on datasets like CIFAR-10 and ImageNet-10. Another study by Nabil et al. (2021) explores DL for predicting academic performance and identifying at-risk students using models like DNN, decision trees, and random forests, achieving 89% accuracy. Various resampling methods address imbalanced data.

A deep cognitive diagnosis framework by Gao et al. (2022) enhances traditional methods with DL to assess students' mastery of skills and problems, considering careless mistakes and guessing. Experiments on real-world datasets show the model's effectiveness. Mishra et al. (2020) reviews ensemble preprocessing strategies in chemometrics, highlighting their application beyond spectral data to improve models where identifying a single best option is challenging.

Finally, Zhao et al. (2025) proposed a novel approach that combines image-based behavioral data transformation with ensemble learning to improve child achievement prediction. In their study, traditional behavioral metrics such as login frequency, session duration, and number of interactions were transformed into visual representations and then processed using ensemble learning techniques such as random forests and gradient boosting. This innovative approach improved the accuracy and robustness of the model, especially when dealing with issues such as class imbalance using techniques such as SMOTE. Their results showed that transforming behavioral data into image form can effectively mine spatial patterns of student engagement, which brings a new dimension to the prediction model.

In addition to this methodological advance, Hasan et al. (2021) also integrated data from various learning platforms, including student information systems (SIS), Moodle learning management systems, and mobile learning applications, "eDify," to develop a comprehensive dataset. The dataset covers a variety of student-related variables, ranging from demographic attributes, academic records to real-time learning behaviors. By providing both static and dynamic data, this study facilitates multimodal analysis and helps develop more context-aware prediction models. Their dataset enables researchers to examine correlations between platform usage patterns and academic outcomes, which is critical for developing early intervention strategies.

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	

4. Experiments

In the initial experiment of the work project, the goal was to establish baselines for predicting student performance using ML models. 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`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`d had higher or lower grades, dataset D1.1 are utilized. D1.1 was checked first to identify students who had either a higher grade or a lower grade in 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 was 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.

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

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

Where 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 Swayam dataset.
Specifically, the tensor $T = [s_1, s_2, s_3, s_4, \dots, s_k]$ represents the actions of (k) students. Each student (s_i^t) has a matrix of actions over time: $s_i^t = [a_1, a, a_3, a_4, \dots, a_W]$ is the vector of actions for student i on day t (for ($t = 1, 2, \dots, N$)). Each element (a_j) within (s_i^t) denotes the number of times action j was performed by the student on that day.
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 a higher grade, while 1 indicates a student who has a lower garde.
Formally, $y = [c_1, c_2, c_3, c_4, \dots, c_k]$, where is the number of students in dictionary d:

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

This vector is used for both approaches.

4.1 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).

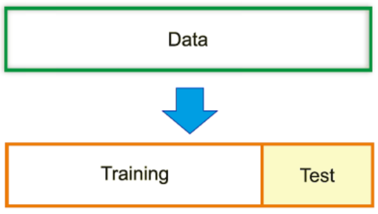


Fig. 1: Train Set/Test Set Splitting

4.1.1 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 $h_\theta(x) = \vartheta_0 + \vartheta_1 x_1 + \vartheta_2 x_2 + \cdots . + \vartheta_k x_k$.

4.1.2 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.

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

4.1.3 Decision Tree

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.

4.1.4 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.

4.1.5 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).

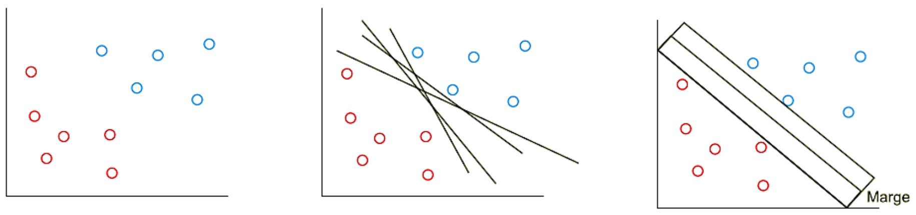


Fig. 2: Representation of SVM margin

Large margins in decision boundaries are preferred because they typically result in lower generalization errors, 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.

4.1.6 Naive Bayes

NB is a classification algorithm based on Bayes' theorem, applying a probabilistic approach to solving classification problems. It assumes independence between features given the class variable yvariable y. Bayes' theorem states the relationship between the class variable y 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 are processed for the prediction of the performance.

4.1.7 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 a 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.

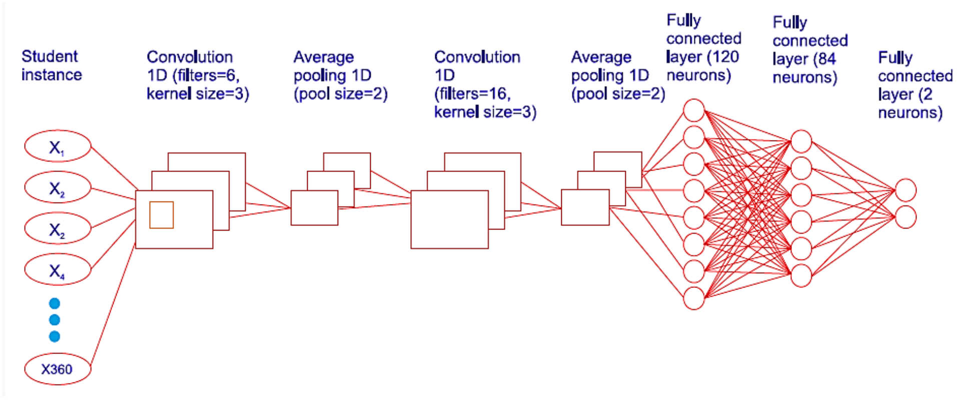

Student instance
Convolution 1D (filters=6, kernel size=3)
Average pooling 1D (pool size=2)
Convolution 1D (filters=16, kernel size=3)
Average pooling 1D (pool size=2)
Fully connected layer (120 neurons)
Fully connected layer (84 neurons)
Fully connected layer (2 neurons)
X1
X2
X2
X4
X360

Fig. 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.

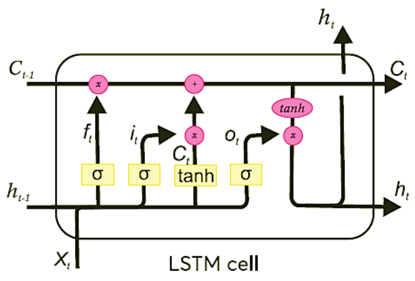
$$\mathrm{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (6)$$

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 $\mathcal{X}$X represents each instance in the batch, $p(x)$as the ground-truth distribution, and $q(x)$ is the estimated distribution by the model, the loss function is defined as:

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

4.1.8 Long-Short Term Memory

Recurrent networks, unlike other types, process input 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, BC employs softmax activation to generate normalized output values. This architecture adeptly handles sequential data, facilitating the production of sequenced output (Fig. 5).

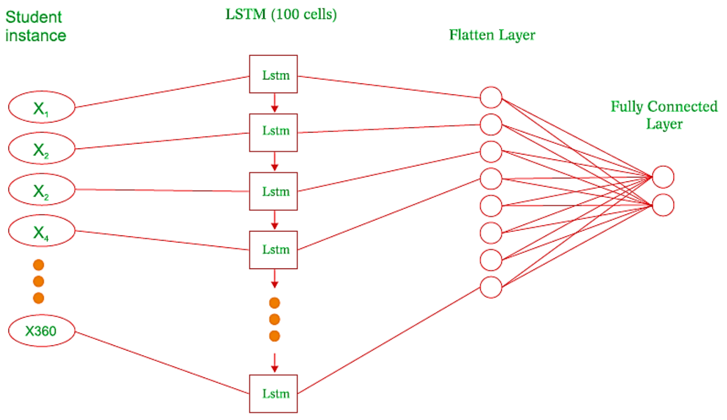


Fig. 5: Architecture of the LSTM neural network

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 [26-27].

4.1.9 Choice of metrics

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

4.2 Achieved results

4.2.1 First approach

This section shows the training results of the algorithms mentioned in the previous paragraphs. The following performances 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

Algorithm	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).

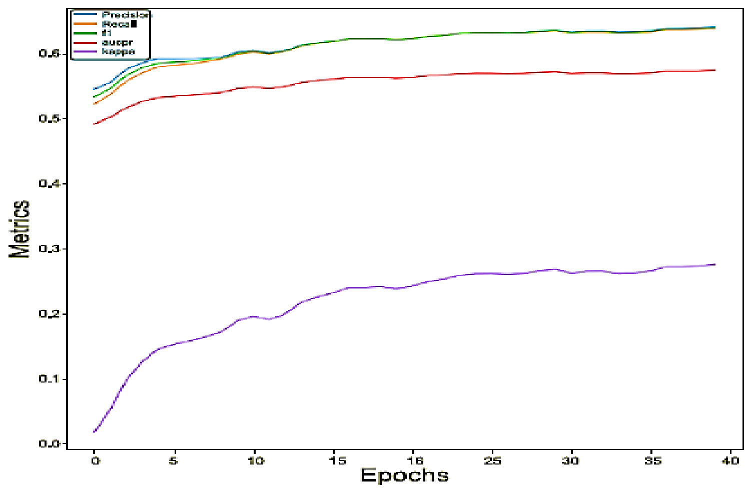


Fig. 6: CNN results

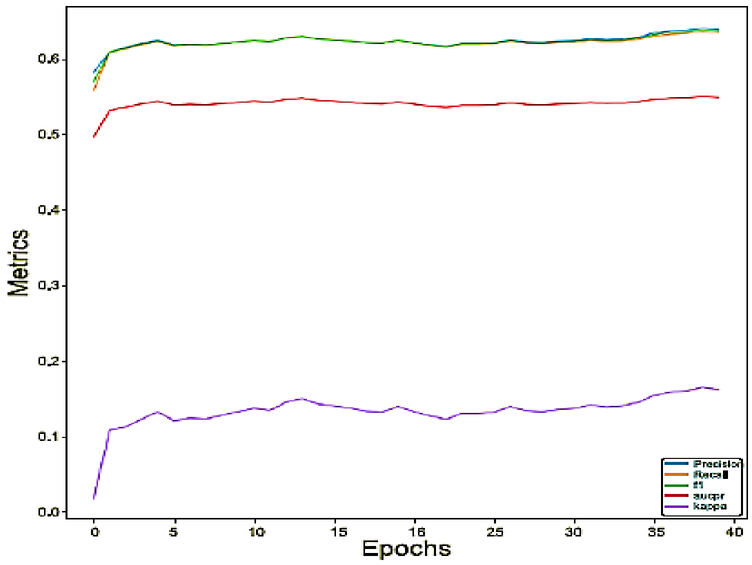


Fig. 7: CNNwith Separable Convolution results

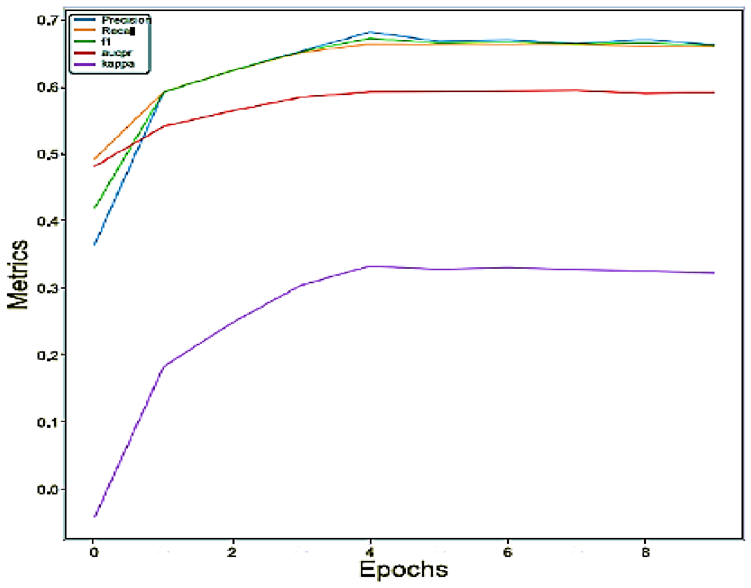


Fig. 8: LSTM results

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

4.2.2 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 Figures 9-11).

Table 3: Performance Comparison: ML vs. DL Algorithms

Algorithm	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
CNN with Separable Convolution	0.700	0.671	0.685	0.617	0.334
LSTM	0.597	0.598	0.597	0.544	0.193

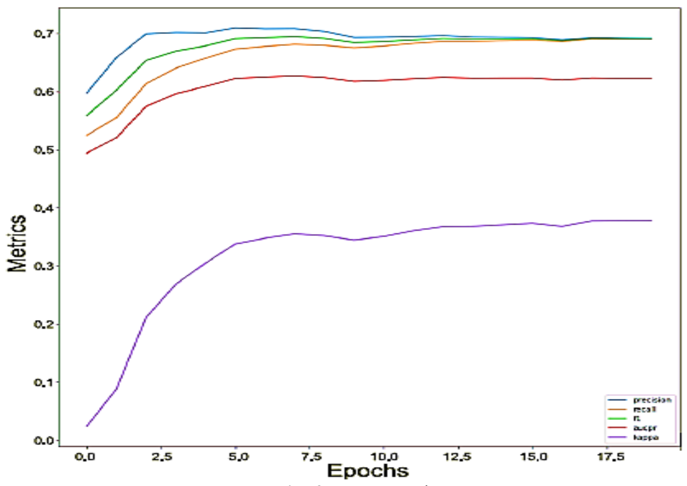


Fig. 9: CNN results

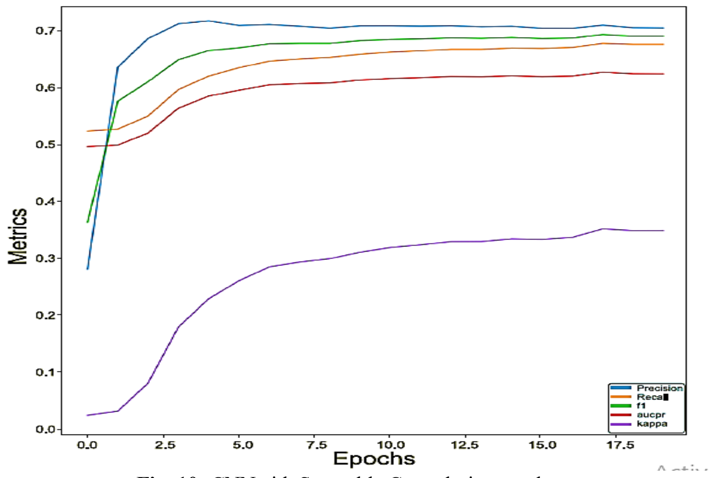


Fig. 10: CNNwith Separable Convolution results

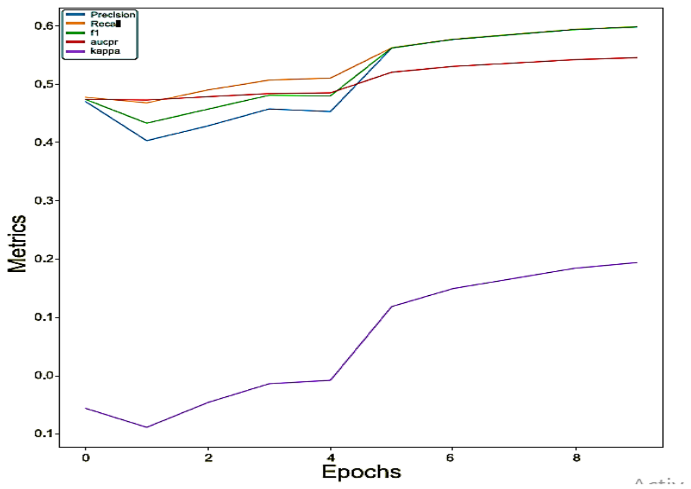


Fig. 11: LSTM results

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

4.3 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 phenomena using time series 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:

- $D^t = \{D_1^t, D_2^t, D_3^t, \dots, D_m^t\}$ is a set of decimal values inherent to the difficulty of the single degree courses contained in the Swayam 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:

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

Where:

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

And $\bar{\mathbf{v}}_{\mathbf{jt}}$ 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:

$$D_{min}^t = \min\{D_1^t, D_2^t, D_3^t, \dots, D_m^t\} \tag{11}$$

The Swayam 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: $\mathbf{p_1}$ = Number of attempts for all courses in the degree course in year t; $\mathbf{p_2}$ = Student grade average up to year t; $\mathbf{p_3}$ = Number of exams taken by the student in year t; $\mathbf{p_4}$ = Student age; $\mathbf{p_5}$ = Binary indicator (0 or 1) denoting if the student attempted the most challenging exam of their degree course in previous years; $\mathbf{p_6}$ = The most difficult exam of the degree course in year t in which the student is enrolled.

$$P_6 = \delta^t_{min} \tag{12}$$

$$\text{And } \delta^t_{min} = mi\{\delta^t_{i1}, \delta^t_{i2}, \delta^t_{i3}, \ldots, \delta^t_{in_i}\} \tag{13}$$

The experiment encountered challenges due to the sparsity of matrices representing student time series 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.

4.3.1 The 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, a 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.

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

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 result will be divided by the batch size and changed in 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 follows:

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

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.

4.3.2 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 (R2).
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.

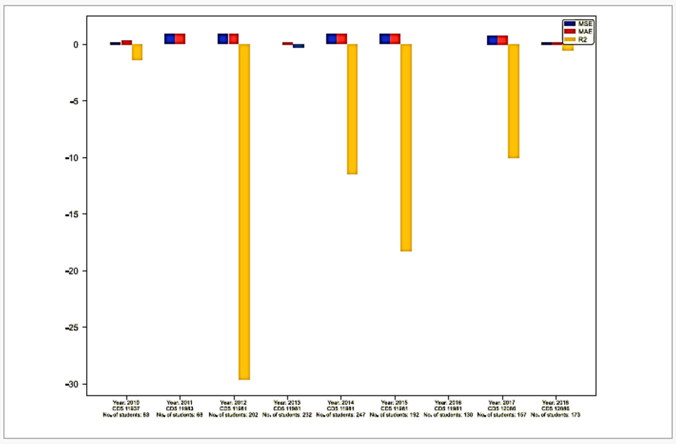


Fig. 12: CNN-LSTM results

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.

4.4. 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 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 NPTEL, a MOOC platform, due to its larger data volume compared to Swayam. 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.

4.4.1 Kalman filter

The Kalman filter is widely used for dynamic estimation problems involving stochastic noise affecting the 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: $\mathbf{y} \in \mathbf{R_n}$ is the measurement value; $\mathbf{w_k}$ and $\mathbf{v_k}$ represent the process noise (p(w) _ N(0,Q)) and measurement noise (p(v) _ N(0,R)) respectively; A: **n x n** matrix relates the previous state to the current state; The B: $\mathbf{n} \times \mathbf{l}$ matrix relates the control input $\mathbf{u} \in \mathbf{R^l}$ to the state x; • The **H**: $\mathbf{m} \times \mathbf{n}$ matrix relates the state to the measurement $\mathbf{y_k}$.

4.4.2 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. 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:

$$F(i) *_l K = \sum_{n=-k}^{k} F(i + n.l) * K(n) \tag{18}$$

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).

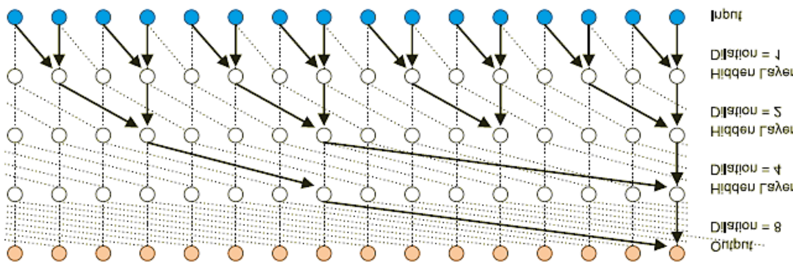


Fig. 13: Stack of Causal Dilated Convolutions with l = [1, 2, 4, 8]

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.

4.4.3 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 in the test set. Below is Figure 14- 15 displaying the distribution of student classes in both the train set and the test set.

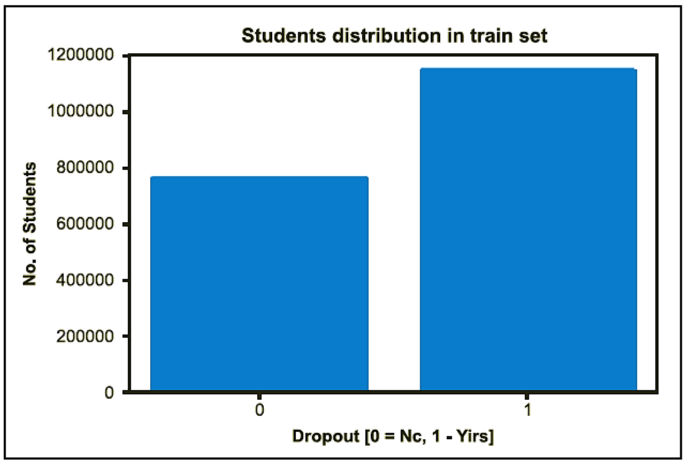


Fig. 14: Distribution of students on the train set of NPTEL

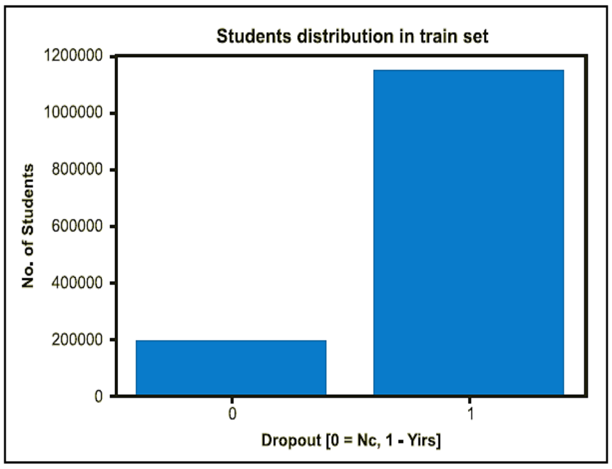


Fig. 15: Distribution of students in the test set of NPTEL

Results from the models applied to classify performance in the NPTEL dataset are depicted in the following images (Figures 16-19). Performance metrics such as AUCPR, precision, recall, and F1 are used for evaluation.

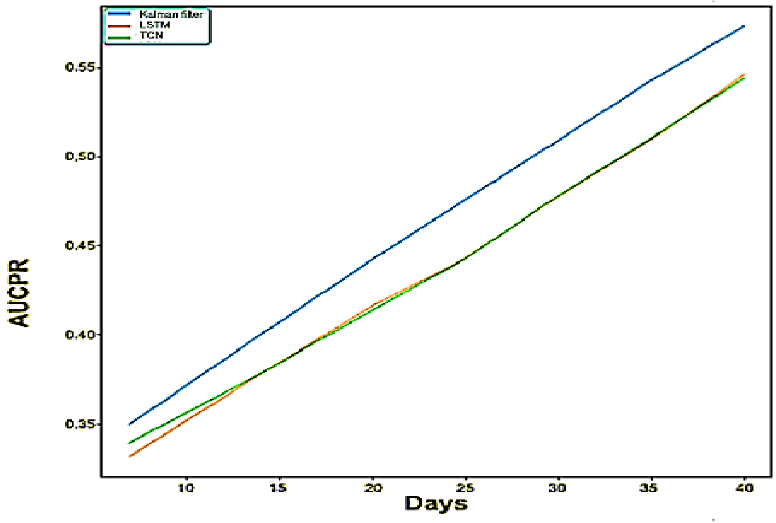


Fig. 16: AUCPR plot

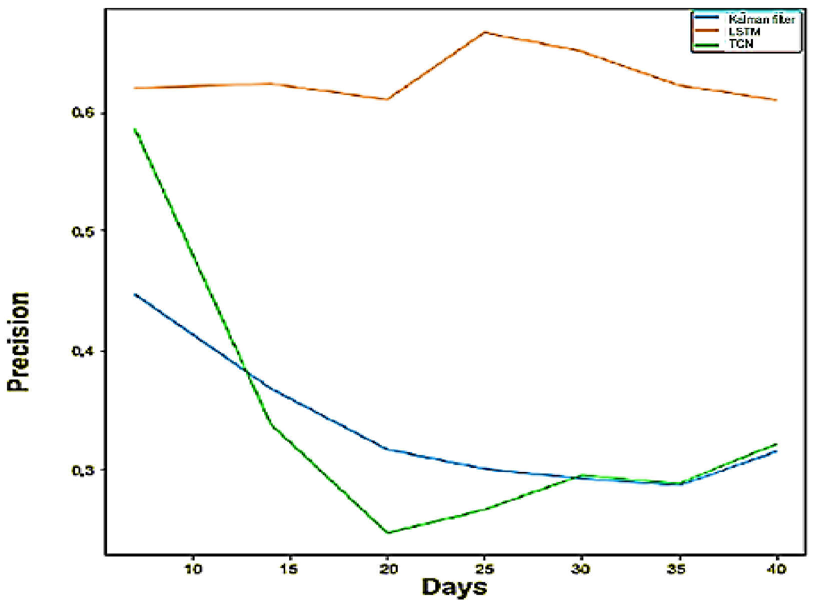


Fig.17: Precision plot

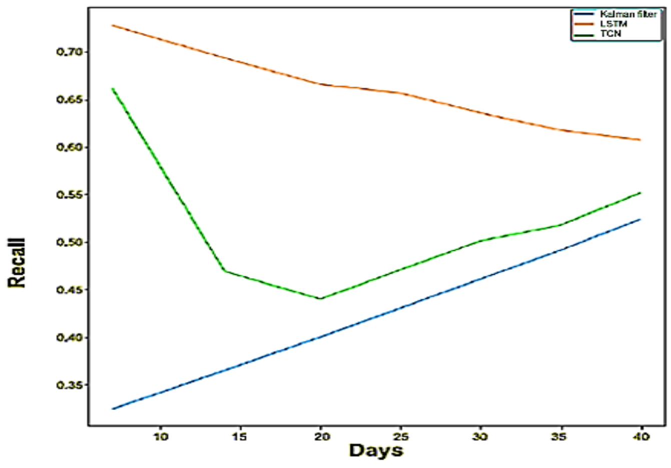


Fig. 18: Recall plot

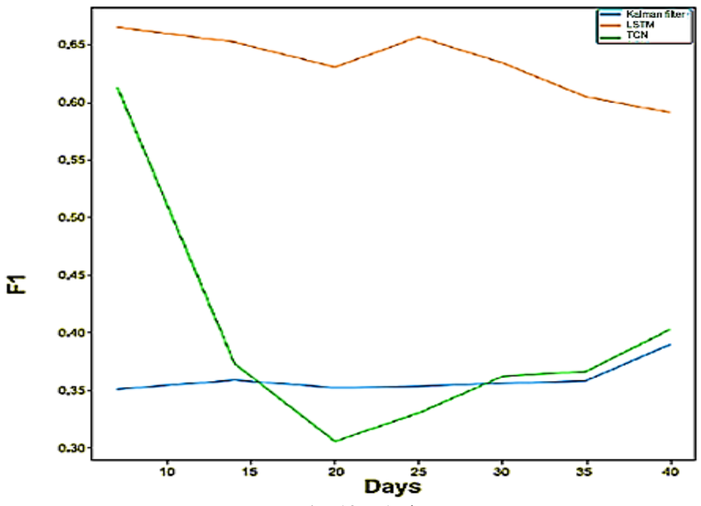


Fig. 19: F1 plot

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.

4.4.5 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 $\mathbf{I} \times \mathbf{V}$, where I am the samples (observations), the latter defined by V independent variables. The covariance matrix is defined as follows:

$$con(X) = \frac{X^T X}{I-1} \tag{19}$$

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

$$\mathrm{X} = \mathrm{M}_1 + \mathrm{M}_2 + \mathrm{M}_3 + \cdots + \mathrm{M}_\mathrm{R} \tag{20}$$

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:

$$\mathrm{X} = \mathrm{t}_1\mathrm{p}_1^\mathrm{T} + \mathrm{t}_2\mathrm{p}_2^\mathrm{T} + \mathrm{t}_3\mathrm{p}_3^\mathrm{T} + \cdots . \mathrm{t}_\mathrm{R}\mathrm{p}_\mathrm{R}^\mathrm{T} \tag{21}$$

PCA performs the algebraic approximation operation:

$$\mathrm{X} = \sum_{a=1}^{k} \mathrm{t}_a\mathrm{p}_a^\mathrm{T} + \mathrm{E} = \mathrm{TP}^\mathrm{T} + \mathrm{E} \tag{22}$$

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 = x, 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 analyses carried out by the authors showed that the ADASYN algorithm achieves competitive results (Figs 20-23), provides greater accuracy for both the minority class and the majority class, and does not sacrifice one class to prefer another [13].

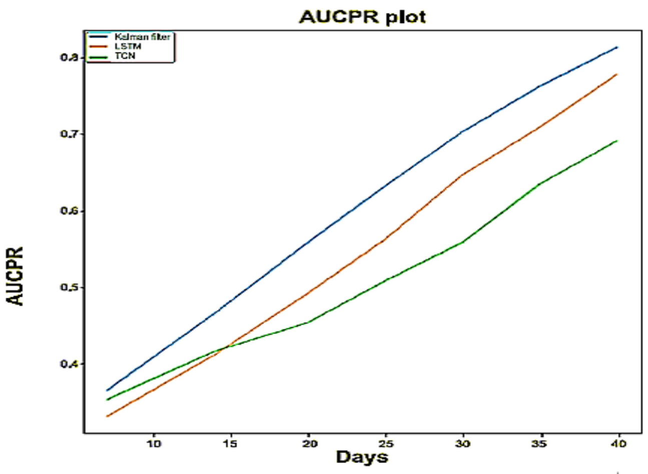


Fig. 20: AUCPR plot

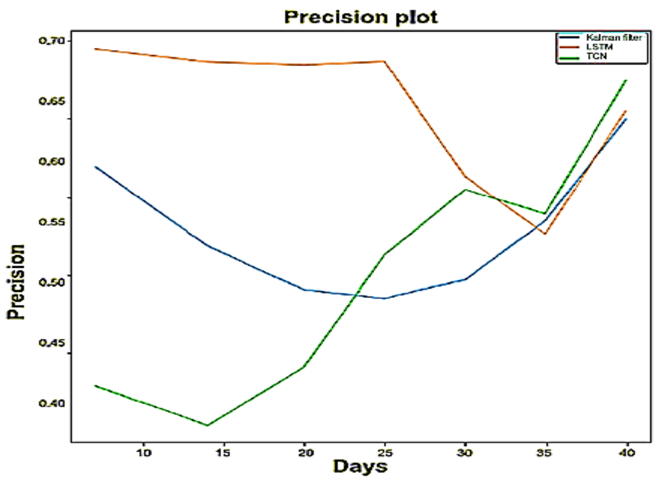


Fig. 21: Precision plot

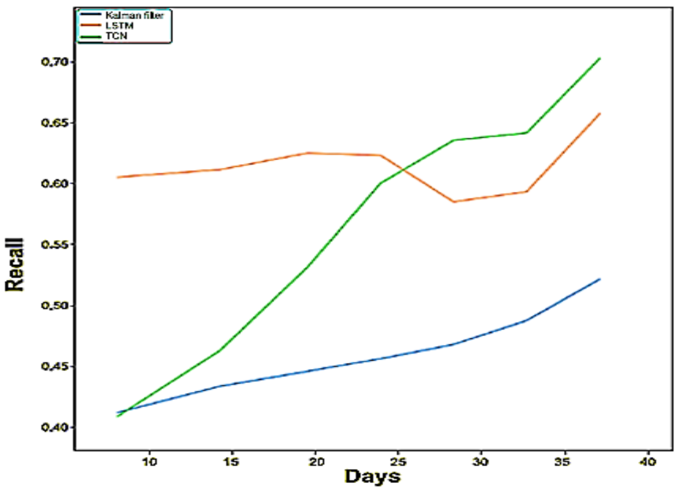


Fig. 22: Recall plot

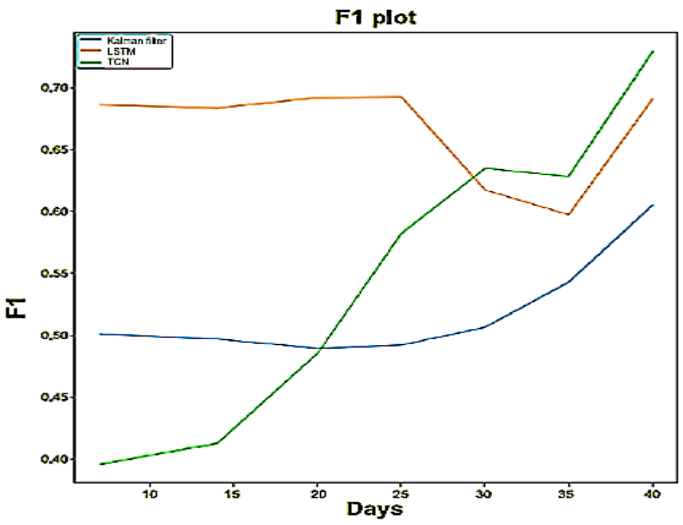


Fig. 23: F1 plot

In the second experiment, the 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.

The results of this study not only advance computational methods in the field of learning analytics but also have important interdisciplinary implications. In psychology, student behavior prediction models challenge cognitive load theory and motivational science, helping to understand learning patterns and dropout risks. From a sociological perspective, the analysis of demographic and socioeconomic inequalities highlights systemic gaps in education and calls for more inclusive support structures. From a policy perspective, these findings may guide targeted interventions, curriculum modifications, and resource allocation to promote equity and improve learning outcomes for diverse student groups.

These experiments not only improve technical accuracy but also demonstrate interdisciplinary relevance. The time series model reflects the psychological patterns of student behavior and examines inequality in classrooms with a focus on social equity. Dimensionality reduction improves comprehension and supports data-driven education policies. In summary, this research connects machine learning to real-world scientific and social understandings.

Although models such as LSTM and TCN show good results in specific scenarios, their performance varies depending on the characteristics of the dataset and experimental conditions. The results do not indicate that there is a superior model, but rather that the effectiveness of the model is highly dependent on the specific situation and deserves further study in a wider range of scenarios.

This study demonstrates the value of combining statistical, machine learning, and deep learning methods to improve student learning. By combining these frameworks, this study can improve the accuracy of the analyses and support more effective educational interventions by demonstrating the power of interdisciplinary collaboration.

Learning Insights helps Western MOOC platforms such as Coursera and EdX increase student engagement and personalize learning by applying the same data analysis and prediction methods to different learner groups.

The findings support Vygotsky's idea that social interaction promotes learning and that online platforms do so in digital ways. The findings are consistent with Bourdieu's social capital theory, which suggests that the use of MOOCs affects educational equity and helps bridge the digital divide. Its research combines psychology and social factors to provide a deeper understanding.

5. 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 the Swayam 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 the NPTEL 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.

Acknowledgement

The authors acknowledge the head of the institutions of the respective colleges for providing support to conduct this research.

References

[1] Waheed, H., Hassan, S. U., Aljohani, N. R., Hardman, J., Alelyani, S., & Nawaz, R. (2020). Predicting the academic performance of students from VLE big data using deep learning models. Computers in Human Behavior, 104, 106189.

[2] Tomasevic, N., Gvozdenovic, N., & Vranes, S. (2020). An overview and comparison of supervised data mining techniques for student exam performance prediction. Computers & Education, 143, 103676.

[3] Wu, Z., He, T., Mao, C., & Huang, C. (2020). Exam paper generation based on the performance prediction of the student group. Information Sciences, 532, 72–90.

[4] Yan, J., Zhang, Z., Lin, K., Yang, F., & Luo, X. (2020). A hybrid scheme-based one-vs-all decision trees for multi-class classification tasks. Knowledge-Based Systems, 198, 105922.

[5] Khan, A., & Ghosh, S. K. (2021). Student performance analysis and prediction in classroom learning: A review of educational data mining studies. Education and Information Technologies, 26(1), 205–240.

[6] Tangirala, S. (2020). Evaluating the impact of GINI index and information gain on classification using the decision tree classifier algorithm. International Journal of Advanced Computer Science and Applications, 11(2), 612–619.

[7] Tsiakmaki, M., Kostopoulos, G., Kotsiantis, S., & Ragos, O. (2020). Implementing AutoML in educational data mining for prediction tasks. Applied Sciences, 10(1), 90–117.

[8] Moises, R. G., Maria, D. P. P. R., & Francisco, O. (2020). Massive LMS log data analysis for the early prediction of course-agnostic student performance. Computers & Education, 163, 104083.

[9] Walsh, J. N., & Rísquez, A. (2020). Using cluster analysis to explore the engagement with a flipped classroom of native and non-native English-speaking management students. The International Journal of Management Education, 18(2), 100362.

[10] Karthikeyan, V. G., Thangaraj, P., & Karthik, S. (2020). Towards developing hybrid educational data mining model (HEDM) for effi-cient and accurate student performance evaluation. Soft Computing, 24(24), 18477–18487.

[11] Crivei, L. M., Czibula, G., Ciubotariu, G., & Dindelegan, M. (2020). Unsupervised learning-based mining of academic data sets for students' performance analysis. Proceedings of the IEEE 14th International Symposium on Applied Computational Intelligence and In-formatics (SACI), 11–16.

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

[13] Okoye, K., Arrona-Palacios, A., Camacho-Zuñiga, C., Achem, J. A. G., Escamilla, J., & Hosseini, S. (2021). Towards teaching analyt-ics: A contextual model for analysis of students' evaluation of teaching through text mining and machine learning classification. Education and Information Technologies, 26, 1–43.

[14] Kumar, E. S. V., Balamurugan, S. A. A., & Sasikala, S. (2021). Multi-tier student performance evaluation model (MTSPEM) with integrated classification techniques for educational decision making. International Journal of Computational Intelligence Systems, 14(1), 1796–1808.

[15] Siddiqa, A., Naqvi, S. A. Z., Ahsan, M., Ditta, A., Alquhayz, H., Khan, M. A., et al. (2022). An improved evolutionary algorithm for data mining and knowledge discovery. Computer Modeling in Engineering & Sciences, 71(1), 1233–1247.

[16] Wen, Y., Tian, Y., Wen, B., Zhou, Q., Cai, G., & Liu, S. (2020). Consideration of the local correlation of learning behaviors to predict dropouts from MOOCs. Tsinghua Science and Technology, 25(3), 336–347.

[17] Lin, S. Y., Wu, C. M., Chen, S. L., & Lin, T. L. (2020). Continuous facial emotion recognition methods are based on deep learning of aca-demic emotions. Sensors and Materials, 32(10), 3243–3259.

[18] Farissi, A., Dahlan, H. M., & Samsuryadi. (2020). Genetic algorithm-based feature selection with ensemble methods for student aca-demic performance prediction. Journal of Physics: Conference Series, 1500(1), 012014.

[19] Turabieh, H., Azwari, S. A., Rokaya, M., Alosaimi, W., Alharbi, A., Alhakami, W., et al. (2021). Enhanced Harris hawk's optimization as a feature selection for the prediction of student performance. Computing, 103, 1417–1438.

[20] Ma, H. B., Yang, S. Y., Feng, D. Z., & Jiao, L. C. (2021). Progressive mimic learning: A new perspective to train lightweight CNN model-els. Neurocomputing, 456, 220–231.

[21] Nabil, A., Seyam, M., & Abou-Elfetouh, A. (2021). Prediction of students' academic performance based on courses' grades using deep neural networks. IEEE Access, 9, 140731–140746.

[22] Gao, L., Zhao, Z., Li, C., Zhao, J., & Zeng, Q. (2022). Deep cognitive diagnosis model for predicting students' performance. Future Generation Computer Systems, 126, 252–262.

[23] Mishra, P., Biancolillo, A., Roger, J. M., Marini, F., & Rutledge, D. N. (2020). New data preprocessing trends based on an ensemble of multiple preprocessing techniques. TrAC Trends in Analytical Chemistry, 132, 116045.

[24] Zhao, S., Zhou, D., Wang, H., Chen, D., & Yu, L. (2025). Enhancing student academic success prediction through ensemble learning and image-based behavioral data transformation. Applied Sciences, 15(3), 1231.

[25] Hasan, R., Palaniappan, S., Mahmood, S., Abbas, A., & Sarker, K. U. (2021). Dataset of students' performance using student infor-mation system, Moodle and the mobile application "eDify". Data, 6(110), 1–7.

[26] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.

[27] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2023). Attention is all you need. arXiv preprint arXiv:1706.03762v7 [cs.CL]. https://doi.org/10.48550/arXiv.1706.03762