Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 1422 – 1441

doi:10.3906/elk-1907-8

Deep neural network based m-learning model for predicting mobile learners' performance

Muhammad ADNAN[1,*], **Asad HABIB**[1], **Jawad ASHRAF**[1], **Shafaq MUSSADIQ**[1], **Arsalan ALI RAZA**[2]

[1]Institute of Computing, Faculty of Numerical and Physical Sciences, Kohat University of Science and Technology, Kohat, Pakistan

[2]Department of Computer Sciences, Faculty of Sciences, COMSATS University Islamabad, Vehari Campus, Pakistan

Received: 01.07.2019 • **Accepted/Published Online:** 11.09.2019 • **Final Version:** 08.05.2020

Abstract: The use of deep learning (DL) techniques for mobile learning is an emerging field aimed at developing methods for finding mobile learners' learning behavior and exploring important learning features. The learning features (learning time, learning location, repetition rate, content types, learning performance, learning time duration, and so on) act as fuel to DL algorithms based on which DL algorithms can classify mobile learners into different learning groups. In this study, a powerful and efficient m-learning model is proposed based on DL techniques to model the learning process of m-learners. The proposed m-learning model determines the impact of independent learning features on the dependent feature i.e. learners' performance. The m-learning model dynamically and intuitively explores the weights of optimum learning features on learning performance for different learners in their learning environment. Then it split learners into different groups based on features differences, weights, and interrelationships. Because of the high accuracy of the DL technique, it was used to classify learners into five different groups whereas random forest (RF) ensemble method was used in determining each feature importance in making adaptive m-learning model. Our experimental study also revealed that the m-learning model was successful in helping m-learners in increasing their performance and taking the right decision during the learning flow.

Key words: M-learning model, artificial neural network, deep learning algorithm, features importance, machine learning, mobile learning

1. Introduction

The aim of this research study is to develop and deploy an m-learning model that can deliver adapted learning contents to individual learners taking into account their preferences, performance, learning contexts, and background knowledge.

For creating an efficient and comprehensive m-learning model, it is important to carefully observe what learning features are generated during the learning process and how these features change for diversified learners. M-learning features can be categorized into different groups which include learning content (text, video, audio, problem-solving), learning context (time, location, mobile device capabilities, learning duration), social interaction (learning contents liked, disliked, commented, shared), target learning object, number of revision for target object, number of clicks, login frequency, and the learner performance.

For deep learning (DL) algorithms, the proper m-learning dataset is a crucial element for creating the right m-learning model [1]. DL algorithms are based on a collection of statistical and machine learning (ML)

*Correspondence: adnan@kust.edu.pk

techniques used to find feature orders, weights, and their relationships often based on deep neural networks [2]. Learning models based on DL algorithms require little guidance from programmers as they are capable of focusing on the right features by themselves.

The primary challenge in m-learning systems is to decide which learning features to store and use as input to DL algorithms. Proper and right learning features are important for efficient modeling of learner knowledge and providing exclusive information in the adaptive learning process. The choice of learning features affects the overall performance of the m-learning system. The other important challenge is to decide which learning content is the most appropriate for a particular learner in a specific learning environment. Various learning models and machine learning algorithms in literature had been researched and studied to map learners to tailored learning contents [3, 4].

A proper learning algorithm affects the overall performance of the m-learning system and can increase/reduce the response time of the m-learning system. For this purpose, different classical machine learning algorithms have been developed and used and their performances have been explored in m-learning and online learning courses [5–7]. For instance, probability estimation algorithms such as naïve Bayes and expectation maximization are the best when it comes to producing efficient results on training dataset but they can be expensive to apply [8, 9]. The computation of conditional probability on every hypothesis can be quite costly. Therefore, other types of machine learning algorithms are needed for modeling learners in mobile and online courses. Machine learning algorithms like support vector machine [10], k-nearest neighbors [11], decision trees [12], k-means [13], and density-based spatial clustering of applications with noise (DBSCAN) [14] etc. are based on classification and clustering. These classification and clustering algorithms work well on small data. These are computationally economical and easier to interpret. The main disadvantage of these algorithms is that they need complex featured engineering processes, do not scale effectively with data, and are not best in accuracy and performance. For example, learner features are dynamic and the online learning process generates a lot of data while these algorithms work well with static features and a small amount of data. Traditional machine learning algorithms are complex and need a lot of human interventions and domain expertise.

DL algorithms are gaining popularity over classical machine learning algorithms due to their accuracy and performance when trained on a huge amount of data [15]. They offer a greater level of flexibility as they can represent information at different levels of granularity [16]. A major advantage of DL algorithms over traditional machine learning algorithms is that they are capable to learn abstract features from low-level data in an incremental manner. This eliminates the need for human intervention, hardcore feature extraction, and domain expertise [17].

The organization of the paper is as follows: Section 2 discusses related studies and approaches in learner modeling. Section 3 presents the learning features and architecture of the m-learning system. The architecture of the m-learning system includes features extraction process, features encoding, features scaling, making the train and test sets, learners classification, and artificial neural network (ANN) learning process i.e. forward and back-propagation. Section 4 explains the m-learning model implementation process using ANN and random forest (RF) ensemble method. Section 5 discusses the use of the RF ensemble method to determine feature weights and importance in predicting the final learning performance of m-learners. Section 6 presents the early intervention experiment along with its results and discussion. Section 7 concludes our paper and mentions the future direction of this research.

2. Related studies (learner modeling approaches)

Machine and deep learning techniques have gained much attention in decision making, education, and knowledge processing in recent years. A considerable amount of literature target predicting learners' performance in completing academic courses, solving quizzes, and reducing dropout rates. Many machines and deep learning techniques such as trained neural networks [18], feed-forward neural networks [19], self-organizing maps [20], recurrent neural networks [21], matrix factorization [22], and probabilistic graphical models [23] have been practiced to develop prediction algorithms in finding the right learning behavior.

M-learning systems were developed under the influence of studies in the area of E-learning, intelligent tutoring systems, adaptive learning, and computer-aided learning [24]. Most m-learning system architectures are an extension of E-learning architectures [25, 26]. Context discovery, user profiles, user tracking, content management, and semantic indexing are key elements in developing adaptive m-learning systems. According to Sharples et al. [28], m-learning "is not confined to pre-specified times or places, but happens whenever there is a break in the flow of routine daily performance and a person reflects on the current situation, resolves to address a problem, to share an idea, or to gain an understanding". All the knowledge and skills that are needed in daily routine cannot be provided by the formal education system. Therefore, to address immediate problems, to participate in continuous professional development and to collaborate in teamwork, a system that gives users the freedom to work independently of temporal, spatial, and social constraints is needed.

The changing context is the central construct that differentiates m-learning from traditional learning. m-learning context changes with location, time, environment, everyday tools and dissimilar people [29, 30]. In a traditional learning environment, there is an agreed curriculum with a single instructor and a stable learning context. In m-learning settings, the fundamental challenge is to create an environment of temporary context that assist learners in making their learning easy and meaningful.

Nordin et al. [27] presented a conceptual mobile learning framework that delivers organized sustenance for mobile lifelong learning involvement design. Learning theories, mobile environment issues, mobile learning context, and learning experience and objectives were the key design factors of a lifelong mobile learning framework. Both constructivism and behaviorism learning theories can be used in the design of mobile learning instructional materials [31]. Collecting m-learners profile data, considering user interface design concerns, and examining learners' mobility constitute user mobile environment issues. The use of mobile devices in the acquisition of knowledge in a different context is often considered as supporting tool. Mobile devices in different learning scenarios can be used in preactivity or postactivity mode. Learning experience and objectives are not only concerned with usability goals but also with how the design of the user interface will have an impact on the user. While using mobile devices, the interface should present engaging experience, pleasing interface along with organized contents, and clear goals and objectives.

Contemporary user modeling systems track and store users' interaction information [36]. One of the basic tasks of user modeling systems is to extract meaningful information from persistent data storage. In the evaluation process of user data, mostly learning classifier systems, rule-based inference systems, machine learning algorithms and information engineering methods are used. The key objective of these systems is classification/prediction/clustering users into different categorical groups. Albert et al. [33] successfully conducted an experiment to detect fall rates and types in elderly people and classified them into different types using support vector machines and regularized logistic regression algorithms. Both of these algorithms were able to recognize the fall with 98% precision and classify the fall types with 99% accuracy. Their research demonstrated

that machine learning algorithms can simplify the data analysis process in fall-related research and how rapid response time can be reduced to heal potential damages due to falls.

O'Mahony and Symth [34] successfully described and implemented a supervised classification algorithm aimed to pinpoint and recommend the most supportive product reviews. Based on the TripAdvisor service case study, they compared the performances of naïve Bayes, JRip, and J48 classification algorithms using features dataset derived from hotel reviews. Those reviews were considered helpful for which at least 75% of the classified opinions were positive. Reviewing instances were composed of features derived from four different categories extracted from individual and community reviewing activity.

Martins et al. [36] presented a user model in which preferences and background knowledge were key components in defining an online adaptation process. Based on user preferences, adaptive navigation paths were defined and based on background knowledge adaptive contents were delivered. A task model was developed by Bezold [37] which considered users' activities in online interaction systems as a series of events. In his study, Bezold used 'probabilistic deterministic finite-state automata PDFA to describe user behavior in different online activities. In order to estimate the next user activity, 'first-order Markov chains' approach was used. First-order Markov chains approach converted user interaction history into vector set and then used a vector set as input parameters for estimating user activities in the future. The problem with task-based user modeling is that there is no proper standard methodology for evaluating the methods used [38].

Guo et al. [39] developed a classification model using an unsupervised sparse auto-encoder algorithm from learners' unlabeled data. The classification model was trained on a relatively large dataset aimed to pretrain hidden layers of features layer wisely. The experimental study showed the effectiveness of the model in an academic setting for learners' prewarning mechanism. The main drawback of sparse auto-encoder is low network architecture performance and it is not suitable for time series data [40].

Bouneffouf [41] used a reinforcement learning technique called the Markov decision process to create a ubiquitous recommender system based on the user's context. Knowing context information, the recommender system offers an appropriate recommendation to the user. Initially, the recommender system recognizes a new user based on user social group and then gradually recommends new actions according to the user's interest. The recommender system associates actions to the perceived context of the user. Associations depend on user feedback to the system behavior. The author succeeded in solving users' cold-start problem experienced by most of the new users who have little experience with the new system.

In previous research studies, many techniques and approaches have been developed to model the behavior of online users; however, most of them are not applied in a real-world setting [43, 44]. The main reason for this problem turns out to be placing the user in application domain constraints and not considering his/her preferences and features. Users are dependent on the system's complexity, low-level details, complex practices, and theoretical models. According to literature, each of the users' features is equally important in defining their exact behavior [42]. In other words, contemporary user modeling approaches assign equal weight to every feature in the user modeling process. Not considering feature weights and their interrelationship is the main cause of misclassification in user modeling approaches. Deep learning (DL), a subset of artificial intelligence, has enabled the development of a comprehensive users/learners model which represent and detect a broader range of user behavior than was previously possible. DL algorithm like ANN with hidden layers is capable of determining important features along with their weights in classifying users in different categories. Assigning a

weight to each feature is called a weight-tuning approach. The weight-tuning approach improves user/learner modeling estimation, prediction, and classification results.

3. Learning features and the architecture of the m-learning system

Deep neural network algorithms like ANN segregate learners learning behavior based on the real-values of learners' features. ANN can identify significant learning features to determine different learners' groups having peculiar m-learning models.

The m-learners features used in ANN processing and modeling are listed in Table 1. We used 21 features to train our m-learning model. These 21 features are independent features which are delivered to deep neural network while one feature called the learner grades derived from the final performance is a dependent feature which is predicted by our proposed m-learning model. The data for 21 independent features were acquired from learners' interaction with the m-learning system for 4 months. Nine hundred and twenty-one learners belonging to the Institute of Computing and Institute of Numerical Sciences, Kohat University of Science and Technology, KUST participated in using the m-learning system for learning and refining their JAVA programming abilities. The data of 894 learners was finally selected as input to ANN-based m-learning model. The JAVA language course delivered on smartphones was divided into three modules namely module 1, module 2, and module 3. The following paragraphs discuss m-learning architecture and workflow in more detail.

Figure 1 shows the architecture of the proposed m-learning system based on the m-learning model. The architecture workflow is divided into seven steps. The learning features of m-learners are processed in these seven steps. In the first step, learners' features are collected from their mobile devices. The m-learning system collects learner's information such as learning activities, learning content type, responses, learning time and learning time duration about the target learning objects. In the second step, the collected data is stored in the online cloud. Initially, this online data represent a generic profile of a learner as the learning features are not classified and weighted.

In the third step of m-learning system workflow, important learner attributes are extracted, preprocessed, and converted into useful data in order to become suitable for input to the deep neural network-based ANN algorithm. The converted features represent the real-values of learner features in the set <MLT_1, MRRO_1, MALP_1, ….. FP>. These independent features are passed through the ANN algorithm to determine what learning features are important for each learner and by what means to arrange learners into different groups. The deep neural network also determines the effect of these independent features on the dependent feature of learners' final grades. Feature preprocessing includes features encoding, making training and testing sets, and features scaling as discussed in the following sections.

3.1. Features encoding

We are confronted with learners' classification problem. The learners are classified into five categorical groups A, B, C, D, and F, representing the final grades. Before delivering independent and dependent features to the ANN algorithm, it is important to encode them into the appropriate format. Luckily, in our dataset, we have one categorical feature called learner grades (dependent feature) which needs to be encoded. We used a one-hot encoding scheme to encode learner grades into the numeric form. One hot encoding scheme is used because with this scheme, deep neural networks do a better job with classification and prediction. One hot encoding is also used to break the ordinal relationship among categorical feature values and due to a higher numerical value, one categorical feature does not get a higher preference over other categorical features.

Table 1. The preprocessed m-learner features gathered during the learning process.

Features	Description
MLT_1	Learning time taken in module 1 (numeric: 0 to 1.9 $\Longrightarrow$ between 0 and 20 min, 2 to 2.9 $\Longrightarrow$ between 20 and 30 min, 3 to 3.9 $\Longrightarrow$ between 30 to 40 min, >= 4 $\Longrightarrow$ 40 and greater than 40 min)
MRRO_1	Repetition rate in module 1 (numeric: 0.01 = 1 time repetition, 0.02 = 2 times .... 0.06 = 6 times)
MALP_1	Degree of academic location visited in module 1, numeric: 0 to 1
MSEP_1	Degree of social and entertainment location visited in module 1, numeric: 0 to 1
TCUM_1	Degree of text content types used in module 1, numeric: 0 to 1
VCUM_1	Degree of video content types used in module 1, numeric: 0 to 1
MP_1	Module 1 performance, numeric: (18 to 20 = very good), (15 to 18 = good), (12 to 15 = average), (9 to 12 = satisfactory), (0 to 9 = fail)
MLT_2	Learning time taken in module 2 (numeric: 0 to 1.9 $\Longrightarrow$ between 0 and 20 min, 2 to 2.9 $\Longrightarrow$ between 20 and 30 min, 3 to 3.9 $\Longrightarrow$ between 30 to 40 min, >= 4 $\Longrightarrow$ 40 and greater than 40 min)
MRRO_2	Repetition rate in module 2 (numeric: 0.01 = 1 time repetition, 0.02 =2 times .... 0.06 = 6 times)
MALP_2	Degree of academic location visited in module 2, numeric: 0 to 1
MSEP_2	Degree of social and entertainment location visited in module 2, numeric: 0 to 1
TCUM_2	Degree of text content types used in module 2, numeric: 0 to 1
VCUM_2	Degree of video content types used in module 2, numeric: 0 to 1
MP_2	Module 2 performance, numeric: (18 to 20 = very good), (15 to 18 = good), (12 to 15 = average), (9 to 12 = satisfactory), (0 to 9 = fail)
MLT_3	Learning time taken in module 3 (numeric: 0 to 1.9 $\Longrightarrow$ between 0 and 20 min, 2 to 2.9 $\Longrightarrow$ between 20 and 30 min, 3 to 3.9 $\Longrightarrow$ between 30 and 40 min, >= 4 $\Longrightarrow$ 40 and greater than 40 min)
MRRO_3	Repetition rate in module 3 (numeric: 0.01 = 1 time repetition, 0.02 =2 times .... 0.06 = 6 times)
MALP_3	Degree of academic location visited in module 3, numeric: 0 to 1
MSEP_3	Degree of social and entertainment location visited in module 3, numeric: 0 to 1
TCUM_3	Degree of text content types used in module 3, numeric: 0 to 1
VCUM_3	Degree of video content types used in module 3, numeric: 0 to 1
MP_3	Module 3 performance, numeric: (18 to 20 = very good), (15 to 18 = good), (12 to 15 = average), (9 to 12 = satisfactory), (0 to 9 = fail)
FG	Final grades derived from final performance score, categorical (A,B,C,D,F)

3.2. Feature scaling

To ease up the ANN processing and calculation, learners' features are scaled (converted into one unit). Feature scaling is necessary as it would not allow one independent feature to dominate another independent feature and it is also needed for deep neural networks parallel and high computations. To scale learner features, the values of features are divided by the maximum value for each feature as shown in equation 1.

$$X_{norm(1-21)} = X_1/MAX(X_1), X_2/MAX(X_2), ..., X_{21}/MAX(X_{21}) \quad (1)$$

Here X1, X2,.....X21 are the 21 independent features to be supplied into m-learner model.

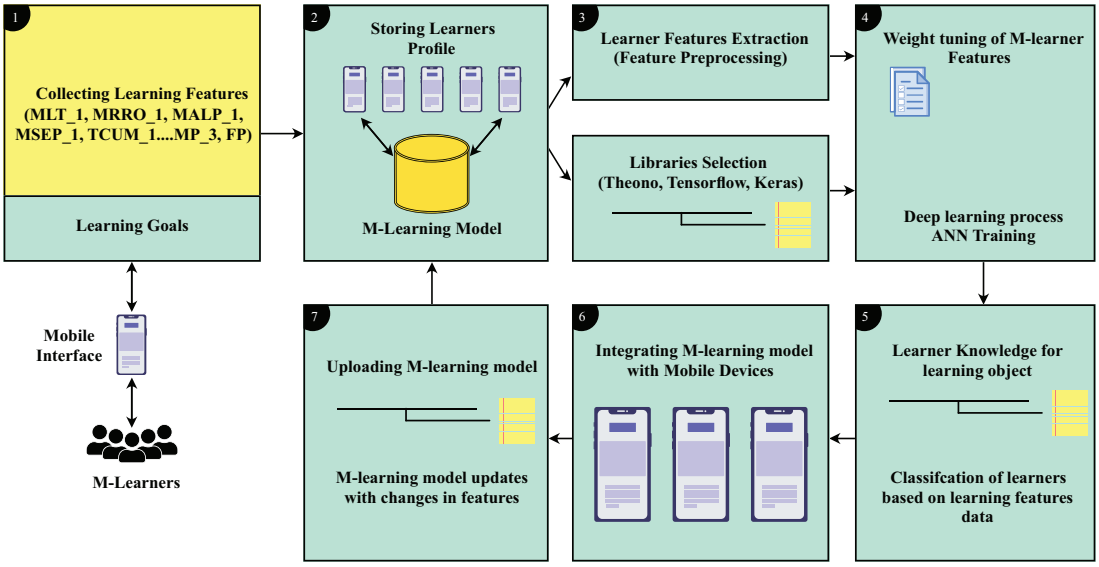


Figure 1. The architecture of the proposed m-learning system

3.3. Splitting feature instances into train and test sets

Training and testing sets were made on learners' dataset with 85% to 15% (85:15) ratios. The data in the training set was used to train the ANN-based m-learning model. For accurate results, we paired the input data with expected output data. The testing set data was used to evaluate the performance and accuracy of the m-learning model. The testing set data provided us m-learning model performance on future and unseen data.

3.4. Applying the ANN to learners classification

In the fourth step of the m-learning system workflow, we used the ANN algorithm for the learners' classification. The ANN algorithm needs historical data of learners to make learners' classification and performance prediction in the future. In our case, the learners' historical data include instances of learning features stored in the online cloud. Based on these learning features, the ANN classifies learners in different groups and can predict learners' overall performance in the future. This approach is called supervised classification deep learning modeling. It is supervised because input attributes have corresponding output results. It is classification modeling because the ANN algorithm is making different learner groups based on their learning performance.

The ANN model shown in Figure 2 has 21 input neurons (for 21 feature instances) in the input layer, 2 hidden layers having 11 neurons each, and 1 output layer having 5 neurons for classifying learners' performance into 5 grades. The 21 inputs represent instances (records) of learner features that were logged on to the cloud. To map the input features to the ANN input layer we used Python Sequential class whereas Python Dense class was used for randomly initializing weights to ANN model synapses (edges) and defining numbers of hidden layers. The output of five neurons in Figure 2 represents learners' performance classified into 5-level grades.

3.5. ANN learning process

We used the following 4 steps for ANN learning process.

- Forward propagation: for feeding features dataset instances,

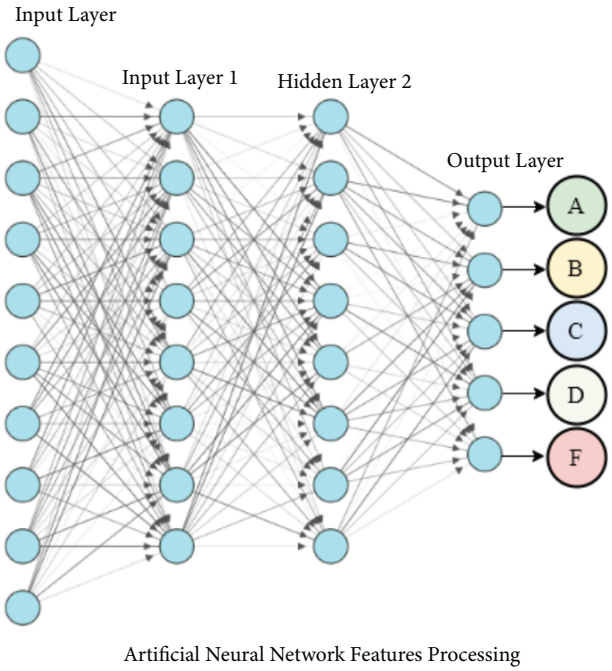


Figure 2. ANN model for classifying learners in five grades (A,B,C,D,F)

- ReLU: Rectifier activation function for neurons activation,
- Softmax function: for categorizing learners' performance into five grades in the output layer,
- Back-propagation: to mitigate the output generated errors.

3.5.1. Forward propagation

To feed dataset instances into the ANN model, we used a technique called forward propagation. Our initial data is of the form **n * 21** where ***21*** are total number of features and ***n*** are the total number of feature instances. The ANN model accepts one input record at a time but in order to speed up the operation of the ANN model, we used the matrix multiplication technique which allows multiple input records to pass through the neural network at once.

Before performing matrix multiplication, we defined two matrices X and W_1. We represented input data by matrix X having N*M dimensions where N is the number of records and M are learning features of learners. Similarly, at ANN layer one, the weights are represented by W_1 matrix having a 21*11 dimension where 21 is input nodes and 11 represent weight values on edges attached to each node (11 edges per node). Initially, weights on each edge are randomly initialized by Python Dense class. Mathematically, input data matrix X and weights matrix W_1 can be represented as

$$X = \begin{bmatrix} X1_1, X2_1, X3_1, X4_1, \dots X21_1 \\ X1_2, X2_2, X3_2, X4_2, \dots X21_2 \\ X1_3, X2_3, X3_3, X4_3, \dots X21_3 \\ \dots\dots \\ X1_n, X2_n, X3_n, X4_n, \dots X21_n \end{bmatrix} W_1 = \begin{bmatrix} W1_1, W1_2, W1_3, W1_4, \dots W1_{11} \\ W2_1, W2_2, W2_3, W2_4, \dots W2_{11} \\ W3_1, W3_2, W3_3, W3_4, \dots W3_{11} \\ \dots\dots \\ W21_1, W21_2, W21_3, W21_4, \dots W21_{11} \end{bmatrix}$$

Equation 2 shows the result of X matrix multiplied by W_1 matrix. Furthermore, multiplication of X matrix with W_1 matrix results in matrix a_2 having size of n * 11 as shown in Equation 3. By using this matrix multiple technique, multiplication feature instances would be able to pass through ANN at once.

$$a_2 = XW_1 \tag{2}$$

$$XW^1 = \begin{bmatrix} \{X1_1 * W1_1 + ... + X21_1 * W21_1\}, \{X1_1 * W1_2 + ... + X21_1 * W21_2\}, ...\{X1_1 * W1_{11} + ... + X21_1 * W21_{11}\} \\ \{X1_2 * W1_1 + ... + X21_2 * W21_1\}, \{X1_2 * W1_2 + ... + X21_2 * W21_2\}, ...\{X1_2 * W1_{11} + ... + X21_2 * W21_{11}\} \\ \{X1_3 * W1_1 + ... + X21_3 * W21_1\}, \{X1_3 * W1_2 + ... + X21_3 * W21_2\}, ...\{X1_3 * W1_{11} + ... + X21_3 * W21_{11}\} \\ ..... \\ \{X1_n * W1_1 + ... + X21_n * W21_1\}, \{X1_n * W1_2 + ... + X21_n * W21_2\}, ...\{X1_n * W1_{11} + ... + X21_n * W21_{11}\} \end{bmatrix} \tag{3}$$

During the forward propagation process, the neurons at the hidden layer one perform two tasks. First, each input in matrix X is multiplied by the weight of the corresponding synapses and then added together with other results from each neuron. A single neuron adds the results of those synapses that are connected to itself. Secondly, a neuron at layer one also performs activation function. In our ANN model, we used the Rectifier Activation Function (ReLU) to apply activation on each entry in the matrix a_2. After applying ReLU activation function to Equation 2, our new equation becomes:

$$z_2 = f(a_2), \tag{4}$$

where f is ReLU activation function. The result of the activation process is stored in the new matrix z_2. The size of z_2 matrix is same as a_2. This is because the activation function is applied independently on each entry in the matrix a_2. The ReLU activation function is performed by all hidden neurons at the hidden layer one of the ANN model. To complete the forward propagation process, we propagated z_2 all the way to output layer result, namely, $\hat{y}$, which is the ANN model predicted value for learner performance. The operation of the hidden layer two neurons is the same as that of the hidden layer one neurons. First, the outputs of hidden layer one neurons i.e. z_2 matrix are multiplied by the hidden layer one weights matrix using the matrix multiplication technique. The dimension of z_2 is n*11 and the dimension of W_2 matrix is 11*11 where 11 represents synapses weights and neurons at hidden layer one. z_2 and W_2 matrices are multiplied in the same way as matrices XW_1. The multiplication result of z_2 and W_2 matrices yields a_3 having a size of n*11 and mathematically can be represented by the following equation:

$$a_3 = z_2W_2 \tag{5}$$

At hidden layer two, the same ReLU activation function is applied to each entity of matrix a_3 resulting in new matrix z_3 having the same dimensions as a_3. This equation can be written as:

$$z_3 = f(a_3) \tag{6}$$

The resultant matrix z_3 at hidden layer two is further multiplied by a corresponding weight matrix and cumulative results of eleven matrices at hidden layer 2 are added up at the output layer. The activity at output layers yields a_4 which is z_3 matrix (11 matrices for 11 neurons at hidden layer 2) multiplied by weight matrix at hidden layer 2. At the output layer, Softmax function is applied to the a_4 matrix for learners' performance grades classification which yields:

$$z_4 = f(a_4). \tag{7}$$

Here z_4 is nothing but the predicted learner performance $\hat{y}$. For the accuracy of the m-learning model based on ANN, we minimized the difference between predicted performance value $\hat{y}$ and actual performance value y. For minimizing the difference between $\hat{y}$ and y, we used cost function $\boldsymbol{C}$ to know how much difference there is between our predicted result and the actual result.

3.5.2. Training m-learning model: back-propagation

Training the m-learning model is nothing but minimizing the cost function. Cost function also revealed how wrong or costly our predicted result was based on input features. We expressed the overall cost function in the following equation:

$$C = \Sigma\, 1/2(y - \hat{y})^2 \tag{8}$$

For minimizing the cost function we had two options; 1) changing the input data, 2) adjusting weights on edges. We did not have control over changing the input data, so the only choice left for us was changing the weights on synapses. To reduce time and computation resources, we used a technique called stochastic gradient descent to determine the suitable weights for synapses. Stochastic gradient descent is more impressive when we have high dimension data and we are trying to find suitable weight values for minimum cost. To perform stochastic gradient descent, we need an equation that involves weights across ANN that are responsible for generated error $\boldsymbol{C}$. Looking across the ANN model we observed that the weights are spread across three matrices W_1, W_2, W_3. We will be calculating $d(C)/d(W_1)$, $d(C)/d(W_2)$, and $d(C)/d(W_3)$ independently resulting in the same numbers of gradient values as number of weights. Here the derivative of $\boldsymbol{C}$ with respect to W_1, W_2, and W_3 will determine how much these weight values are responsible for generating cost or error in $\boldsymbol{C}$ ($\boldsymbol{C}$ is the difference between $\hat{y}$ and y).

First we calculated $d(C)/d(W_3)$ using equation 9. Summation in this equation adds errors from hidden layer two weights and then adds them up to find the overall cost generated by weights in hidden layer two.

$$\frac{d(C)}{d(W_3)} = \frac{d(\Sigma \frac{1}{2}(y - \hat{y})^2)}{d(W_3)}. \tag{9}$$

While solving the overall cost, we will take advantage of sum rule in differentiation, which implies that the derivative of sums is the same as the sum of the derivatives. We moved summation Σ outside and calculated the derivative of inside expression. Thus, equation 9 can be written as:

$$\frac{d(C)}{d(W_3)} = \Sigma \frac{d(\frac{1}{2}(y - \hat{y})^2)}{d(W_3)}.$$

Taking the derivative of the above equation results:

$$\frac{d(C)}{d(W_3)} = \Sigma - (y - \hat{y}) \frac{d(\hat{y})}{d(W_3)}.$$

For clarity, we will ignore summation as shown in equation 10. Once we calculate the derivatives for all synapses weights, we will add them up to find the overall cost generated by weights.

$$\frac{d(C)}{d(W_3)} = -(y - \hat{y}) \frac{d(\hat{y})}{d(W_3)} \tag{10}$$

Here y is our performance score result, which will not change, acting as a constant term. Thus, the derivative of y will be zero. On the other hand, $\hat{y}$ changes with W_3 (hidden layer two weights), so we will apply chain rule and multiply Equation 10 by $d(\hat{y})/d(W_3)$.

$$\frac{d(C)}{d(W_3)} = -(y-\hat{y})\frac{d(\hat{y})}{d(z_4)}\frac{d(z_4)}{d(W_3)}$$

$$\frac{d(C)}{d(W_3)} = -(y-\hat{y})f^{'}(z_4)\frac{d(z_4)}{d(W_3)} \tag{11}$$

We also need the rate of change of $\hat{y}$ with respect to z_4, so derivative of ReLU activation with respect to z_4 yields

$$f(z) = \frac{1}{1+e^{-z}} \implies f^{'}(z) = \frac{e^{-z}}{(1+e^{-z})^2}$$

The weights that are more responsible for error generation will change more when the gradient descent process is applied to them. We already know that $d(z_4)/d(W_3)$ is the rate of change of $d(z_4)$ with respect to $d(W_3)$. Each value in back-propagating error i.e. δ^3 needs to be multiplied by each activity within neuron. We can achieve this by taking the transpose of a_3 and then multiplying it by δ^3.

$$\frac{d(C)}{d(W_3)} = (a_3)^T\delta^{(3)} \quad \text{where} \quad a_3 = z_2 * W_3$$

$$\frac{d(C)}{d(W_3)} = (a_3)^T\delta^{(3)} \quad \text{where} \quad \delta^{(3)} = -(y-\hat{y})f^{'}(z_4) \tag{12}$$

Equation 11 represents the rate of change of z_4 with respect to hidden layer 2 weights. Now we calculate how much weights at hidden layer two are responsible for overall generated error/cost. In this backpropagation method, we are back disseminating the error to each weight. The weight that is responsible for more error generation will have larger activation and will result in larger $d(C)/d(W)$ values.

Next, we calculated $d(C)/d(W_2)$, the rate of change of $d(C)$ (cost) with respect to W_2. However, for computing $d(C)/d(W_2)$, first we need to determine $d(z_4)/d(W_2)$. By chain rule multiplication we can write:

$$\frac{d(z_4)}{d(W_2)} = \frac{d(z_4)}{d(a_3)}\frac{d(a_3)}{d(W_2)}$$

$$\frac{d(C)}{d(W_2)} = \delta^{(3)}(a_3)^T\frac{d(a_3)}{d(W_2)}$$

$$\frac{d(C)}{d(W_2)} = \delta^{(3)}(a_3)^T\frac{d(a_2)}{d(z_2)}\frac{d(a_2)}{d(W_2)}$$

$$\frac{d(C)}{d(W_2)} = \delta^{(3)}(a_3)^Tf^{'}(z_2)\frac{d(z_2)}{d(W_2)}$$

$$\frac{d(C)}{d(W_2)} = X^T\delta^{(3)}(a_3)^Tf^{'}(z_2)$$

or

$$\frac{d(C)}{d(W_2)} = X^T\delta^{(2)} \quad \text{where} \quad \delta^{(2)} = \delta^{(3)}(a_3)^Tf^{'}(z_2) \tag{13}$$

In the last step of the back-propagation process, we calculated $d(C)/d(W_1)$, layer one weights responsible for generating errors in the results of the output layer.

$$\frac{d(z_3)}{d(W_1)} = \frac{d(z_3)}{d(a_2)}\frac{d(a_2)}{d(W_1)}$$
$$\frac{d(C)}{d(W_1)} = \delta^{(2)}\delta^{(3)}(W_2)^T\frac{d(a_2)}{d(W_1)}$$
$$\frac{d(C)}{d(W_1)} = \delta^{(2)}\delta^{(3)}(W_2)^T\frac{d(a_2)}{d(z_2)}\frac{d(a_2)}{d(W_1)}$$
$$\frac{d(C)}{d(W_1)} = \delta^{(2)}\delta^{(3)}(W_2)^T f^{'}(z_2)\frac{d(z_2)}{d(W_1)}$$
$$\frac{d(C)}{d(W_1)} = X^T\delta^{(2)}\delta^{(3)}(W_2)^T f^{'}(z_2)$$

or

$$\frac{d(C)}{d(W_1)} = X^T\delta^{(1)} \quad \text{where} \quad \delta^{(1)} = \delta^{(2)}\delta^{(3)}(W_1)^T f^{'}(z_2) \tag{14}$$

The objective of the ANN model is to learn what features affect the final performance of learners. The ANN model creates a relationship between the independent and dependent features of each learner. Initially, on training data, it may not find the proper relationship between independent and dependent features. Therefore, the ANN model has to be involved in the learning process where if the predicted output is not near to actual output, the error (the difference between y and $\hat{y}$) is back-propagated to the entire ANN. The equation 14 tells us the amount of error to be back-propagated to the neural network and the difference in cost between the actual output (y) and predicted output $\hat{y}$. The lower we get the value of cost, the smaller will be the difference between y and $\hat{y}$. Every time the error is back-propagated, synapses weights get updated. The process of back-propagation and synapses weights updation continues until the predicted output is equal or very near to actual output.

4. Implementation of m-learning model using ANN

We trained the ANN-based m-learning model by first randomly initializing the weights of the synapses to numbers close to zero (but not zero) using Keras Dense class. Then we used ANN forward propagation technique to input the first observation from the dataset using Keras Sequential class. In the forward propagation technique, the neurons are activated in such a way that the impact of each neuron activation is restricted by the weights of the synapses. We performed a multi-class classification of learners' final grades using Keras API which corresponds to stage five of m-learning system workflow. Table 2 shows five-level classification scheme of learner grades while the pie chart in Figure 3 shows the grades percentage of learners in five classes.

Table 2. Five-level performance classification scheme.

Final performance classification	Very good	Good	Average	Satisfactory	Fail
Earned points	18–20[a]	15–17	12–14	9–11	0–8
Grades	A	B	C	D	F
a, the number is inclusive in the range					

The goal of performing multiclass classification was to construct the m-learning model which, when given a new dataset, will correctly predict the grade in which the new learner belongs to. The Keras API provided fully-configurable modules that can be plugged together with as little restrictions as possible. In

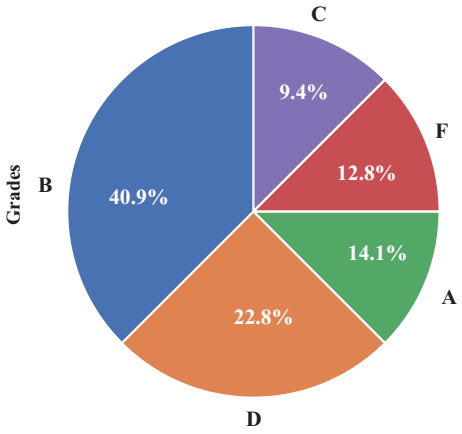


Figure 3. Grade percentages of learners in five classes.

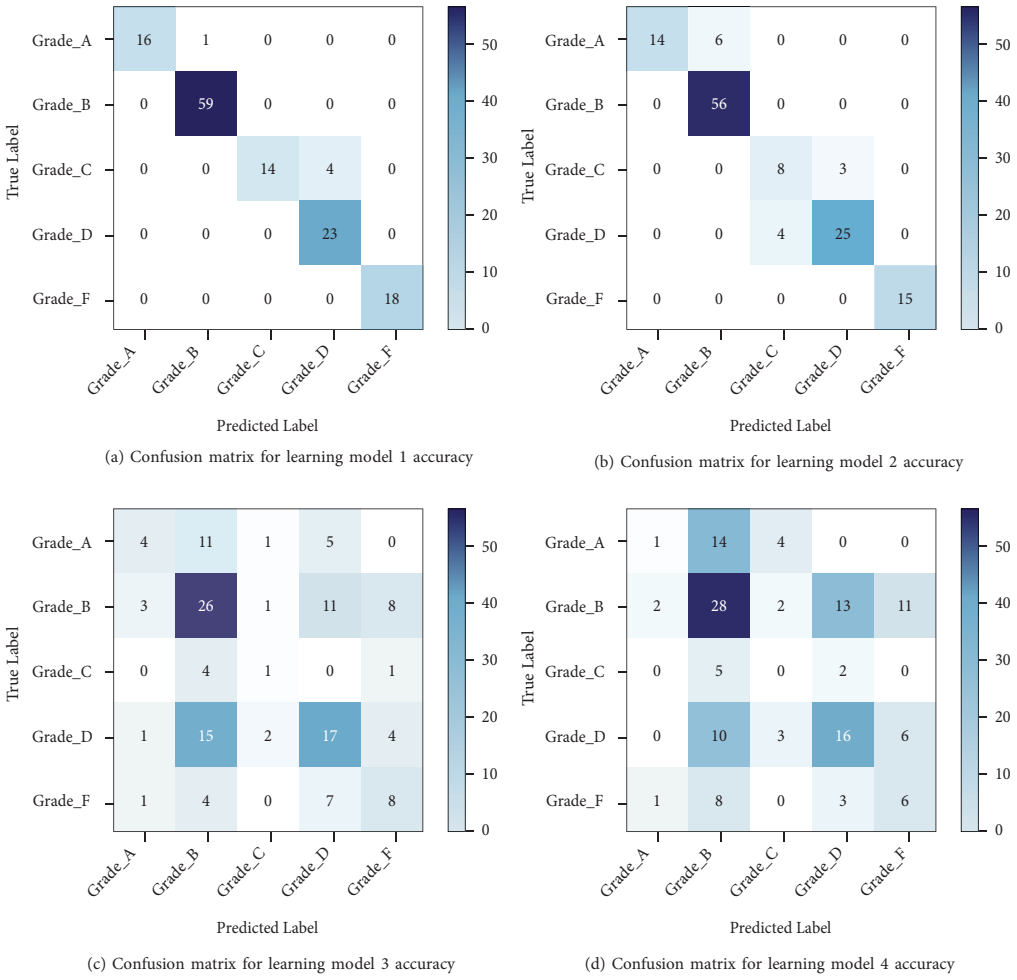
particular, Keras API provided the ANN cost function optimizers, initialization schemes, activation functions, and regularization schemes modules that we combined together to create the m-learning model. We performed a multiclass classification using four m-learning models as mentioned below:

Learning model 1: This model contains all the features from the dataset except the final performance.
Learning model 2: This model is similar to model 1 but without the module 3 performance feature.
Learning model 3: This model is similar to model 2 but without the module 2 performance feature.
Learning model 4: This model is similar to model 3 but without the module 1 performance feature.

Before creating learning model 1, we performed dataset scaling and cleaning and split the dataset into train and test sets. We used all the features in learning model 1 to predict the final grades. Learning model 1 is composed of an input layer having 21 neurons (for input features), 2 hidden layers with 11 neurons each and 1 output layer having 5 neurons. The five output neurons correspond to the five-level grading scheme. We used ReLU (Rectifier Linear Unit) as the activation function in the input and hidden layers whereas the Softmax activation function was used in the output layer to predict learners' five-level categorical grades. Lastly, we compiled the learning model 1 with 200 epochs and a batch size of 5. After compiling learning model 1 we achieved an accuracy of 96.30%. For better analysis and visualization we constructed a confusion matrix on predicted grade scores and actual grade scores as shown in Figure 4.1. By analyzing the confusion matrix, we concluded that 130 grades out of 135 were predicted correctly and the remaining five grades were classified incorrectly. Similarly, we constructed learning models 2, 3, and 4 and after the compilation achieved an accuracy of 87.41%, 83.70%, and 60.74% respectively. The confusion matrices for learning models 2, 3, and 4 are also shown in Figures 4.2–4.4. The diagonal elements in confusion matrices are those elements that are predicted accurately. They are called true positive. The remaining elements that are out of diagonal elements are predicted incorrectly. We can see that the accuracy of learning models decreases as we remove module scores from independent features. This means that module scores are an important factor in predicting the final learning performance of learners.

5. Feature weights and importance

We used a random forest ensemble method to determine each feature weight and importance in predicting the output of the m-learning models i.e. the final grades. Random forest ensemble method gives robustness,



(a) Confusion matrix for learning model 1 accuracy

(b) Confusion matrix for learning model 2 accuracy

(c) Confusion matrix for learning model 3 accuracy

(d) Confusion matrix for learning model 4 accuracy

Figure 4. Confusion matrices for learning models

accuracy, and control over the overfitting problems. Random forest works by running multiple decision trees and aggregating their output for result prediction. Looking at features' relative weights and importance gives a good knowledge of which features contribute most in predicting model outputs. These important features can help us in making the m-learning model more adaptive and riveting. Integrating the importance of features in m-learning systems leads to the development of a comprehensive and complete system that knows how to guide m-learners in the learning process.

Using the random forest ensemble method, we first split our dataset into subsets and built decision trees on them. We continue to build different trees on dataset subsets until we come up with the relationship of all independent features with dependent feature (i.e. final grades). Table 3 shows the independent features along with their weight-values (importance) in predicting the target-dependent feature while Figure 5 illustrates independent features with their importance.

The graph in Figure 5 shows that m-learners performance in modules 1, 2, 3 and their learning time duration in all three modules have highest weights in predicting the final performance score. The graph also shows that m-learners use text and video content equally while learning. Interestingly, the graph also shows

Table 3. Relative weights of independent features for ensemble method.

Feature	Feature weight	Feature	Feature weight	Feature	Feature
MLT_1	0.047015	MLT_2	0.039092	MLT_3	0.033671
MRRO_1	0.004964	MRRO_2	0.002833	MRRO_3	0.005588
MALP_1	0.008757	MALP_2	0.006129	MALP_3	0.010782
MSEP_1	0.013866	MSEP_2	0.005301	MSEP_3	0.007375
TCUM_1	0.005581	TCUM_2	0.009339	TCUM_3	0.007387
VCUM_1	0.013831	VCUM_2	0.00296	VCUM_3	0.004365
MP_1	0.202859	MP_2	0.194152	MP_3	0.082769
FP	0.291383				

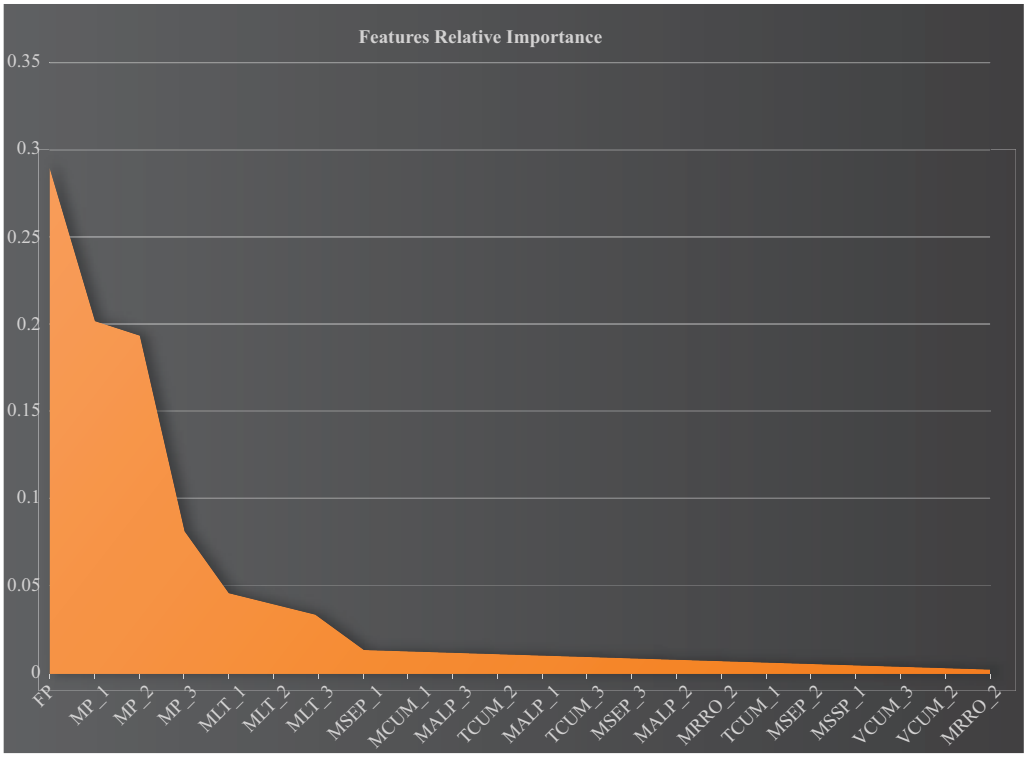


Figure 5. Features' relative weights and importance in predicting learning performance.

that m-learners use mobile phones for learning in academic as well as in entertainment/social places. This suggests that during learners' free time span they may utilize mobile phones for learning purposes irrespective of physical location type.

6. Early intervention experiment

In the last stage of our research, we performed early intervention experiment on those learners who obtained a score of 11 and less than 11 in the final performance quiz (learners having D and F grades). The objective of the early intervention experiment was to reveal whether early intervention in learners' study process improves their performance or not. In this experiment, the learners were motivated to improve their grades and make

m-learning model predictions false. A total of 318 learners having grades D and F were selected. Learners were randomly divided into two groups of the same size (159 learners in each group): Control group learners were independent of early intervention and received normal programming contents and exercises whereas experimental group learners were intervened during m-learning activities through motivational messages, adaptive contents, and adaptive navigational paths. The experiment lasted for one month. The experimental group learners were motivated to increase their learning performance through the following measures and interventions:

- Sending adaptive triggers (messages) on mobile devices based on learners preferences. The purpose of adaptive triggers was to increase learners' study capacity and aid them in learning important points. An example of an adaptive trigger is: Please try to revise the current topic and make sure that you have mastered it.
- Sending motivational triggers on mobile devices based on learners' performance. The aim of motivational triggers was to increase student motivation toward learning. The factors of fear, hope, and suggestion were included in motivation triggers for increasing learners' inspiration towards learning. An example of motivational trigger having a fear factor is: You may never become a good programmer if you continue to show below-average performance. An example of motivational trigger having hope factor is: Two-hour programming practice will get you top grades. Example of motivation trigger having a suggestion factor is: Please refer to LMS to new post uploaded by the instructor regarding programming.

6.1. Early intervention experiment results

After one month, both the control and experimental group learners' results were analyzed. Figure 6 shows the performance of both groups on the line chart. From the line chart, it is concluded that intervened learners shows overall high performance than un intervened learners. The statistical results revealed that intervened learners show an overall 12% higher performance than unintervened students. Moreover, the results indicated that through proper guidance and intervention, the learning performance of m-learners could be improved and enhanced.

6.2. Early intervention experiment discussion

Our learner model was successful in identifying how the key features of m-learners contribute to predicting their final performance. The learner model can identify learners with different performances because it has learned from existing learners' data. Based on the learner's performance features, our m-learning model will also output the weight of each feature and how important each feature is in predicting the final performance of learners.

Based on the aforementioned facts, our m-learning model can also predict whether new m-learners will show good or bad performance. This time we do not have information about new learners; however, we do have m-learning model that has learned from previous m-learners' performance. The new learner's features along with their respective weights are fed into a learner model and the m-learning model would predict the output performance categories of new learners. Subsequently, we can take preventive measures on learners who are unlikely to show good performance.

7. Conclusions and future work

The contribution of this research is the generation of the m-learning model from learners' features that can be used in an adaptive learning environment. Each learner has tailored m-learning model determined by learners' weighted features.

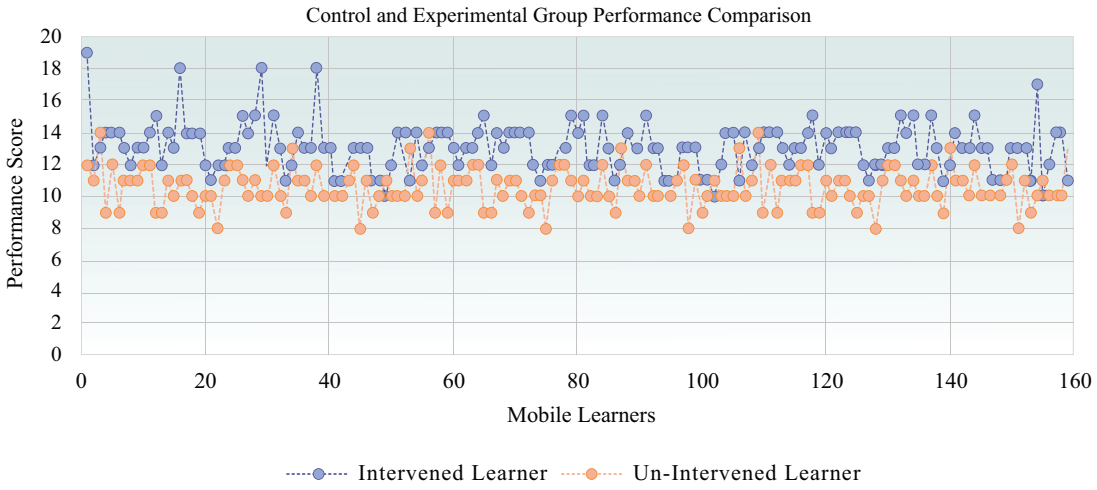


Figure 6. Control and experimental groups performance.

The ANN algorithm establishes the m-learning model by developing a relationship between m-learners input features and learner performance. After training the m-learning model, the multiclass classification was performed and the testing process was conducted on four models, i.e. learning model 1, learning model 2, learning model 3, and learning model 4. For these four learning models we achieved the corresponding accuracy of 96.30%, 87.41%, 83.70%, and 60.74%, respectively. We observed that the accuracy of learning models decreases as we remove module score features from independent input features. The testing process showed that among all independent features, module score features are the most important features in predicting the final learning performance of learners. The proposed m-learning model is then deployed on learners' mobile devices for delivering adaptive learning contents and suggestions to learners. Additionally, the results of the early intervention experiment revealed that m-learning model can efficiently increase the learning performance of m-learners through adaptive and motivational triggers.

In the future, we plan to integrate our proposed m-learning model with learning management system to test its effectiveness on a larger scale in order to reduce the drop-out rate of learners and improve their learning performance. Moreover, other deep neural network algorithms such as recurrent neural network, self-organizing maps, and deep autoencoder algorithm can be combined with our proposed m-learning model for potential improvements.

References

[1] Mukhopadhyay S. Advanced Data Analytics Using Python: With Machine Learning, Deep Learning and NLP Examples. Apress, NY, USA: Springer, 2018. doi: 10.1007/978-1-4842-3450-1

[2] Schmidhuber J. Deep learning in neural networks: an overview. Neural networks 2015; 61 : 85-117. doi: 10.1016/j.neunet.2014.09.003

[3] Hsu T, Chiou C, Tseng J, Hwang G. Development and evaluation of an active learning support system for context-aware ubiquitous learning. IEEE Transactions on Learning Technologies 2015; 9 (1): 37-45. doi: 10.1109/TLT.2015.2439683

[4] Virtanen M, Haavisto E, Liikanen E, Kääriäinen M. Ubiquitous learning environments in higher education: a scoping literature review. Education and Information Technologies 2018; 23(2): 985-998. doi: 10.1007/s10639-017-9646-6

[5] Amara S, Macedo J, Bendella F, Santos A. Group formation in mobile computer supported collaborative learning contexts: a systematic literature review. Journal of Educational Technology & Society 2016; 19 (2): 258-273.

[6] Hsu Y, Ching Y. A review of models and frameworks for designing mobile learning experiences and environments. Canadian Journal of Learning and Technology 2015; 14 (3): 12-24. doi: 10.21432/T2V616

[7] Sun G, Shen J. Facilitating social collaboration in mobile cloud-based learning: A teamwork as a service (TaaS) approach. IEEE Transactions on Learning Technologies 2014; 7 (3): 207-220. doi: 10.1109/TLT.2014.2340402

[8] Jadhav S, Channe H. Comparative study of K-NN, naive Bayes and decision tree classification techniques. International Journal of Science and Research (IJSR) 2016; 5 (1): 1842-1845. doi: 10.21275/v5i1.NOV153131

[9] Do CB, Batzoglou S. What is the expectation maximization algorithm? Nature Biotechnology 2008; 26 (8): 897. doi: 10.1038/nbt1406

[10] Suthaharan S. Support vector machine. In: Ramesh S (editor). Machine learning models and algorithms for big data classification. Boston, MA, USA: Springer, 2016, pp. 207-235. doi: 10.1007/978-1-4899-7641-3_9

[11] Adeniyi D, Wei Z, Yongquan Y. Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method. Applied Computing and Informatics 2016; 12 (1): 90-108. doi: 10.1016/j.aci.2014.10.001

[12] Ciolacu M, Tehrani A, Beer R, Popp H. Education 4.0 Fostering student's performance with machine learning methods. In: 2017 IEEE 23rd International Symposium for Design and Technology in Electronic Packaging (SIITME); Constanta, Romania; 2017. pp. 438-443. doi: 10.1109/SIITME.2017.8259941

[13] Baradwaj B, Pal S. Mining educational data to analyze students' performance. International Journal of Advanced Computer Science and Applications 2012; 2 (6): 63-69. doi: 10.14569/IJACSA.2011.020609

[14] Nanda S, Panda G. Design of computationally efficient density-based clustering algorithms. Data & Knowledge Engineering 2015; 95: 23-38. doi: 10.1016/j.datak.2014.11.004

[15] Le QV, Ngiam J, Coates A, Lahiri A, Prochnow B et al. On optimization methods for deep learning. In: Proceedings of the 28th International Conference on International Conference on Machine Learning; Bellevue, Washington, USA; 2011. pp. 265-272.

[16] Yosinski J, Clune J, Bengio Y, Lipson H. How transferable are features in deep neural networks? In: Michael J, Yann L, Sara S (editors). Advances in Neural Information Processing Systems. London, UK: The MIT Press, 2014, pp. 3320-3328.

[17] Bengio Y, Lamblin P, Popovici D, Larochelle H. Greedy layer-wise training of deep networks. In: Proceedings of the 20th International Conference on Neural Information Processing Systems; British Columbia, Canada; 2007. pp. 153-160.

[18] Li H, Wang X, Ding S. Research and development of neural network ensembles: A survey. Artificial Intelligence Review 2018; 49 (4): 455-79. doi: 10.1007/s10462-016-9535-1

[19] Aljarah I, Faris H, Mirjalili S. Optimizing connection weights in neural networks using the whale optimization algorithm. Soft Computing 2018; 22 (1): 1-5. doi: 10.1007/s00500-016-2442-1

[20] Bara M, Ahmad N, Modu M, Ali H. Self-organizing map clustering method for the analysis of e-learning activities. In: IEEE 2018 Majan international conference (MIC); Muscat, Oman; 2018. pp. 1-5. doi: 10.1109/MINTC.2018.8363155

[21] Almeida A, Azkune G. Predicting human behaviour with recurrent neural networks. Applied Sciences 2018; 8 (2): 305. doi: 10.3390/app8020305

[22] Thai-Nghe N, Horváth T, Schmidt-Thieme L. Factorization models for forecasting student performance. In: The 4th International Conference on Educational Data Mining; Eindhoven, The Netherlands; 2011. pp. 11-20. doi: 10.1109/ICALT.2011.130

[23] Pardos ZA, Heffernan NT. Modeling individualization in a bayesian networks implementation of knowledge tracing. In: Springer 2010 International Conference on User Modeling, Adaptation, and Personalization; Berlin, Heidelberg; 2010. pp. 255-266. doi: 10.1007/978-3-642-13470-8_24

[24] Al-Emran M, Elsherif H, Shaalan K. Investigating attitudes towards the use of mobile learning in higher education. Computers in Human behavior 2016; 56: 93-102. doi: 10.1016/j.chb.2015.11.033

[25] Cerna M. Modified recommender system model for the utilized eLearning platform. Journal of Computers in Education 2019; 1-25. doi: 10.1007/s40692-019-00133-9

[26] Prieto J, Migueláñez S, García-Peñalvo F. Understanding mobile learning: devices, pedagogical implications and research lines. Teoría de la Educación. Educación y Cultura en la Sociedad de la Información 2014; 15 (1): 20-42.

[27] Nordin N, Embi M, Yunus M. Mobile learning framework for lifelong learning. Procedia-Social and Behavioral Sciences 2010; 7: 130-8. doi: 10.1016/j.sbspro.2010.10.019

[28] Sharples M, Corlett D, Westmancott O. The design and implementation of a mobile learning resource. Personal and Ubiquitous Computing 2002; 6 (3): 220-234. doi: 10.1007/s007790200021

[29] Schuck S, Kearney M, Burden K. Exploring mobile learning in the third space. Technology, Pedagogy and Education 2017; 26 (2): 121-137. doi: 10.1080/1475939X.2016.1230555

[30] Christensen R, Knezek G. Readiness for integrating mobile learning in the classroom: challenges, preferences and possibilities. Computers in Human Behavior 2017; 76: 112-21. doi: 10.1016/j.chb.2017.07.014

[31] Dennen V, Burner J, Cates L. Information and communication technologies, and learning theories: putting pedagogy into practice. In: Voogt J, Knezek G, Christensen R, Lai KW (editors). Second Handbook of Information Technology in Primary and Secondary Education. Springer, Basel, Switzerland: Springer International Publishing, 2018, pp. 143-60. doi: 10.1007/978-3-319-71054-9_9

[32] Martins C, Faria L, De Carvalho C, Carrapatoso E. User modeling in adaptive hypermedia educational systems. Educational Technology & Society 2008; 11 (1): 194-207.

[33] Albert M, Kording K, Herrmann M, Jayaraman A. Fall classification by machine learning using mobile phones. PloS one 2012; 7 (5): e36556. doi: 10.1371/journal.pone.0036556

[34] O'Mahony MP, Smyth B. A Classification-Based Review Recommender. Research and Development in Intelligent Systems XXVI. Springer-Verlag London: Springer, 2010.

[35] Portugal I, Alencar P, Cowan D. The use of machine learning algorithms in recommender systems: A systematic review. Expert Systems with Applications 2018; 97: 205-227. doi: 10.1016/j.eswa.2017.12.020

[36] Martins C, Faria L, De Carvalho V, Carrapatoso E. User modeling in adaptive hypermedia educational systems. Educational Technology & Society 2008; 11(1): 194-207.

[37] Hussain J, Hassan AU, Bilal HS, Ali R, Afzal M et al. Model-based adaptive user interface based on context and user experience evaluation. Journal on Multimodal User Interfaces 2018; 12 (1): 1-6. doi: 10.1007/s12193-018-0258-2

[38] Abela C, Staff C, Handschuh S. Task-based user modelling for knowledge work support. In: Springer 2010 International Conference on User Modeling, Adaptation, and Personalization; Big Island, HI, USA; 2010. pp. 419-422. doi: 10.1007/978-3-642-13470-8_44

[39] Guo B, Zhang R, Xu G, Shi C, Yang L. Predicting students performance in educational data mining. In: IEEE 2015 International Symposium on Educational Technology (ISET); Wuhan, China; 2015. pp. 125-128.

[40] Meng Q, Catchpoole D, Skillicom D, Kennedy PJ. Relational autoencoder for feature extraction. In: IEEE 2017 International Joint Conference on Neural Networks (IJCNN); Anchorage, Alaska; 2017. pp. 364-371. doi: 10.1109/IJCNN.2017.7965877

[41] Kurilovas E. Advanced machine learning approaches to personalise learning: Learning analytics and decision making. Behaviour & Information Technology 2019; 38(4): 410-21. doi: 10.1080/0144929X.2018.1539517

[42] Kahraman HT, Sagiroglu S, Colak I. The development of intuitive knowledge classifier and the modeling of domain dependent data. Knowledge-Based Systems. 2013; 37: 283-95. doi: 10.1016/j.knosys.2012.08.009

[43] Piech C, Bassen J, Huang J, Ganguli S, Sahami M et al. Deep knowledge tracing. In: Proceedings of the 28th Annual Conference of the Advances in neural information processing systems; Montreal, Canada; 2015. pp. 505-513.

[44] Xiong X, Zhao S, Van Inwegen EG, Beck JE. Going deeper with deep knowledge tracing. In: Proceedings of the 9th International Conference on Educational Data Mining (EDM); Raleigh, North Carolina; 2016. pp. 545-550.