Building Student Course Performance Prediction Model Based on Deep Learning

JONG YIH KUO, HAO-TING CHUNG, PING-FENG WANG AND BAIYING LEI

1Department of Computer Science and Information Engineering
National Taipei University of Technology
Taipei, 106 Taiwan
E-mail: jykuo@ntut.edu.tw, colin19940702@gmail.com

2Institute for Information Industry
Taipei, 106 Taiwan
Email: pfwang@iii.org.tw

3School of Biomedical Engineering
Health Science Center, Shenzhen University, China
Email: leiby@szu.edu.cn

The deferral of graduation rate in Taiwan’s universities is estimated 16%, which will affect the scheduling of school resources. Therefore, if we can expect to take notice of students’ academic performance and provide guidance to students who cannot pass the threshold as expected, the waste of school resources can effectively be reduced. In this research, the recent years’ student data and course results was used as training data to construct student performance prediction models. The K-Means algorithm was used to classify all courses from the freshman to the senior. The related courses was grouped in the same cluster, which can more likely to find similar features and improve the accuracy of the prediction. Then, this study constructed independent neural networks for each course according to the different academic year. Each model was pre-trained by using Denoising Auto-encoder. After pre-training phase, the corresponding structure and weights was taken as the initial value of the prediction neural network model. Each neural network is treated as a base predictor. All predictors was integrated into an Ensemble predictor according to different years’ weights to predict the current student’s course performance. As the students finish the course at the end of each semester, the prediction model can continue track and update to enhance model accuracy through online learning.

Keywords: Deep Learning, Neural Network, Denoising Auto-encoder, Ensemble Learning

1. INTRODUCTION

According to the statistics of the Chief Officer of the Executive Yuan [1], the deferral of graduation rate in Taiwan is about 16% in the 2016 academic year. When the number of students who postpone graduation increases, it will affect the allocation of school resources, including computer equipment, classroom space, and curriculum configuration. From the social cost side, students may be dissociated from society and even went astray if students dropped out of school when they do not plan for the future or have a professional skill. Baas [2] noted that the costs of dropout are six times more than preventing students from dropping out the school. From the above mentioned, it is extremely important for duly counseling students who have high possibility of dropout. In order to let students to
graduate as scheduled, it is necessary to review the student's academic performance. In this way, the teachers and teaching assistants can provide guidance to those students who cannot pass the threshold as expected.

However, predicting student performance within a degree program faces some challenges [16]. First, students can differ tremendously in terms of backgrounds, and the same course can be taken by students in different areas. Since predicting student performance in a particular course relies on the student past performance in other courses. The key challenge for training an effective predictor is how to handle heterogeneous student data due to different areas and interests [17]. Similarly, predictions of students’ performance in courses are often based on in-course assessments which are designed to be the same for all students [10]. Second, students should take many courses but not all courses are equally informative for predicting students’ future performance. Utilizing the student’s past performance in all courses that he/she has completed not only increases complexity but also introduces noise in the prediction, thereby degrading the prediction performance. Third, the collection and cleanup of data is difficult. The number of students is limited, so as the training data set is limited. However, the grades and student data attributes is diverged to easily cause overfitting. This study should train with different number of layers and the number of neurons, to cluster all the courses, and to filter the effective attributes.

This research proposed a deep learning model that used the student academic big data in the school information system to build a prediction model. First, this study grouped all the courses data of the students into different clusters by K-Means method. After clustering phase, the training input data may be more useful to be the predicting data. The K-Means method was used to classify all the courses in the first to fourth grades so that those courses have the same attributes and can be grouped together in the same cluster, which helps the model to find similar features faster when training, to improve the accuracy of the analysis. If all the courses are included in the training data set, the feature vector will be it is very large and the complexity is increased, thus reducing the accuracy of prediction. Therefore, this study used a compulsory course for students to do training. Before training phase, the model applied De-noising autoencoder as pre-training method to learn more stable data features. During the training phase, this study applied dropout, cross-validation method to prevent overfitting, and to integrate all model of each school year according to different weights to predict students’ score. The current student data is used to track and update the proposed models continuously. Our proposed system can continuously track and update the model and increase the robustness of the model. For the school administrator, predicting a student's performance in the course is an ongoing task. As the student completes the course every semester, the prediction model needs to be continuously tracked and updated.

The remainder of this paper is organized as follows. Section 2 presents related work and method for finding the solution. Section 3, we present an effective model to improve the prediction for the student course performance. Section 4 is the model implementation. Section 5, we conduct two experiments including the comparison for the effectiveness of using a specific algorithm and comparison the other machine learning method. Section 6, we summarize the conclusions and the contributions of this paper.

2. RELATED WORK
(A) De-noising Autoencoder

Hinton et al. [5] proposed the model of Autoencoder. The model is divided into two parts: an encoder and a decoder. In order to make the data reconstructed by the encoder and decoder close to the original input data $x$, the model used mean square error as the loss function to measure the error between the input data and the reconstructed data.

$$
L(x, Z) = \frac{1}{d} \sum_{i=1}^{d} (x^{(i)} - x^{(i)})^2
$$

In order to let the hidden layer learn more stable features, Vincent et al. [15] mentioned that we can add random noises to the input layer of the network data, which is called De-noising Autoencoder (DAE). That is, adding a process of adding an artificial noise before the data is encoded. In the end, let the decoder restore the data without adding noise, learning a model that can eliminate the noise by itself. This model can be regarded as the pre-training of the neural network. Apply the pre-trained weights to the neural network as the initial value of the network structure, this neural network have the ability to suppress noise, which can be regarded as a normalization.

(B) Ensemble Learning

The concept of machine learning is to find a good enough function $h$ in the hypothesis space so that the function $h$ approximates the distribution function model $f$ of the real data. This approximate function is the predictor. The concept of ensemble learning is to improve the generalization of the overall predictors by training multiple predictors and combining the classification results of these predictors. The key to ensemble learning is to build differentiated predictors and integrate the results of these predictors. Common ensemble learning algorithms include AdaBoost and Bagging.

AdaBoost is an iterative algorithm proposed by Freund et al. [4]. A new weak classifier is added to each round. The misclassified data of the previous predictor have a higher probability to be used to train in the next predictor. At the beginning of the training, each data have the same weight, representing the probability that it was selected into the training set. If the data is correctly classified, its weight is decreased, and in the next training, the probability of the data being selected is decreased; on the contrary, if the data is not misclassified, its weight is increased. The algorithm is shown in Fig. 1.

```
1 Training data: \{(x^1, y^1), \ldots, (x^n, y^n), \ldots x^N, y^N\}
2 for t = 1, 2, \ldots, T:
3 Training weak classifier $f_t(x)$ with weight \{u^1, \ldots, u^T\}
4 $e_t$ is the error rate of $f_t(x)$ with weight \{u^1, \ldots, u^T\}
5 for n = 1, 2, \ldots, N:
6 if $x^n$ is misclassified by $f_t(x)$:
7 $u_{t+1}^n = u_t^n \times \exp(a_t)$
8 else:
9 $u_{t+1}^n = u_t^n \times \exp(-a_t)$

Fig. 1. Algorithm of AdaBoost.
```
AdaBoost can focus on those data that are difficult to classify. Finally, all predictors are multiplied by their respective weights $a_t$ through Eq. (3) to get the final model.

The Bagging algorithm is proposed by Breiman [3]. Given a training set of size $n$, each time select $n'$ data and put it back into the training set, and finally generate $m$ subsets of size $n'$ as new training and then build $m$ models in sequence. Finally, integrate the results of all models. If it is a regression problem, averaging all results; if it is a classification problem, using voting to determine which category appears the most. They used Bagging when the model is too complicated and overfitting, since Bagging averages each model, the final result should approach the overall average performance, so the variance should decrease and the possibility of overfitting should reduce.

(C) Cross Validation

Cross-validation is the process of cutting the sample into multiple small subsets for testing and training in machine learning, avoiding the bias caused by relying on specific training and test data. Kohavi [8] proposed that the cross-validation can effectively improve the accuracy of the model, a common form of cross-validation is K-fold cross-validation.

The K-fold cross-validation method first cuts the training data into K equal parts, the first set is used as the test data for validation, and the other K-1 sets are used for training. The cross-validation repeats K times, and finally the average the accuracy of K times to obtain a result without deviation, as shown in Fig. 2. Andrew Ng [12] mentioned that using the best training results in K cross-validation as a prediction model is not better than average K time’s cross-validation model.

\[
a_t = \ln(\sqrt{(1 - \epsilon_t)/\epsilon_t}) \\
H(x) = \text{sign} \left( \sum_{t=1}^T a_t f_t(x) \right)
\]

Fig. 2. K-fold cross validation.

(D) Related Course Prediction Case

Xu et al. [16] constructed a two-layer structure to use the students' materials for three years, including the students' high school GPA and SAT scores. Analysis, through the two-
tier architecture model, to predict the GPA scores of college graduation. This paper proposes a data-driven approach based on latent factor models and probability matrix analysis to discover the correlation between courses, and to establish a series of vector combinations of student learning results, which can effectively reduce the complexity between courses. However, this method is not suitable for us because the students’ high school grades do not fully implement the GPA method to evaluate the scores, so that the forecast results should have a large error. Our previous work [9] is to build a deep network model using the architecture of the Stacked De-noise Autoencoder, using the average ranking of the department, the average grade of the semester, and the scores of the professional subjects to establish the model. The study grouped the student’s data by the course. The predictors for each course of the semester are different through the overall learning and training methods. The overall predictor performance proposed in the study is better than not using the overall predictor, and it is more accurate to predict the results.

Our proposed approach adopt the idea of using students’ relevant attributes as training data, then further analyzed each subject score and made a more accurate judgment to let teachers clearly understand the learning effectiveness of students. The study [16] predicted the finally college GPA and the study [9] predicted the dropout rate of a student. However, neither of these studies can truly reflect the performance of students. Therefore, our proposed approach take advantage of the structure or method of the two studies, respectively, to focus on building a better, more accurate model, and to predict the score of each subject of students.

E. Tanuar et al. [13] integrated generalized linear model, deep learning and decision tree techniques to predict the student’s final year GPA. The data used in their experiment are from the computer science subjects, 6 subjects, 1 laboratories results and the GPA on their graduation year. According their experimental results, the important factors can be extracted to help students prepared themselves earlier. The accuracy results of the three proposed predication approaches are just 66.6%, 67.6%, and 60.6%. M. Tsiakmaki et al. [14] studied the predicting university students’ grades based on previous academic achievements. They carried out several experiments using eight courses modeled by some familiar mining methods, including linear regression, support vector machines, decision trees, M5 rules, and k-nearest neighbors. The evaluation metric used in their study for determining the efficiency of each regression method is the Mean Absolute Error but not the accuracy.

3. THE PROPOSED APPROACH

In order to use the data efficiently, we divided the data into training data, validation data and test data to train, fine-tune and evaluate the performance of the model. The model design flow chart is shown in Fig. 3.
Fig. 3. Model design flow chart.

(A) Data Preparation
The quality of the data affects the results of neural network learning and affects the accuracy of classification and prediction. Therefore, in order to ensure the correctness of the model analysis results, this research performed data reprocessing and data transformation.

(1) Student Course Data and Attributes
This research used the course grades and related attributes completed by the students of the Department of Electrical Engineering in the 2010-2017 academic year as offline training data sets. Use the courses that are currently not taken by the students to do the prediction.

Table 1. Student course name fields.

<table>
<thead>
<tr>
<th>Required course name</th>
<th>Type</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics 1&amp;2</td>
<td>Digital Logic</td>
<td></td>
</tr>
<tr>
<td>Physics Lab. 1&amp;2</td>
<td>Programming and Lab</td>
<td></td>
</tr>
<tr>
<td>Circuit Theory 1&amp;2</td>
<td>Probability</td>
<td></td>
</tr>
<tr>
<td>Engineering Mathematics 1&amp;2</td>
<td>Microprocessor</td>
<td></td>
</tr>
<tr>
<td>Electronics 1&amp;2</td>
<td>Signals and Systems</td>
<td></td>
</tr>
<tr>
<td>Electronics Lab. 1&amp;2</td>
<td>Special Projects 1&amp;2</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Student related attributes fields.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sex</td>
<td>Male - Female</td>
</tr>
</tbody>
</table>
Course selection should vary depending on the interest and learning field. Also, some courses are newly opened and some have been closed. If all courses are included in the training data set, the feature vector will become very large. It is not only increase the complexity and also reduce the prediction accuracy. Therefore, we focus on predicting the required subjects. The data field descriptions are shown in Table 1, Table 2, respectively.

(2) Data Integration and Cleaning

The quality of the data and the selection of features greatly affect the results of the analysis. There are three types of preprocessing for data.

- **Feature scaling**
  Data must limit the value range of the training sample from 0 to 1 to avoid excessive weight in the training process of the neural network. The formula is as follows:

  \[
  X' = \frac{X - X_{min}}{X_{max} - X_{min}}
  \]

  \(X_{max}\) and \(X_{min}\) are the maximum and minimum values of one attribute. By subtracting \(X_{min}\) from the original value and dividing by \(X_{max} - X_{min}\), the value range of the attribute can be compressed to 0 to 1, reducing the impact of the outliers to the model.

- **Deal with missing data**
  Data with missing values are divided into two processing ways, one is to directly discard, and the other is to use the average value of all the data in the column to replace. For example, if a student lacks too many course grades, we directly deleted the student data. If there is only one or two-course grade missing, we used the average score of that subject to replace.

- **Non-numeric type**
  Non-numeric data must encode into numbers to be input into the neural network for training. It can separate into ordered and disordered. We use One-Hot Encoding and Label Encoding to encode the disordered and ordered data. For example, the identity of the normal student is 100, the foreign student is 010, the others are 001; the parent's education level is ordered, according to the level of education, labeled in the range of 0 to 1.

(3) Data Integration and Cleaning

We use related attributes and course scores of students in the 2010-2016 academic year as training data, in which the K-Fold algorithm was used for cross-validation. We use 2017 academic year students as the testing data to evaluate the performance of the model.

(B) Course Cluster
There are 22 required courses in the department of electrical engineering. It should be ineffective if we use all the completed courses to predict a new course because there is no correlation between many courses, for example, the correlation between calculus and physics may be small. However, the score of calculus 1 affecting calculus 2 may be more significant. The aim of this research is to find the course cluster of the related course so that the relevant courses were classified in the same cluster, as shown in Eq. (5). \( C \) represents the set of courses that have been completed up to the \( t \) semester, \( K(j) \) indicates the set of courses in which the course \( j \) is located. The intersection vector of the two sets is regarded as the predictive input of the course \( j \). The students and the corresponding course scores are organized into a matrix, as shown in Fig. 4. The K-Means [6] is used to classify related courses to the same cluster.

\[
\text{input of } j = C^t \cap K(j) \tag{5}
\]

(C) Build a Deep Neural Network

We used a fully-connected four layer deep neural network as the network architecture. The first hidden layer has 256 units, the second hidden layer has 128 units, and the output layer has one unit. The neural network of each course is considered as a predictor of the prediction model. Before training the neural network, we use DAE as the pre-training to initialize the value of the weight.

Fig. 4. Student course score matrix.

(1) Pre-training

Because the traditional neural network usually uses 0 or a value close to 0 as the initial weight value. Such initial values tend to converge to a local minimum in the deep network.
of the multi-layer hidden layer. Using DAE as a pre-trained weight can reduce the probability of the neural network converge to a local minimum.

The architecture of the pre-training model is shown in Fig. 5. The model divided into two parts: encoder and decoder. Input \( x \) first add some noise to prevent overfitting and increase restore the ability of the model. The \( x' \) reduces the dimension through neural network layer encoding, and then decodes the restored data into \( \hat{x} \). We use mean square error to calculate the loss between \( x \) and \( \hat{x} \) in each iteration. After the pre-training, the corresponding structure and weight \( \theta \) are taken as the initial values of the neural network model.

(2) Base Predictor

We establish a base predictor for each course in each academic year. Each base predictor is an independent neural network. The trained weight is used as the initial value of the predictor. As shown in Fig. 6, in each academic year, we trained \( n \) base predictors \( h \), \( n \) is the number of courses to be predicted. Finally, we combine every base predictors \( h \) of the same course into the ensemble predictor \( f \).

![Fig. 6. Architecture of base predictor.](image)

(3) Ensemble Predictor

The ensemble predictor integrated the base predictor of all the same courses each year, as shown in Eq. (6). \( \text{Current} \) represents the current year, \( n \) is the course to be predicted and \( f^\text{Current}_n \) is the ensemble predictor of the course to be predicted. This study used the students’ academic data for 99th to 106th school year as training data. “99th school year” is equal to A.C 2010. Therefore, the “totalYear” should be “Current-2010”. The assignment of weights should increase by the year as shown in Fig. 7. Because the closer year may be taught by the same teacher and there should be higher similarity in the scores.

\[
f^\text{Current}_n = \sum_{y=99}^{Current-1} w_y \times h^y_n \tag{6}
\]

1 \( \text{totalYear} = \text{Current} - 2010 \)
2 \( \text{index} = \text{totalYear} \times (\text{totalYear}+1) / 2 \)
3 \( \text{for } y \text{ in range(2010, Current - 1)}: \)
4 \( w_y = (1/\text{index}) \times y \)

![Fig. 7. Algorithm of weight assignment.](image)
Even though a particular base predictor performs best during training, it may not have good predictive power because the scores for each semester combined many different factors. If only use one specific base predictors, the result may be very unsatisfied. Therefore, integrating those results of the annual predictor can improve the generalization ability and robustness of the predictor.

(D) Prediction

We use ensemble predictor to predict each course performance of current semester, and then use the predicted scores to predict future performance. By the end of the semester, we further use the student's grades to update the model and establish a system that continuously tracks student grades and accurately predicts their future performance. Fig. 8 introduces the process of prediction, as detailed below:

1. Use the student's data $x$ as input to fit into different ensemble predictors model $f_{current}$ to predict the course score in the t-semester.
2. Use the predicted score $preY$ and $x$ as new input to predict the score for the next semester, and so on, and get the score for each semester course iteratively.
3. After the end of the semester, use the actual score $realY$ to create a new base predictor model $h_{current}$.

This research used CSV to present the results of the prediction, showing the student class, student id, name and the course score.

4. MODEL IMPLEMENTATION

The system environment is i7-6700 and 16G memory, the prediction model is implemented by Python and TensorFlow framework.

(A) Neural Network Model

The loss function is used to estimate the accuracy of the model's predicted value $f(x)$ and the true value $y$. Because predicting score is a regression problem, we use root mean square error to find the loss of the model. The smaller the loss function is, the better the model fits. However, when the neural network model is too complex, it will easily overfitting. We use L2 regularization to measure the complexity of the model. Reduce unimportant weight values in the neural network, as Eq. (8) shown.
Before training phase, we applied DAE as pre-training to initialize the weight. The algorithm is shown in Fig. 9. The initial value weight $\theta$ is set, and the iteration is 3,000 times. In each iteration, the training data may contain noise. The ratio $c$ is 0.3. The root mean square error is used to calculate the loss between the original data $x$ and the decoded restored data $\hat{x}$. Adam method [7] is applied as the optimization method to reduce the loss to the lowest. After finish the training phase, the algorithm return the weight $\theta$.

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)}$$ (7)

$$\text{loss} = \text{RMSE} + \alpha \times \frac{1}{2} \sum w_i$$ (8)

After pre-training, every weight $\theta$ have been trained as shown in Fig. 10. The ReLU function [11] is used as the activation function of the network. According to the back propagation algorithm, the Adam method is used to fine-tune the weight value, and to calculate the loss function of the model. The prediction result $O_t$ is obtained to compare with the expected result $y$ through linear regression between the hidden layer and the output layer.

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)}$$ (7)

$$\text{loss} = \text{RMSE} + \alpha \times \frac{1}{2} \sum w_i$$ (8)

![Fig. 9 Algorithm of DAE](image)

![Fig. 10 Neural network diagram](image)
The training of the model adopts K-fold cross-validation to divide the training data and validation data. The K value is 5 times. In order to ensure that the model is not over-trained, each cross-certification was cooperated with early stop, as shown in Fig. 11. Although this method should reduce the accuracy of the model to the training data, it can effectively improve the versatility of the model. We obtain root mean square error of the validation data by each epoch. If the error rate continues to decrease, save each updated model; if the error rate of the validation data does not improve more than 500 epochs, then the model stop training. The algorithm is shown in Fig. 12. After completing five cross-validations, we averaged five models to obtain the final neural network model.

**Initialization: min** is the minimal validation loss threshold

```python
1 If val_loss < min:
2    min ← val_loss
3    Save current model, and Record current epoch
4 If val_loss didn’t get smaller more than 500 epochs:
5    break
```

**Initialization** \( \alpha \) is the learning rate, \( p \) is the dropout rate, iteration

```python
1 Deep Neural Network \((x, y, x_{\text{val}}, y_{\text{val}}, \theta)\):
2 \( x = [x^{(1)}, x^{(2)}, ..., x^{(m)}] \in \mathbb{R}^{m \times n} \) is the input matrix, where \( m \) is the number of data
3 \( y = [y^{(1)}, y^{(2)}, ..., y^{(m)}] \in \mathbb{R}^{m \times 1} \) is the real score
4 \( \theta = [\theta_1, \theta_2, ..., \theta_l] \), pre-training by De-noising Autoencoder, where \( \theta_i = [W_i, b_i] \)
5 \( O = [O_0, O_1, ..., O_k] \in \mathbb{R}^{m \times k} \) is the output of each layer, which \( O_0 = x \).
6 for epoch in range(iteration):
7    \( \theta = \text{dropout}(\theta, p) \)
8    for i in range(l - 1):
9        \( O_i = \text{ReLU}(O_{i-1} \cdot W_i + b_i) \)
10       regularization = L2Regularization(W_i)
11    end for
12    \( O_l = \text{Linear Regression}(O_{l-1}, \theta) \)
13    loss = RMSE(y, O_l) + regularization \cdot \alpha
14    y_{\text{val_predict}} = \text{Linear Regression}(x_{\text{val}}, \theta)
15    val_loss = RMSE(y_{\text{val}}, y_{\text{val_predict}})
16    EarlyStop(val_loss)
```
Compute the gradients of the loss with respect to θ:

\[ g = \frac{\partial L}{\partial \theta} \]

for \( \theta_i, g_i \) in \( (\theta, g) \):

\[ \theta_i = \theta_i - \alpha \times g_i \]

end for

end procedure

Fig. 13 algorithm of neural network model

Fig. 13 is the complete deep neural network algorithm. The number of iterations is set to 5,000 times. Each iteration should do dropout with dropout ratio \( p = 0.2 \), the learning rate \( \alpha = 0.01 \), and use L2 regularization for loss function. Use cross-validation, early stop, and other algorithms to make the model more stable. Finally, we use Adam to update the weight and then training for the next iteration.

(B) Combine Neural Networks

Combine neural network model \( h_i^y \) of each course in different years with different weights \( w_y \) to form an ensemble predictor, as shown in Fig. 14 algorithm.

\[ f_i = \sum w_y \times h_i^y \]

Fig. 14 algorithm of ensemble learning

5. EXPERIMENTAL RESULTS

This chapter is divided into two categories of experiments, one for comparing the effectiveness of using a specific algorithm, another for comparing the research model with other machine learning methods. The chart for each experiment is the average of ten experiments. Table 3 is the code number of each course to be predicted, and it is sorted according to the order of the courses specified in each semester from the first year to the fourth year.

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Code</th>
<th>Course Name</th>
<th>Code</th>
<th>Course Name</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engineering Mathematics 1</td>
<td>A</td>
<td>Probability</td>
<td>B</td>
<td>Electronics 1</td>
<td>C</td>
</tr>
<tr>
<td>Electronics Lab. 1</td>
<td>D</td>
<td>Circuit Theory 1</td>
<td>E</td>
<td>Engineering Mathematics 2</td>
<td>F</td>
</tr>
<tr>
<td>Microprocessor</td>
<td>G</td>
<td>Electronics 2</td>
<td>H</td>
<td>Electronics Lab. 2</td>
<td>I</td>
</tr>
<tr>
<td>Circuit Theory 2</td>
<td>J</td>
<td>Signals and Systems</td>
<td>J</td>
<td>Special Projects 1</td>
<td>L</td>
</tr>
<tr>
<td>Special Projects 2</td>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(A) The Efficiency of DAE

This experiment compares the performance between using DAE or not, and uses the scores of the students as test data to evaluate the loss of the model.

In Fig. 15, it can be seen that the model without pre-trained has a very high loss at the initial stage, and the loss segment is jagged, which means that the training process cannot be stably reduced. Although the loss of training data is getting smaller and smaller at a late stage, the gap between the training data and the validation data is gradually widening, which means the model does not have the general ability to adapt to data. On the contrary, the error between the pre-trained model prediction result and the expected result is dropped very fast in the early stage, and after about 1500 iterations, the best local minimum can be quickly found to early stop the training.

![Fig. 15 Loss of pre-training and without pre-training](image1)

This experiment uses the student's grades as test data to compare the differences between the pre-trained and without pre-trained courses in each subject. As shown in Fig. 16, we can observe that each course performs better after pre-training.

![Fig. 16 Compare with each course](image2)

(B) The Efficiency of Course Cluster

In order to understand whether the clustering of the course helps to improve the performance of the model, this experiment selects three course: Electronic Internship 1, Electronics 2, and Circuit 2, based on the pre-training model of the previous chapter. To explore the comparison of the loss of training scores between clustering and the non-clustering. As shown in Fig. 17, the degree of course loss of the clustering has been far better than that of
the non-clustering at the beginning of the training. After 3000 iterations, all of the clustering courses perform better than the non-clustering courses.

![Fig. 17 Efficiency of course cluster](image)

(C) The Efficiency of Ensemble Learning

The purpose of this experiment is to understand whether the performance of the model combination in each school year has improved. Take the course of signal and system as an example. The students who take this course in each school year was used as training data to train the model. The loss of the model to the test data is shown in Table 4. The loss distribution is from 7.123 to 8.268, and the closer to the year of test data is, the better performance it is. Combine 2012 to 2016 academic years with different weights to build an ensemble predictor. Use the students who took the 2016 academic year course as test data obtained the loss of 6.831. Therefore, the ensemble predictor did improve the performance of the model.

<table>
<thead>
<tr>
<th>Training Model (Academic year)</th>
<th>2012</th>
<th>2013</th>
<th>2014</th>
<th>2015</th>
<th>2016</th>
<th>Ensemble Predictor</th>
</tr>
</thead>
</table>

Further, we compared the performance results of using the ensemble predictor architecture or not. This experiment compares the test data using the ensemble predictor performance difference in each course, as shown in Fig. 18, because the network model has been pre-trained. The two models have performed very well, but our proposed model combines the results of different predictor, making the prediction more precise.
(D) The Efficiency of Ensemble Learning

This section compared the predictive power of our proposed model with Support Vector Regression, Linear Regression, and Random Forest, three model learning algorithms. Fig. 19 shows the loss of each course in different methods. The loss function of our model is the smallest and performs the best among the test data.

In order to examine the performance, this experiment established the confusion matrix, uses the score of the pass as a judgment of whether the model performs well. The condition of True Positive is the number of students who cannot pass the course in the real situation and the model correctly predicts the student didn’t pass the course. The results are shown in Table 5. The F-measure score of our model is the highest, which means that we can effectively pick out the failed students.

Table 5 Efficiency of different algorithms

<table>
<thead>
<tr>
<th></th>
<th>Our Model</th>
<th>Random Forest</th>
<th>SVR</th>
<th>Linear Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSE</td>
<td>7.625</td>
<td>9.222</td>
<td>11.241</td>
<td>10.79</td>
</tr>
<tr>
<td>Accuracy</td>
<td>89.62%</td>
<td>85.84%</td>
<td>78.3%</td>
<td>77.35%</td>
</tr>
<tr>
<td>Precision</td>
<td>76.47%</td>
<td>60.86%</td>
<td>43.47%</td>
<td>40.9%</td>
</tr>
<tr>
<td>Recall</td>
<td>65%</td>
<td>70%</td>
<td>50%</td>
<td>45%</td>
</tr>
<tr>
<td>F-measure</td>
<td>70.27%</td>
<td>65.11%</td>
<td>46.51%</td>
<td>42.8%</td>
</tr>
</tbody>
</table>
6. CONCLUSION

In this paper, we proposed a novel method for predicting students’ future performance in degree program given their academic performance and the related attributes. A course clustering method was developed to discover relevant courses for constructing the pre-trained models that can effectively find similar course features to improve prediction accuracy and make the training process relatively stable so that it is not easy to fall into local optimization. An ensemble-based progressive prediction model was developed to incorporate students’ evolving performance into the prediction. The model can continue track and update students’ academic data to improve the prediction accuracy. The experimental results demonstrated that the training data according to different years of the course can be effectively reduced one to two points. Finally, we compared our model with other machine learning model, so that our model had the highest predictive ability in measuring the student’s performance. Additionally, this work can impact curriculum design in degree programs and education policy design in general. This study also can provide teachers with predictive reports to review student performance, reducing the possibility of deferral graduation.

ACKNOWLEDGMENT

This research was supported by Ministry of Science and Technology, R.O.C. program MOST 106-2221-E-027-018-MY2, and National Taipei University of Technology program NTUT-SZU-107-06.

REFERENCES