JSSE JOURNAL OF SCIENCE & SCIENCE EDUCATION Vol. 7 No. 1 (Jan. – Jun. 2024): 21 - 36
Faculty of Science, Ubon Ratchathani University http://doi.org/10.14456/jsse.2024.3

Research Article

Enhancing Prediction of Student Learning Success in Higher Education using Deep Learning

Worayoot Wongnin[1] and Chayaporn Kaensar[1, *]

[1]Department of Mathematics, Statistics and Computer, Faculty of Science, Ubon Ratchathani University

**Email: chayaporn.k@ubu.ac.th*

Received < 30 March 2024 >; Revised < 22 April 2024 >; Accepted < 28 April 2024 >

Abstract

Tackling the global challenge of student attrition and underperformance that widely affect educational quality is a huge challenge. This is a very significant issue in Thailand. Thus, the current research analyze data from 51,106 students and 11 faculties of Ubon Ratchathani University in Thailand for the 2011-2021 academic years. This was done to develop enhanced predictions of academic success utilizing two deep learning methods (Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN)), each fine-tuned for maximum accuracy. The predictive models were scored using the Confusion Metrics and AUC-ROC. The experimental results show that, our research stands out for its robust predictive power, achieving accuracies between 81% to 98%. Notably, RNN achieved high accuracy in all faculties. Additionally, leveraging Django, Python, and PostgreSQL, we propose a research methodology and develop a web application that operationalizes our findings to stakeholders for effectiveness. It has the capability of providing a group of users with a practical tool enabling individual students to forecast academic outcomes, teachers to identify at-risk students early, and faculty as well as, university staff to support informed decision-making thereby improving educational strategies and outcomes. This comprehensive methodology illuminates effective predictive models and system tools driving academic achievement in our university and other higher education institutes.

Keywords: Student performance prediction, Deep learning, Web application, Graduation rate

บทความวิจัย

การเพิ่มประสิทธิภาพการทำนายความสำเร็จของผู้เรียนในระดับอุดมศึกษา ด้วยเทคนิคการเรียนรู้เชิงลึก

วรยุทธ วงศ์นิล[1] และชยาพร แก่นสาร์ [1,*]

[1]ภาควิชาคณิตศาสตร์ สถิติ และคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยอุบลราชธานี

**Email: chayaporn.k@ubu.ac.th*

รับบทความ: 30 มีนาคม 2567 แก้ไขบทความ: 22 เมษายน 2567 ยอมรับตีพิมพ์: 28 เมษายน 2567

บทคัดย่อ

การจัดการปัญหาการตกออกและระดับผลการเรียนที่ต่ำกว่าเกณฑ์ของนักศึกษาในมหาวิทยาลัย พบว่าเป็นเรื่องสำคัญและมีความท้าทายต่อคุณภาพการศึกษาทั่วโลกและในประเทศไทยเป็นอย่างมาก ดังนั้นในงานวิจัยนี้จึงมีวัตถุประสงค์ที่ต้องการวิเคราะห์ข้อมูลของนักศึกษาในมหาวิทยาลัยอุบลราชธานี ระหว่างปีการศึกษา 2554-2564 จำนวน 11 คณะ รวมทั้งสิ้น 51,106 คน เพื่อนำมาเปรียบเทียบและสร้างแบบจำลองโมเดลสำหรับพยากรณ์ความสำเร็จของนักศึกษาด้วยเทคนิคการเรียนรู้เชิงลึกที่มีประสิทธิภาพ 2 เทคนิคคือ โครงข่ายประสาทเทียมแบบคอนโวลูชัน และโครงข่ายประสาทเทียมแบบวนซ้ำ ซึ่งได้มีการปรับแต่งพารามิเตอร์ที่เหมาะสมสำหรับอัลกอริทึมเพื่อให้มีค่าความถูกต้องสูงที่สุดร่วมด้วย นอกจากนี้งานวิจัยยังต้องการนำเสนอโครงสร้างระบบที่สามารถนำผลลัพธ์แบบจำลองโมเดลที่ปรับแต่งและให้ประสิทธิภาพที่สูงไปแทนค่าความรู้ในระบบแอปพลิเคชัน ซึ่งพัฒนาด้วย ดีจังโก้ ไพทอน และโพสต์เกรสคิวเอล ให้สามารถทำงานร่วมกันได้ โดยผลการทดลองพบว่าในส่วนการวัดประสิทธิภาพโมเดลด้วย Confusion Matrix และ AUC-ROC พบว่าเทคนิคโครงข่ายประสาทเทียมแบบวนซ้ำ มีค่าความถูกต้องสูงที่สุดในทุกคณะ โดยมีค่าความถูกต้องระหว่าง 81%-98% ในขณะที่เว็บแอปพลิเคชันมีฟังก์ชันที่ทำงานได้ถูกต้อง ซึ่งประโยชน์ของงานวิจัยนี้จะได้แบบจำลองโมเดลสำหรับพยากรณ์ความสำเร็จของนักศึกษาและเว็บแอปพลิเคชันที่ช่วยผู้ใช้งานในระดับต่าง ๆ ได้ โดยผู้เรียนสามารถใช้พยากรณ์ความสำเร็จแบบรายบุคคล ในขณะที่ผู้สอนสามารถเข้าถึงข้อมูลผู้เรียนที่มีความเสี่ยงที่อาจจะตกออกได้ล่วงหน้า คณะหรือมหาวิทยาลัยมีระบบรายงานสารสนเทศที่สามารถนำไปใช้วางแผนหรือตัดสินใจเพื่อพัฒนาคุณภาพการศึกษาให้มีประสิทธิภาพมากขึ้น

คำสำคัญ: การทำนายประสิทธิภาพการเรียนรู้ของผู้เรียน เทคนิคการเรียนรู้เชิงลึก เว็บแอปพลิเคชัน อัตราการสำเร็จการศึกษา

Introduction

Currently, many universities are faced with poor academic performance, delayed graduation, early dropouts, or even ceased studies. This causes problems for students, their families, and institutes, which are widely perceived as time and resources wasted. Even though many universities have invested in the education system and provided well-equipped facilities for digital advancement, student performance and success are still critical. (Mangaroska and Giannakos, 2018)

Similarly, Ubon Ratchathani University (UBU) was founded in 1988, approving establishment of 11 faculty in the fields of (1) Medicine, (2) Nursing, (3) Pharmaceutical Sciences, (4) Agriculture, (5) Engineering, (6) Science, (7) Architecture and Applied Art, (8) Business and Management (9) Law, (10) Liberal Arts, and (11) Political Science. Most all programs have a four year duration, while medicine and pharmaceutical sciences are six-year undergraduate degrees. However, even though our university works to achieve a high number of graduated students with good academic results, student success has declined and many are vastly underperforming. As reported in the UBU REG system (REG UBU, 2010) during the 2010-2021 academic years, about 17% of the students overstay their program duration, while the remaining students completed their degree within the expected timeframe, but 29% had low scores.

To tackle this problem, preparing and graduating students with good performance are crucial tasks. However, current approaches, such as preparing new students with skills in digital literacy, problem-solving and communication to enhance their success, may not be sufficient in our universities. This is due to a lack of useful data and limited understanding of student performance. In particular, there is a lack of online decision-making tools which help students, faculties or even institute understand student's learning performances and their risks at an early stage. Therefore, focusing on the development of an early prediction or warning system for forecasting student performance is essential and considered a key factor in academic improvement.

Currently, although many systems have been proposed as tools to predict student performance and have achieved high accuracy (Wibowo, Andreswari and Hasibuan, 2018; Badal and Sungkur, 2023; Kaensar and Wongnin, 2023; Park and Seong, 2021), some limitations remain unaddressed. For example, there is limited comparability of learning model performance, especially with deep learning. Additionally, there is a lack of tunable models with appropriate parameters and optimizers. In some cases, few diverse input data are chosen for prediction with insufficient quantification of each factor's impact. Furthermore, there is no web implementation of such systems.

To address these challenges, this study aims to develop alternative and new-day wise approach using deep learning model, which are flexible and capable of handling vast amounts of data for prediction (Bhushan *et al*., 2023; Streib *et al*., 2020). Two deep learning models (Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN)) with fine-tuned parameters are employed to ensure analysis accuracy and prediction of student success based on UBU's student data. More specifically, it was implemented as a web-based tool with integration of Django, Python, and PostgreSQL to support prediction processes and make useful decisions for the university.

To achieve these goals, the main research questions (RQ) are:

1) Which of the deep learning models (CNN, RNN) with fine-tuned parameters can best predict student performance of each faculty in UBU?
2) How will the proposed framework be deployed to generate predictions and report these results via a web system?

To answer these questions, this study collected and used a large dataset comprise of 51,106 student records with 27 variables, covering three datasets, student demographics, school data, and university data, from the 2010-2021 academic years. To generate output and achieve more accurate predictions, the tasks of

preprocessing data, training the models with fine-tuned parameters, evaluating the models using five-fold cross-validation, and developing a web system were done.

The main primary contributions of this study are development of methods to make high-accuracy predictions of successful UBU students in each faculty. These results can be used to make predictions and identify at-risk students early, plan appropriate interventions, analyze reported data. Furthermore, this is a practical concept and its results can be implemented through web applications designed to enhance learning quality at a university and other higher education institutes by supporting decision-making.

Research objectives

1. To construct, compare and evaluate student success prediction model using deep learning of each faculty in UBU.
2. To propose a research methodology and develop a web application that operationalizes our findings to stakeholders for effectiveness

Literature review

Student prediction approach in deep learning and their applications

Deep learning is important for classification and prediction in a variety of domains. These approaches were applied in many studies with remarkable results, especially in education (Sultana, Rani and Farquad, 2019; Bhushan *et al*., 2023). By analyzing LMS logged data, (Sultana, Rani and Farquad, 2019) employed 11 features to predict student success, using data collected from 1,100 records with student interaction, activities and student learning in an LMS. Techniques such as a Deep Neural Network (DNN) and some machine learning with WEKA and Rapid Miner Tools were applied. Decision Tree and DNN gave the highest prediction accuracies, 99.81% and 99.45% respectively. Similarly, (Park and Seong, 2021) conducted similar experiments with deep learning and other machine learnings but used a larger dataset with approximately 1,480,275 records of 98,685 students who accessed online coursework between 2012-2019. The result showed that Random Forest and DNN can predict early dropout of online learner with an accuracy of 96% and 85%. Likewise, (Vijayalakshmi and Venkatachalapathy, 2019) developed a prediction model to forecast post-graduate student performance employing various techniques using R Programming. The data obtained from Kaggle contained 480 instances, with 16 demographic attributes (gender, nationality, place of birth, parents), academic data (school grades, days of absence), behavior (raised hands, visited and view materials) and parental participation (answering, posting). DNN showed high accuracy, 84%. Moreover, estimation of parameters using deep learning and machine learning techniques to improve prediction accuracy have gain recent attention. Such as (Mohammad *et al*., 2023) explored the effectiveness of using deep learning and other algorithms to more precisely predict student performance. CNN with 7, 9 and 11 epochs yielded the best accuracy, 88.89%.

Improvements continued when much research also focuses on development of web tools to enhance outcomes. For example, (Roy *et al*., 2022) developed a web application with a DNN model to predict CGPA and recommend study plans. Although it provides high accuracy, 92.94%, only a single academic term result is considered. To provide more student data, the study of (Wibowo, Andreswari and Hasibuan, 2018) added school grades, parent background, number of credits and student activity data to develop a decision support system (DSS) with C4.5 decision tree to predict student graduation time, while (Inamdar *et al*., 2022) analyzed previous GPAs and use Linear Regression to develop a powerful decision tool with a Django framework for cutoff predictions in a college admission process. With limited input data, accuracy and even system functionality are still improved. (Badal and Sungkur, 2023) enhanced a system by focusing on three functions, predicting student performance, identifying factors and proposing a framework as a decision support tool for grade prediction. Deep learning, machine learning and 34 features (*e.g.,* demographic, certificates obtained, experience and activities data) were employed. Although at-risk student assessment and early

prediction by the web system were strengths, utilization of default parameters and some factors might affect the results are less considered.

Summary of existing research and motivation

Although most of the existing studies may have high accuracy, these works suffer from a limited input data, a lack of model tuning and undeployment of the model into a system. Therefore, this study employs five main components: (1) integrating more powerful deep learning methods, (2) tuning appropriated parameter settings, (3) analyzing a wider range of datasets (*e.g.,* student demographics, school data, and university data), (4) developing web-based prediction system, and (5) investigating deep learning for prediction which has emerged as a new research area of machine learning. The description of our methodology will be described in the next section.

Research methodology

In this section, a proposed methodology based on two deep learning models (CNN, RNN) are built. The models and their settings were trained and tuned. Then, model performance was evaluated. After that, the fine-tuned model is integrated with a web tool to predict and report results. The steps in building the system are depicted in Figure 1.

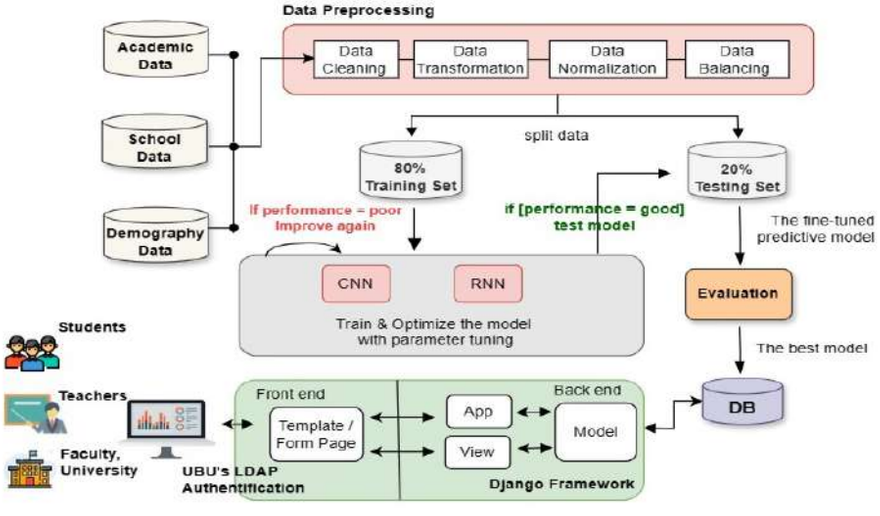

Figure 1 Research methodology

Data collection

Real-world data acquisition from UBU spanned from 2011 to 2021, combining three distinct sources (demographic, school data and university data). The input data are divided into seven demographic variables, fourteen school data variables, and six university variables. From this, we not only consolidated all the information into a single table while considering the data relationships, but also categorized it into three groups based on the data type. These categories include categorical data (such as entry type, parental status, parent occupation, and their incomes) and numerical data (including ONET score and high school grades). These descriptions are depicted in Table 1.

To ensure the accuracy and reliability of our predictive models, the dataset required careful preprocessing. Initially, the classes were significantly imbalanced, with 12,955 instances labeled as '0' and 38,151 labeled as '1', resulting in a class ratio of approximately 1:3. To correct this and mitigate model bias towards the majority class, we implemented resampling techniques to equalize the distribution between

classes. This crucial step enhances the robustness and generalizability of the deep learning algorithms applied in our study.

Table 1 The description of features in UBU

Type	ID	Attribute	Description
Demographic Data	1	Sex	Ex. male, female
	2-4	Parent detail	Ex. parental status, income and their career
	5	School detail	Ex. is school in town?
	6-7	Home detail	Ex. home location and is home in town?
School Data	8	Entry GPA	Cumulative GPA in high school
	9-13	Thailand's Ordinary National Education Test (ONET_Subject)	ONET score for Thai, Social, English, Mathematics and Science
	14-21	High school grade (scho_gpax01-08)	01 - Thai, 02 - Mathematics, 03 - Science, 04 - Social, 05 - Health and Physical education, 06 - Art, 07 - Technology and Career and 08 - English
University Data	22-23	Entry detail	Ex. entry year, entry type
	24-25	Field of studying	Ex. program name, major
	26-27	Grade of first year	GPA of term1, GPA of term2
Target		Graduation status	1 - Student who complete the degree 0 - Student who fail or ungraduated

Data preprocessing

Effective data preprocessing is crucial for deep learning analysis. We utilized Conda for managing Python dependencies and installing essential libraries such as Keras and TensorFlow. The Anaconda platform streamlined this process, integrating necessary tools for data science. We employed a Jupyter Notebook for interactive data handling and visualization. This produced a versatile web application that allows for dynamic coding as well as presentation of data and analysis workflows. Key preprocessing tasks were done as follows.

1. Data cleaning In the initial stages of our analytical process, we undertook a comprehensive data cleaning and preprocessing routine. This fundamental step encompassed a variety of tasks aimed at refining the dataset for subsequent analysis. Key among these tasks was standardization of column names to a uniform lowercase format for consistency. In this step, an automated method cleans the student dataset using a Python script. It consists of processes incorporating data, managing missing data using interpolation methods, handling data anomalies, sklearn one-hot encoding and label encoding, and finally standardizing the data. Standardization is necessary for data compatibility with deep learning algorithms that require numerical input. Categorical variables, such as those indicating program enrolment or faculty, were transformed into numerical values through label encoding. Abnormal data are recognized using threshold and clustering methods to eliminate faulty data from the data set. Missing values can be handled by removing those records that lacked complete information, thereby cleaning the dataset for more accurate model training.

2. Data transformation This study employed a label encoding technique to transform raw data into a format amenable to analysis. This process can handle abnormal values and outliers in the dataset, reduce the influence of abnormal values on the prediction model, and improve the robustness of the model. It was

necessary to convert categorical features into numerical data, thereby ensuring compatibility with sklearn functions (*e.g.,* with methods such as transform (array of trained_data)).

3. Data normalization Following data transformation, normalization was implemented to mitigate the large variances between maximum and minimum values of features. This normalization scales the feature values to a decimal range from 0 to 1, contingent on their respective maximum and minimum values. Such a step is fundamental for deep neural networks, aiding in more consistent and expedited model processing (Bhushan *et al*., 2023; Mohammad *et al*., 2023) From this context, the Min-Max Scaler technique was employed. In this equation, 'valueold' denotes the original value of the feature, while 'value' represents the normalized value. The Min-Max Scaler formula is (Fullan, 2018):

$$value = \frac{value_{old} - min_A}{max_A - min_A} * (new\_max_A - new\_min_A) + new\_min_A \quad (1)$$

This formula was used to adjust the values of each feature to a normalized scale, thereby enhancing the homogeneity of the dataset and optimizing it for effective analysis.

Deep learning model training

This section presents the methodology for training deep learning models, following extensive data preprocessing. Numpy libraries were used for loading datasets, while Keras and TensorFlow served as the backend for model training, which proceeded in structured stages.

1. Data splitting In this step, we implemented a five-fold cross-validation strategy for dividing the data and reducing overfitting, ensuring a thorough evaluation of model performance. The dataset was segmented into five equal parts, with each serving as a validation set (20% of dataset) once during the process, and the remaining data combined to create a training set (80% of the dataset). This cyclical validation guarantees that every data point contributes to both training and validation, thereby enhancing the model's robustness.

2. Hyperparameter values setting We explored a range of parameter combinations to determine the most effective settings for our models. To fine-tune parameters for the CNN and RNN models, variables were set to create 108 and 36 patterns using different values and some default settings for each faculty. The possible hyperparameters for each classifier are shown in Table 2.

Table 2 Experimental setup showing using various classifier values.

Classifier	Hyperparameter List
CNN	filters=[32,64,128], kernel_size=[3,5,7], strides=[1,2], padding=['valid','same'], activation=['relu', 'tanh', 'sigmoid']
RNN	epochs=[10, 20, 50], batch_size=[16, 32, 64], activation=['relu', 'tanh', 'sigmoid', 'softmax']

3. Construct and fit the model We explored a range of parameter combinations by tuning parameters for each model to determine their most effective settings. Hence, variables were set to create CNN and RNN with 108 and 36 patterns using different values and some default values for each faculty. After that, we applied seven algorithms that were suitable for the nature of our models. The display of code example for building the deep learning model is presented in Figure 2.

```
parameters = {'filter_p':[32,64,128], 'kernel_p':[3,5,7], 'stride_p': [1,2],
              'padding_p': ['valid','same'],'activation_p':['relu','tanh','sigmoid'] }
paras = [len(e) for e in parameters.values()]
size = 1
for s in paras:
    size = size*s
model_abb = 'CNN'       # Convolutional Neural Network
for fac in fac_id:
  result = pd.DataFrame(columns=['fac','model', 'parameters','accuracy','precision','recall','f1'])
  for ft in parameters['filter_p']:
    for ks in parameters['kernel_p']:
      for std in parameters['stride_p']:
        for pad in parameters['padding_p']:
          for act in parameters['activation_p']:
            kf = KFold(n_splits=5, random_state=16, shuffle=True)       # K-fold cross-validation (k=5)
            for train_ind, test_ind in kf.split(dataset[fac]['x']):
              x_train, x_test, y_train, y_test = dataset[fac]['x'].iloc[train_ind], \
                                                 dataset[fac]['x'].iloc[test_ind], \
                                                 dataset[fac]['y'].iloc[train_ind]['target'], \
                                                 dataset[fac]['y'].iloc[test_ind]['target']
              model_str = f'{filters=},{kernel_size=},{strides=},{padding=},{activation=}'
              try:
                #Step1. Preparing data
                scaler = MinMaxScaler()                          # Normalize input features
                X_train_scaled = scaler.fit_transform(x_train)  # Fit,transform input data&converts data
                X_test_scaled = scaler.transform(x_test)        # Apply learned transformation to test data
                X_train_reshaped = X_train_scaled.reshape((X_train_scaled.shape[0], X_train_scaled.shape[1], 1))
                X_test_reshaped = X_test_scaled.reshape((X_test_scaled.shape[0], X_test_scaled.shape[1], 1))

                #Step2. Defining and Fitting model
                model = Sequential()
                model.add(Conv1D(filters=ft, kernel_size=ks, activation=act, input_shape=(X_train_reshaped.shape[1], 1)))
                model.add(MaxPooling1D(pool_size=2))       # For 1D temporal data.
                model.add(Flatten())                       # Collapsed into one dimension.
                model.add(Dense(64, activation='relu'))    # Apply rectified linear unit activation function.
                model.add(Dense(1, activation='sigmoid'))  # Apply sigmoid for binary classification
                model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])   #Compile model
                model.fit(X_train_reshaped, y_train, epochs=10, validation_data=(X_test_reshaped, y_test)) #Fit model

                #3. *** Predict and Visualize results
                y_pred = model.predict(X_test_reshaped)     # Evaluate model
                y_pred = np.round(y_pred)                   # Convert probability to binary prediction (0|1)
                accuracy = accuracy_score(y_test, y_pred)   # Get result score of accuracy
                precision = precision_score(y_test, y_pred) # Get result score of precision
                recall = recall_score(y_test, y_pred)       # Get result score of recall
                f1 = f1_score(y_test, y_pred)               # Get result score of f1-measure
                result.loc[len(result)] = [fac, model_abb, model_str, accuracy, precision, recall, f1]  #Collect result
              except Exception as E:
                print(E)
              break
```

Figure 2 Code from the implementation of predictions using CNN1D with Keras of Python

Model building generally is a three-step procedure as follows:

1) **Preparing the data:** This stage loads the dataset and check the data.
2) **Defining and fitting model**: This stage defines, compiles the model with an optimizer and fits the model to training data.
3) **Predicting and visualizing the result**: This stage out puts predictions and evaluates model performance.

As illustrated in Figure 2, a CNN applied to a 1D Convolutional network reshapes the input data to perform predictions. Precisely, its starts with preparing the data by adjusting its dimensionality with the "reshape()" method, creating indices to split data into training and testing sets [xtrain, xtest, ytrain, ytest] with "split()". Next, it defines the model by integrating Flatten(), Dense() layers and compiles it with optimizers using "compile()". Last, prediction are made with the trained model using "predict()". However, model performance must be evaluated predictions visualizing in a plot, which is described in the next section.

Evaluation

The evaluation phase is to measure model performance**.** Post-training, models with varied parameters were evaluated to ascertain differential performance across several optimizers. For a comprehensive assessment, key metrics such as accuracy, precision, recall, and F1-score, were employed as the principal indicators. Additionally, the Area Under the Receiver Operating Characteristic curve (AUC-ROC)

score was utilized as a metric to gauge the model's discriminative capability. Validation of model performance utilized the remaining 20% of the dataset, which was subject to a five-fold cross validation procedure. The mathematical expressions for each metric are provided as follows (Fullan, 2018):

1. **Accuracy** measures the proportion of predictions that the model classified correctly, defined as:

$$Accuracy = \frac{True\ Positive + True\ Negative}{Total\ Examples} \quad (2)$$

2. **Precision** indicates the frequency of actual positives among positive predictions, which is useful in minimizing the number of false positives, defined as:

$$Precision = \frac{True\ Positive}{(True\ Positive + False\ Positive)} \quad (3)$$

3. **Recall** or the true positive rate, represents the model's capacity to identify all relevant instances of true positive values, defined as:

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (4)$$

4. **F1-score**, the harmonic mean of precision and recall. This parameter provides a balance between the precision and recall metrics, defined as:

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (5)$$

5. **AUC-ROC** is a method of validation and visualization the performance of a classification model at different threshold values. ROC curves and AUC provide a discriminating form to support model evaluation.

Developing a web-based application

To use a trained model and implement the system, a web application was developed that contained front-end and back-end workflow, developed using Django, Python, and PostgreSQL to forecast and report the results. This system was designed based on Django (Model-View-Template) and run within the Celery framework for task management.

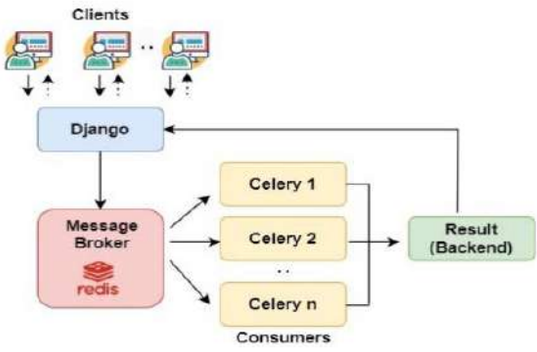

Figure 3 Django and Celery process flow

To begin in Figure 3, data required for prediction is collected from clients for entry in Django templates as a front-end on the client side. Next, it is hosted through a Django Framework on a back-end system where the stored version of deep learning is integrated. To accomplish this task, Django sends data to Redis Message Broker and invokes Celery to consume it. When the task finishes, Celery saves results and updates the task status. Simultaneously, Django checks this status in parallel and directs results to the front-end to display and report results pages via web sockets and the Django channel library. The program code shows a prediction function in loading, transforming, and predicting using Django and Python to conduct an analysis, which is illustrated in Figure 4.

```
import pickle       # For model deployment
import numpy as np  # For data management

with open('MODEL.pkl', 'rb') as model_file:
    loaded_model = pickle.load(model_file)      # Load model

with open('SCALER.pkl', 'rb') as scaler_file:
    scaler = pickle.load(scaler_file)           # Load scaler

def Prediction(faulty_id, sex_enc, par_stat_enc, is_sch_intown_enc, home_loc_enc, is_home_intown_enc,
               entrytype_enc, par_inc_enc,par_occ_enc, entry_gpax, on_thai, on_soc, on_eng, on_math,
                on_sci, GPAX01, GPAX02, GPAX03, GPAX04, GPAX05, GPAX06, GPAX07, GPAX08,entry_year,
                pgm_enc, major_enc, gpa_term1_enc, gpa_term2_enc): # Define function and their parameters

  input_data = [sex_enc, par_stat_enc, is_sch_intown_enc, home_loc_enc, is_home_intown_enc, entrytype_enc,
                par_inc_enc, par_occ_enc, entry_gpax, on_thai, on_soc, on_eng, on_math, on_sci,
                GPAX01, GPAX02, GPAX03, GPAX04, GPAX05, GPAX06, GPAX07, GPAX08,
                entry_year, pgm_enc, major_enc, gpa_term1_enc, gpa_term2_enc] # used 27 input data

  input_data = np.array(input_data).reshape(1, -1)  # Convert input to 2D array
  scaler_fac = scaler[faulty_id]                    # Perform scaling on selected feature
  X_new_scaled = scaler_fac.transform(input_data)   # Apply the learned transformation to test data

  model_faculty = loaded_model[faulty_id]           # Load the model

  prediction = model_faculty.predict(input_data)    # Predict the labels of the data value

  result = "Fail" if prediciton!=1 else "Pass"      # Prediction=1 (Pass), others=Fail
  return result                                     # Return output

```

Figure 4 Code snippet for predicting output

Experimental results and discussion

Two sets of experimental results are discussed to address the research questions of this study.

Evaluation of model prediction results

To address RQ1, we compared and evaluated two deep learning approaches to predict whether students would graduate from a program (achieve a GPAX of 2.00 or more) for each faculty using the UBU student dataset. Furthermore, we tuned parameter settings and reduced the imbalanced information using Synthetic Minority Over-sampling Technique (SMOTE) to enhance performance.

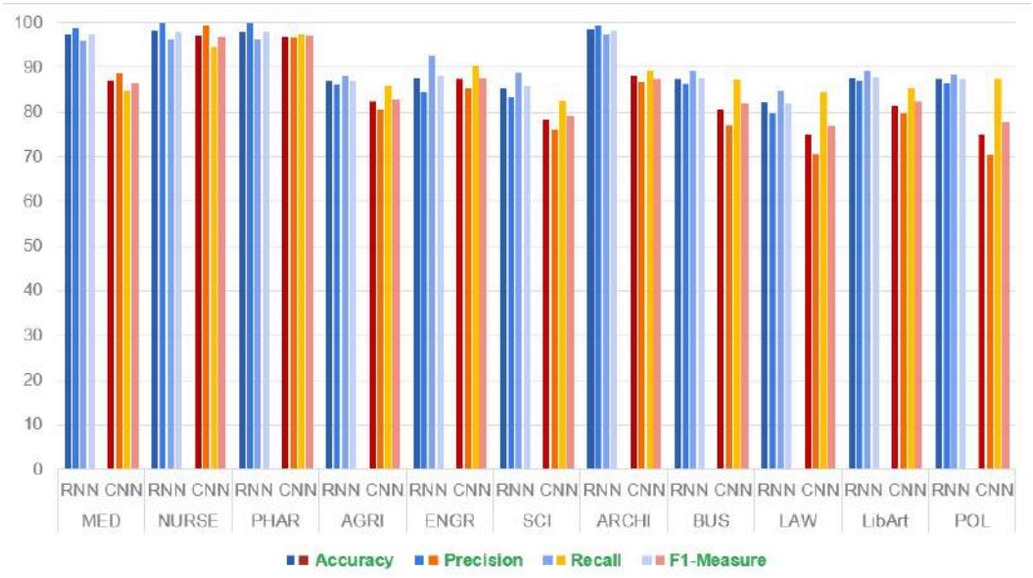

Figure 5 Performance results of 11 faculties with a parameter tuning algorithm

As shown in Figure 5, student outcome prediction is very high for each faculty with accuracy ranging from 81% to 98%. Hence, the outcomes showed that the RNN model produced the most accurate predictions in all faculties. The model prediction for the Faculty of Architecture and Applied art produced an F1-measure of 98.39% that reported the highest accuracy. However, the model for evaluation of Law student might be less than the others, but in fact, the accuracy of 81.97% is still high. The list of all 11 faculties with their fine-tuned predictive models is shown in Table 3.

As such, we concluded that RNN works best, which is accordance with the research of (Vijayalakshmi and Venkatachalapathy, 2019 and Korchi *et al*., 2023). It is proved that the RNN provided better accuracy than the other methods such as Regression, Naive Bayes.

Table 3 Performance evaluation results applied to 11 faculties

Faculty	Fine-tuned Predictive Model	Accuracy (%)		Precision (%)		Recall (%)		F1-measure (%)	
		RNN	CNN	RNN	CNN	RNN	CNN	RNN	CNN
MED	RNN	97.47	87.10	98.88	88.71	96.07	84.48	97.44	86.49
NURSE	RNN	98.28	97.12	100	99.51	96.33	94.55	98.12	96.88
PHAR	RNN	98.16	97.01	100	96.78	96.34	97.29	98.11	96.99
AGRI	RNN	86.88	82.34	86.07	80.58	88.14	85.61	87.05	82.83
ENGR	RNN	87.67	87.40	84.47	85.45	92.63	90.40	88.21	87.68
SCI	RNN	85.44	78.28	83.37	76.22	88.75	82.49	85.86	79.07
ARCHI	RNN	98.62	88.22	99.46	86.83	97.41	89.14	98.39	87.33
BUS	RNN	87.53	80.59	86.30	77.14	89.34	87.14	87.75	81.77
LAW	RNN	82.07	75.00	79.77	70.61	84.65	84.31	81.97	76.71
LIB-ART	RNN	87.79	81.46	86.98	79.74	89.29	85.35	87.95	82.21
POL	RNN	87.38	75.05	86.57	70.28	88.37	87.26	87.44	77.74

In Figure 6, ROC curves of all compared algorithms are used to assess the predictive performance of models. It displays a comparative analysis a model across various academic disciplines, with each curve demonstrating the model's capability to balance the true positive rate (TPR) against the false positive rate (FPR) at various thresholds, as indicated by the respective Area Under the Curve (AUC) metrics. The AUC values, ranging from 0.88 to 1.00, reveal significant variation in model performance, with the model for the health care and medical fields achieving the highest discriminative accuracy.

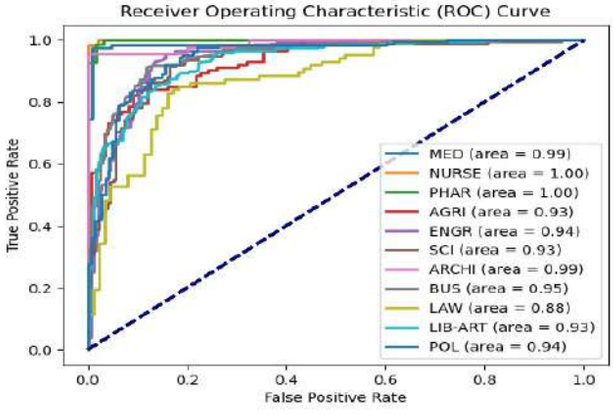

Figure 6 Representation of ROC curve of RNN algorithms

Furthermore, we fine-tuned deep learning algorithms to evaluate the influence of hyperparameters on predicting student success. Iterative experimentation across various settings focused on the most effective hyperparameter combinations applied to faculty-specific datasets, which are given in Table 4. Specifically, the RNN model was tuned by varying three optimal parameters (*i.e.,* epochs, batch_size and activation function) to obtain a better result. As a result, we found that setting an optimal epoch=50, defining a batch size in [16, 32, 64] and using a 'relu' or 'tanh' activation function could provide high accuracy. However, even though a small filter size and large batch size can improve the model performance, they might increase the computational cost of the model.

Table 4 Optimized Hyperparameters of all the best performing classifiers.

Faculty	Classifier	Optimized Parameter Settings
MED	RNN	epochs=50, batch_size=16, activation='relu'
NURSE	RNN	epochs=50, batch_size=16, activation='tanh'
PHAR	RNN	epochs=50, batch_size=32, activation='tanh'
AGRI	RNN	epochs=50, batch_size=16, activation='relu'
ENGR	RNN	epochs=50, batch_size=32, activation='relu'
SCI	RNN	epochs=50, batch_size=16, activation='tanh'
ARCHI	RNN	epochs=50, batch_size=32, activation='relu'
BUS	RNN	epochs=50, batch_size=16, activation='relu'
LAW	RNN	epochs=50, batch_size=64, activation='relu'
LIB-ART	RNN	epochs=50, batch_size=16, activation='tanh'
POL	RNN	epochs=50, batch_size=16, activation='tanh'

Web-based Application in prediction and report panel

To answer RQ2, we developed a web-based prediction and reporting system that can provide a mechanism to analyze student performance and provide summary reports using a fine-tuned predictive model. The prediction and report panel are shown in Figure 7.

Figure 7 Prediction results dashboard for the Faculty of Science, which include dashboard showing a comparison of success prediction for each faculty.

The system consists of the following two modules. First, it provides a brief summary of student backgrounds, grading, retention rates and especially prediction in completion rate for the whole university as well as the individual faculties. For example, Figure 7 shows a report of Faculty of Science student percentages for the 2022 academic year indicating the likelihood that they will complete their degrees used an RNN fine-tuned model. Moreover, it also shows the percentage of students who will likely succeed in their studies across all the faculties.

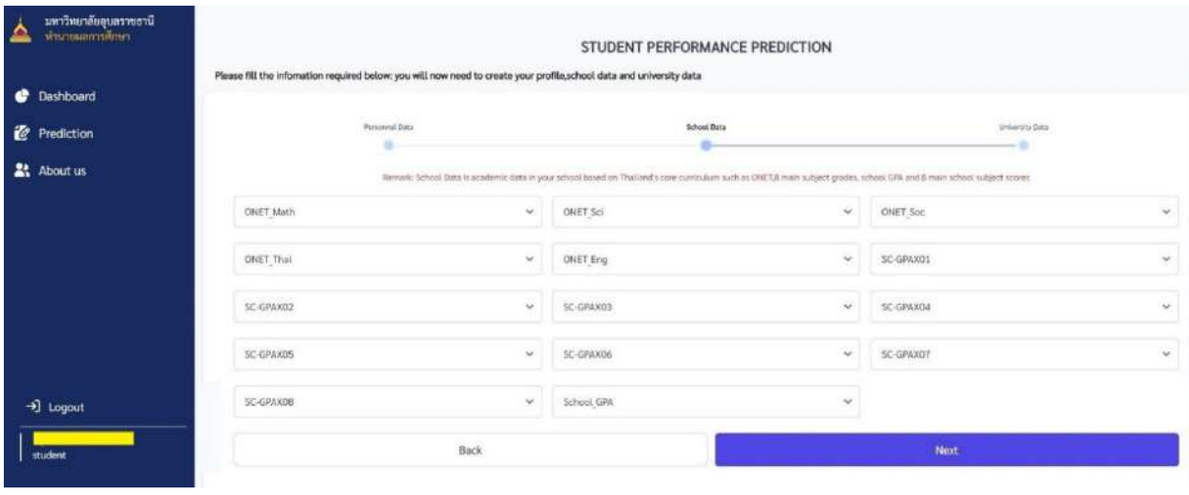

Figure 8 Input form for individual predictions

The second part was done to demonstrate the impacts of various factors on academic performance for specified faculties. As such, faculty and university administrators can better understand student success in a data-driven approach for decision making. For example, when a user submitted their form in Figure 8, prediction results of a fine-tuned predictive model are displayed indicating whether they are likely to complete their studies. Histograms highlight important features affecting student performance in Figure 9.

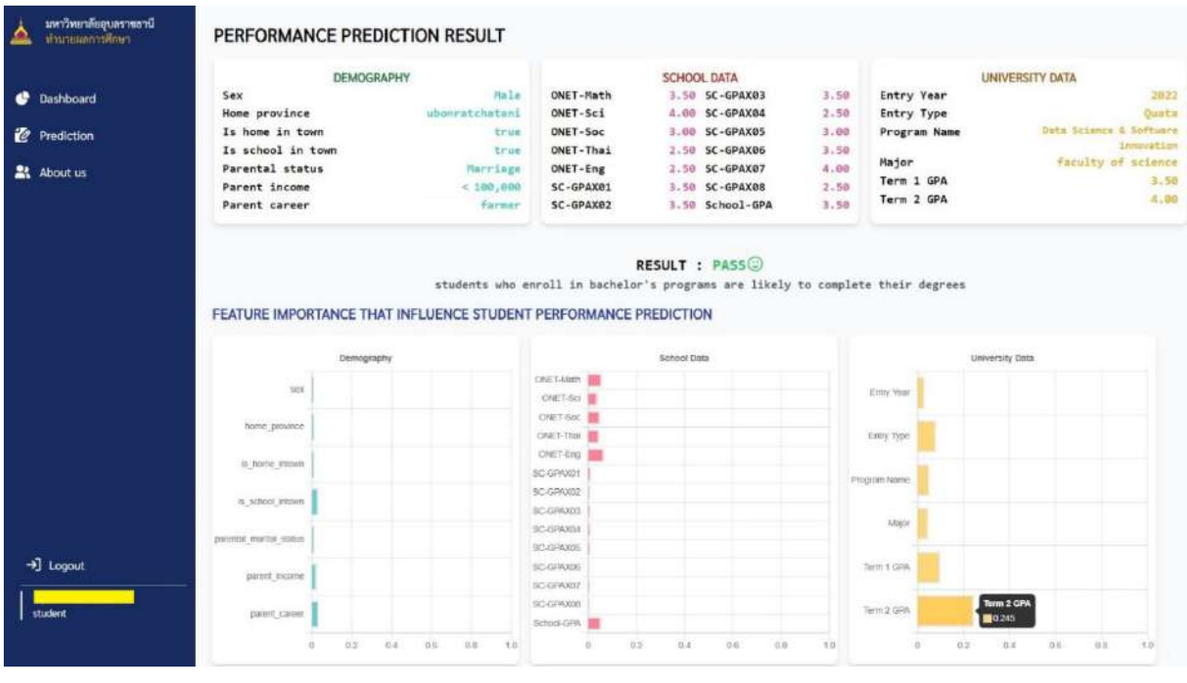

Figure 9 Form reporting individual student prediction results

Successful testing of HTTP and web sockets shows the final application is working correctly, and the database can support deep learning algorithms. This establishes interoperability between the subsystems, and that the fine-tuned algorithm is working well to produce results.

Furthermore, we conducted the time measurement in the process of the prediction process from when a client made a request to the system until the task was completed and the results were displayed to

users. These measurements are repeated 35 times to reach the result. The result of execution times to complete a prediction were captured and used within application developed in Python, which are depicted in Table 5.

Table 5 Performance of prediction speed in web application

Minimum time (sec)	Maximum time (sec)	Average time (sec)
0.0649	0.1849	0.1125

Conclusions and future work

This study is a thorough analysis of a dataset obtained from Ubon Ratchathani University. The dataset consists of 27 attributes, covering 11 faculties, and includes data on 51,106 students over the period of 2011 to 2021. Utilizing deep learning algorithms, we unveil patterns in student performance, with a particular focus on the predictive capabilities of specific models in various academic fields. This comprehensive dataset enables us to improve predictive models and expand their usefulness in educational environments. Moreover, development of a web application was done as a prediction and decision-making tool by implementing the optimal model with Scikit-Learn of Python, PostgreSQL, and a Django framework.

In summary, addressing student success can be done using the approach of this study to enhance educational quality, identify at-risk students early, and in decision-making, Furthermore, it would be beneficial to integrate this approach into other relevant educational platforms.

In future research, the study intends to include additional factors such as Learning Management System (LMS) activities and social network datasets. This will expand the range of student data analysis to include correlation and prediction. This expansion aims to improve the predictive models by utilizing advanced deep learning algorithms and ensemble classification methods. The goal is to increase the accuracy and performance of student success predictions. Furthermore, this will be directed towards investigating and comparing several machine learning and deep learning models to predict student performance at our university.

Acknowledgements

We acknowledge and thank the Faculty of Science and Ubon Ratchathani University for supporting this research.

References

Badal, Y. T. and Sungkur, R. K. (2023). Predictive modelling and analytics of students' grades using machine learning algorithms. **Journal of Education and Information Technologies**, 28(1), 3027–3057.

Bhushan, M., Vyas, S., Mall, S. and Negi, A. (2023). A comparative study of machine learning and deep learning algorithms for predicting student's academic performance. **International Journal of System Assurance Engineering and Management**, 14(6), 2674–2683.

Fullan, M. (2018). **New pedagogies for deep learning**. California: Corwin Press.

Inamdar A., Mhatre, T., Nadar, P. and Joshi, S. (2022), Personalized college recommender and cutoff predictor for direct second year engineering. In **The IEEE 7th International conference for Convergence in Technology** (pp.1-4). Mumbai: India.

Kaensar, C. and Wongnin, W. (2023). Analysis and prediction of student performance based on moodle log data using machine learning techniques. **International Journal of Emerging Technologies in Learning**, 18(10), 184-203.

Kaensar, C. and Wongnin, W. (2023). Predicting new student performances and identifying important attributes of admission data using machine learning techniques with hyperparameter tuning. **Eurasia Journal of Mathematics, Science and Technology Education**, 19(12), em2369.

Korchi, A., Messaoudi, F., Abatal, A. and Manzali, Y. (2023). Machine learning and deep learning-based students' grade prediction. **Operations Research Forum**, 4, 87.

Mangaroska, K. and Giannakos, M. (2018). Learning analytics for learning design: a systematic literature review of analytics-driven design to enhance learning. In **IEEE Transactions Learning Technologies**, 9(1), 1-19.

Mohammad, A., Al-Kaltakchi, S., Alshehabi, A.J. and Chambers, J. (2023). Comprehensive evaluations of student performance estimation via machine learning. **Mathematics**, 11(14), 3153

Park, H. S. and Seong, J. Y. (2021), Early dropout prediction in online learning of university using machine learning. **JOIV: International Journal on Informatics Visualization**, 5(4), 347-353,

Roy, A., Rahman, M. R., Islam, M. N., Saimon, N. I., Alfaz, M. and Jaber, A. (2022). A deep learning approach to predict academic result and recommend study plan for improving student's academic performance. **Springer: Ubiquitous Intelligent Systems, Smart Innovation, Systems and Technologies**, 253–266.

Sultana, J., Rani, M. U. and Farquad, H. (2019). Student's performance prediction using deep learning and data mining methods. **International Journal of Recent Technology and Engineering**, 8(1), 1018-1021.

Streib, F., Yang, Z., M. U., Tripathi, S. and Dehmer, M. (2020). An introductory review of deep learning for prediction models with big data. **Frontiers in Artificial Intelligence**, 3(1), 1-23.

Ubon Ratchathani University. (2010). REG UBU Office of Registration. Retrieved 5 Dec 2023, from **REG UBU**: https://www.reg.ubu.ac.th

Vijayalakshmi, V. and Venkatachalapathy, K. (2019). Comparison of predicting student's performance using machine learning algorithms. **International Journal of Intelligent Systems and Applications**, 11(12), 34-45.

Wibowo, S., Andreswari, R. and Hasibuan, A. M. (2018). Analysis and design of decision support system dashboard for predicting student graduation time. In **The 5th International Conference on Electrical Engineering**, **Computer Science and Informatics** (pp.684-689). Malang: Indonesia.