IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 19, 2020, accepted July 1, 2020, date of publication July 7, 2020, date of current version July 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3007727

Improving M-Learners' Performance Through Deep Learning Techniques by Leveraging Features Weights

MUHAMMAD ADNAN[1], ASAD HABIB[1], JAWAD ASHRAF[1], BABAR SHAH[2], AND GOHAR ALI[3]

[1]Institute of Computing, Kohat University of Science and Technology (KUST), Kohat 26000, Pakistan
[2]College of Technological Innovation, Zayed University, Abu Dhabi, United Arab Emirates
[3]Department of Information Systems and Technology, Sur University College, Sur 411, Oman

Corresponding author: Muhammad Adnan (adnan@kust.edu.pk)

This research was funded by RIF grant activity code R19045 of Zayed University, Abu Dhabi, UAE.

ABSTRACT Mobile learning (M-learning) has gained tremendous attention in the educational environment in the past decade. For effective M-learning, it is important to create an efficient M-learning model that can identify the exact requirements of mobile learners (M-learners). M-learning model is composed of features that are generated during M-learners' interaction with mobile devices. For an adaptive M-learning model, not only learning features are required, but it is also important to determine how they differ for various M-learners, their weights, and interrelationship. This study proposes a robust and adaptive M-learning model that is based on machine learning and deep learning (ML/DL) techniques. The proposed M-learning model dynamically explores learning features, their corresponding weights, and association for M-learners. Based on learning features, the M-learning model categorizes M-learners into different performance groups. The M-learning model then provides adaptive content, suggestions, and recommendations to M-learners in order to make learning adaptive and stimulating. For comparative analysis, the prediction accuracy of five baseline ML models was compared with the deep Artificial Neural Network (deep ANN). The results demonstrated that deep ANN and Random Forest (RF) models exhibited better prediction accuracy. Subsequently, both models were selected for developing the M-learning model which included the performance categorization of M-learners under a five-level classification scheme and assigning weights to various features for providing adaptive help and support to M-learners. Our explanatory analysis has shown that behavioral features besides contextual features also influence the learning performance of M-learners. As a direct outcome of this research, more efficient, interactive, and useful mobile learning applications can be developed that accurately predict learning objectives and requirements of diverse M-learners thus helping M-learners in enhancing their study behavior.

INDEX TERMS Deep neural networks, deep learning, machine learning, learners' classification, early engagement, adaptive M-learning, feature weights.

I. INTRODUCTION

Mobile devices have become an integral part of life and society. A current-day challenge is to make Mobile learning (M-learning) adaptive for those who use mobile devices for learning purposes. For making M-learning effective, contextual and behavioral features of individual learners have to be considered. Contextual features include learning time, background knowledge, and learners' preferences, etc. whereas behavior features include M-learners interaction behavior with mobile devices e.g. discussion group participation, preferred learning content types, problems posted and learning performance, etc. The M-learning features are important for the input, processing, and output of the M-learning model. For a comprehensive and operational M-learning model, these features are essential and act as fuel. Therefore, development of a M-learning model that intuitively and intelligently selects learning resources for various

The associate editor coordinating the review of this manuscript and approving it for publication was Syed Muhammad Anwar.

learners to improve their study behavior is the prime need of the modern M-learning environments.

In this research, we examine the application of M-learning model in predicting the learning performance of M-learners. The specific focus of our research includes M-learners' performance prediction, learning features weight tuning, features ranking and their interrelationship for diversified M-learners. M-learning model uses machine learning and deep learning (ML/DL) algorithms for features identification, processing, and analysis. M-learning model based on DL algorithms is capable of considering the most relevant feature by themselves, requiring little intervention and guidance by programmers. DL algorithms can analyze M-learners' features and properly classify them into various groups based on their learning performance.

The deep learning paradigm uses statistical and machine learning techniques to find feature hierarchies, weights, the hidden patterns and features relationships based on Deep Neural Networks (DNN) [1], [2]. The DNN differs from Neural Networks (NNs) in the way that they use hidden layers to find hidden patterns, modeling laws, and features ranks. The basic idea of DL allows computers to learn from the experience and apply those heuristics on the new data. The more the data and experience, the accurate the final prediction would be. Features weights and hidden patterns are mathematical which can be easily identified and analyzed by the ML/DL algorithms. Established on old data, features, and rules, the DL algorithms can implicitly predict the outcomes of new data. The accuracy of prediction and creation of rules from features is an automatic process and improves with newly obtainable features data.

Business intelligence (BI) refers to the techniques, tools, procedures, and applications responsible for data elicitation, analysis, integration and presentation for business information [3]. The interest in DL/ML techniques has increased due to advancements in information technology (IT), computers, and the Internet. These advancements have triggered the exponential growth in business centralized and distributed databases. These databases hold important information suitable for making the intelligent decisions for organization success. It is very difficult for human experts to analyze the huge amount of data continuously growing and they may overlook important business intelligence details. Hence, an alternative solution is to use ML/DL techniques to extract meaningful high-level information from raw data for timely and right decisions.

Mobile devices continuously consume and generate a huge amount of data offering fertile ground for BI. In M-learning settings, there are multiple sources of data e.g. learning management systems (LMS), online study groups, online web and database servers, etc. DL and BI techniques collectively can be used to answer several interesting questions. For example, DL and BI can tell us: which users are the M-learners? How mobile devices could be used for learning purposes? What types of learning content are liked by particular learners? Can M-learning assist the traditional learning approach? How learning performance of learners can be predicted? and how M-learning can improve learners' study performance? The focus of this research article is to provide suitable answers to these questions. Modeling the M-learning behavior of learners is important for both learners and developers since it can help in a better understanding of the user experience and ultimately improve it.

The primary challenge in creating the M-learning model is to decide which learning features best represent the learning behavior of learners and how to store and use them for input to ML/DL algorithms. Proper learning features are important for efficiently modeling the learner's understanding and for providing discerning information to M-learning systems [4]. The performance of M-learning systems is directly affected by the right learning features. The other important challenge is to decide how to guide M-learners in their learning process once their features are analyzed and weighted. Moreover, providing tailored learning content to the learners based on their learning preferences and inclinations is a significant need for M-learning environments.

For the last two decades, different ML/DL algorithms have been developed, evaluated and their performance explored in online and M-learning settings [5], [6]. It is crucial to decide which type of ML/DL algorithm to choose for modeling the learning behavior of M-learners as proper learning algorithm increases/decreases the response time of the M-learning system [7], [8]. The right algorithm also affects the overall performance of the M-learning system. For instance, Naïve Bayes and Expectation Maximization are probability estimation algorithms. Their performance is excellent in producing efficient and correct results on training and testing datasets but they can be quite expensive to implement [9]. Computation of conditional probability on every hypothesis can be quite costly in terms of time and software resources. Therefore, other types of ML algorithms are needed to create an M-learning model. ML algorithms like K-means, Decision Trees (DT), K-Nearest Neighbors (KNNs), Support Vector Machine (SVM) and Density-based Spatial Clustering of Applications with Noise (DBSCAN), etc. use classification and clustering techniques [10], [11]. These algorithms can accurately classify and cluster a small amount of dataset. They are computationally and financially easier to implement and interpret but the drawback of these algorithms is that they need complex features engineering processes. They do not scale with an increase in data and do not report the best results in terms of performance and accuracy. For example, in mobile and online learning settings, data related to learning features is huge and changes frequently while these algorithms are best for static features and a small amount of data. The other disadvantages of traditional ML algorithms are that they are complex, need domain expertise and a lot of human interventions.

In stark contrast to the ML algorithms, the DL algorithms use layers to create an artificial neural network similar to the human brain network. With neurons processing inside each layer, DL algorithms can learn and make decisions on their

own without human intervention. They can represent data at different levels of granularity thus they intrinsically have a greater level of flexibility and robustness. DL algorithms are also ahead of the classical ML algorithm due to their performance and accuracy when trained and tested on a huge amount of data [12]. One of the prime advantages of DL algorithms over traditional ML algorithms is their capability to excerpt abstract features information from low-level data in an incremental fashion [13]. This technique eliminates the hardcore features engineering process, human intervention, and domain expertise. For example, DL algorithms can automatically discover new features to be used for classification while for ML algorithms, new features have to be provided manually.

In this research, we analyzed the features of M-learners using the Deep Artificial Neural Network (Deep ANN). The features of M-learners were identified during their interaction with the M-learning system. M-learners feature data contained information about M-learners participation in an online discussion group, type of learning contents accessed, average study time, online problems posted, online problems solved, quiz attempts, repetition rate, and module performance. The online M-learning course consisted of three JAVA and three Python programming modules. The aforementioned features are independent whereas the final performance is a dependent feature that deep ANN would try to predict. The aim was to predict M-learners' attainments and identifying important features that affect the learning performance of M-learners. M-learners were modeled using a five-level classification scheme ranging from A (excellent) to F (insufficient).

The rest of this article is organized as follows. In Section 2, we review the applications of ML/DL algorithms in mobile and web learning environments. Section 3 discusses the dataset, its features and how it was acquired from the M-learners. Section 4 explains three basic elements of the M-learning model: 1) the M-learner model, 2) the M-learner domain model, and 3) the M-learner adaptation model. Understanding these elements is important in understanding the working of the M-learning system. Section 5 presents the proposed M-learning system architecture which consists of gathering M-learning features, features pre-processing, features weight-tuning process, M-learning model generation, and M-learning model deployment. Section 6 briefly describes baseline multi-class classification models and their prediction accuracy when compared to deep ANN. Section 7 presents deep ANN model evaluation using accuracy, precision, recall, and F1 metrics. Section 8 discusses the early engagement experiment and M-learning model evaluation using the End-User Computing Satisfaction (EUCS) instrument. Section 9 summarizes this article and points to future directions.

II. RELATED WORK

M-learning systems emerged under the inspiration of studies in the area of Intelligent Tutoring System (ITS), E-learning, adaptive learning and Computer-Aided Learning (CAD) [14]–[16]. M-learning system architecture is considered as an extension of E-learning system architecture, although both architectures have differences. Unlike in E-learning systems, the learning in M-learning systems occurs in different contexts. Context discovery, background knowledge, learner profiling, learner tracking, learning preferences, content discovery and management, and semantically indexing important features are important steps during the development of adaptive M-learning systems. In contrast to E-learning systems, M-learning does not occur in predefined space and time but befall whenever run time problem is created and users need to get information about it [17]. M-learning allows learners to address current problems, works independently of social, temporal, spatial constraints and keep them engaged in continuous professional development.

The generic ML/DL approaches used in educational settings target prediction of learners' dropouts [18], [19], performance prediction [20]), predicting learners' engagement [21], [22], and failures prediction [23], [24], etc. Marbouti *et al.*, used linear regression (LR) to assess learners at-risk of failure [11]. Using attendance, exams, assessment features, the on-risk learners were predicted in different weeks of their first year. Moreover, Marbouti and Diefes-Dux used different ML algorithms including artificial neural networks, support vector machine, decision tree, and naïve Bayes for predicting risky learners and compared their results with LR as baseline algorithm [25]. In educational settings, the use of DL models is still in its infancy stage with a limited number of studies. Fei and Yeung evaluated several DL models for prediction of learners' dropout [26]. They interpreted features generated from learners' interaction with online learning systems as time-series problems, processed learners' features weekwise, to analyze their study behavior and predict at-risk learners. Using LR and SVM as baseline models, they compared the results of Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and hidden Markov model on the feature set encompassing data about lectures viewed and downloaded, assignment submitted, attempted quizzes, and interaction on online forum platform.

In literature, there has been substantial debate on online learning environments (M-learning, Virtual Learning Environment (VLE)) features that impact the learning performance of learners [27], [28]. Various studies in the past have been carried out that identified the key reasons contributing to the low performance of online learners [18], [29]. Jagger and Xu in their research study revealed that student-instructor communication is the key factor that strongly influences the learning performance of online learners [30]. Similarly, Shahiri and Husain conducted a comprehensive literature review to determine the key features that contribute significantly to classroom performance prediction [31]. J. Naren argued that assignments, quizzes, background knowledge prior to final examination are the key features in predicting the final performance of learners [32]. Another perception incorporates learners' past performance

in quizzes/assignments and demographics as important contributors in assessing the final performance. A study carried out by Daud *et al.*, employed family attributes such as family income, family expenditure, learners' characteristics, and learner's study orientation to assess their effect on learners' performance [33]. They concluded that low income, extensive family expenditure, job during study and health expenses are the key features impacting the overall family environment and ultimately affecting learning performance. Social influence, family education, learner's inherent features were also considered as significant factors in the final performance prediction.

According to Kahraman *et al.*, the design of adaptive learning systems require three phases: 1) organizing learning contents i.e. establishing a relationship between target learning content and prerequisite learning content, 2) identifying learners needs, requirements, and features, 3) defining the connection between learners needs and learning content [34]. The key to the successful adaptive system is to identify learners' features, their weights and establishing weight difference metrics amid learners' features. In contemporary classroom settings, there is a fixed and agreed curriculum with a single instructor and organized learning content whereas, in M-learning, the learning environment consists of temporary learning contexts. The fundamental challenge in M-learning is to identify the exact requirements of learners in temporary contexts and assist them accordingly thus making learning easy, adaptive, and meaningful. Nordin *et al.*, presented a theoretical mobile learning framework with the aim to assist M-learners in lifelong learning [35]. Key design factors of their mobile lifelong learning framework included mobile environment issues, learning theories, mobile learning context, learning objectives and learning experience. According to V. P. Dennen *et al.*, both behaviorism and constructivism learning theories can be used in designing instructional materials for M-learning [36]. They identified user mobile environment issues which include collection of M-learners profile data, inspecting learners' mobility, considering mobile interface design issues and learning context. In general, mobile devices are considered as supporting tools when used in the acquisition of knowledge in a different context. Because of unrestricted time and space constraints, mobile devices can also be used in different learning scenarios in pre/post activity mode. The success of M-learning depends on better usability offering professional Graphical User Interface (GUI) that presents an appealing user experience, attractive interaction along with clear goals and objectives.

Manwaring *et al.* used a cross-lagged modeling technique to understand learners' engagement in higher education blended classrooms [37]. The study found that learners' course interest, orientation, course design, and learners' perception features greatly influence learners' performance and engagement in the course. Mutahi *et al.* used ML and statistical techniques to determine the relationship between learners' engagement and learners' final performance score [38]. They found that learners' having high levels of engagement in reading learning content, taking quizzes, submitting assignments earned higher grades in final examinations. Aguiar *et al.* incorporated ML algorithms to investigate the factors that greatly influence learners' engagements and performance in classroom settings [39]. Their results showed that ML algorithms are very good in recognizing learners' facial expressions, eye gazes, gestures, and head poses and subsequently categorizing learners' into different engagement categories. Atherton *et al.* found that learners who accessed course content more often achieved better scores than learners who accessed less course content [40]. Hamid *et al.* in their study employed Support Vector Machine (SVM) and K-nearest Neighbor (KNN) algorithms to classify learners into different performance/engagement categories and the results concluded that SVM and K-NN are appropriate ML algorithms for predicting learners performance and engagement [6].

Baker presented a user model for the online adaption process in which users' preferences and background knowledge were the key components [41]. Adaptive navigation paths were established using user preferences and tailored contents were delivered using background knowledge. With an increase in a user performance, complex contents and more challenging tasks were presented to the user so that the user could control pace over the learning process. Bezold developed a task model that considered users navigations and interactions in online systems as a series of events [42]. A 'Probabilistic Deterministic Finite-State Automata PDFA' was used to label user behavior in online systems. For predicting and estimating the user's next activity 'first-order Markov chains' were used. The first-order Markov chains converted user interaction history into vector set and used them as an input parameter for predicting user next activities. The problem with task-based user modeling is that there is no settled standard procedure for gauging the methods used [43].

Guo *et al.* used an unsupervised sparse auto-encoder algorithm to develop a classification model from learners' unlabeled data [44]. The classification model was trained and tested on a relatively large dataset aimed at pre-train hidden layers. The classification was efficacious in an academic setting for learners' pre-warning mechanism. The main disadvantage of sparse auto-encoder is their failure to work with time-series data and have a low network architecture performance [45], [46].

Bouneffouf used a Markov decision process, a type of reinforcement learning technique to create a ubiquitous recommender system established on the user's changing context [47]. The recommender system delivers appropriate suggestions and recommendations to users based on their diverse context. A new user is recognized by a recommendation system based on his/her social group information and then gradually recommends new suggestions and actions according to the user's interest. The recommendation system links new actions according to the observed context of the user. Associations depend on the user's behavior and feedback to the recommendation system. The researcher was successful

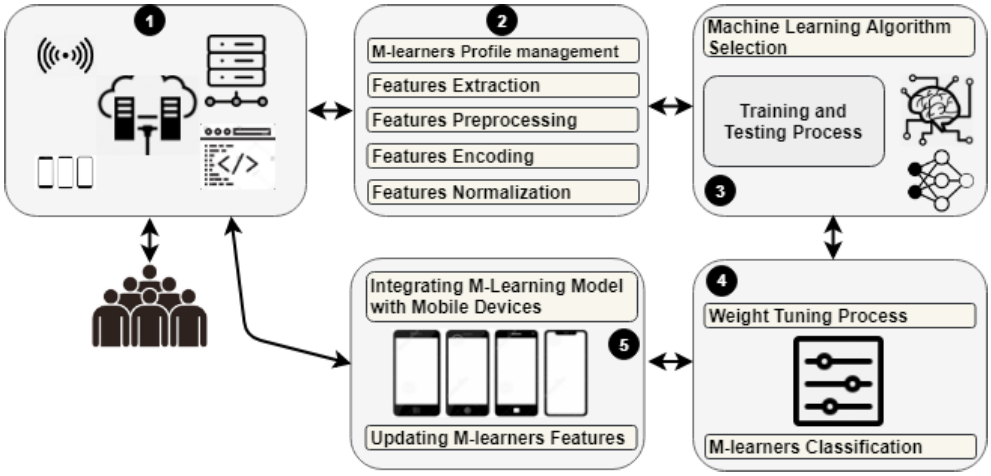


FIGURE 1. Proposed M-learning System Architecture.

in solving users cold-start problem that commonly occurs when new users have little experience with the existing system and they hesitate to perform basic interactions.

Sun *et al.* in their pilot study designed a mobile service-oriented system based on educational data mining (EDM) techniques, which targets organizing learning contents in the virtual learning environment (VLE) to support collaborative and microlearning in a massive open online course (MOOC) [48]. To make learning easy and self-paced, learning content was divided into small chunks that were supposed to be learned by students in short time duration. Based on learners' preferences, course chunks were sequenced into series of the identified paths, therefore, to enable learners to make the best use of fragmented pieces of time, to effectivity implicate in MOOC learning. Without a doubt, mobile learning is becoming more and more ubiquitous and a major means of learning. As a result, MOOC providers frequently release and update their mobile apps on major mobile operating systems (i.e. Android, iOS, Windows phones, etc) to catch mobile learning trends and to make learning easy and convenient for M-learners.

Arguably, the popularity of M-learning is compelling MOOC designers to allow M-learners to take MOOC courses on mobile devices [49], [50]. Standard models of M-learning look very much like traditional classroom learning where learners are taken out from normal living environments to spend five to six hours in learning stuff which they may or may not encounter in their daily lives [51]. Recently, standard M-learning models are swept out the door by new learning methods where not only M-learning takes place inside a normal work environment but smack right in the middle of it. In a working environment where mobile devices are considered an integral part of people, any type of learning activity is carried out in very short bursts of the period.

In previous research studies, many techniques and methodologies have been established to model the behavior of online learners, however, most of them had not been applied in real-world situations [52], [53]. The main reason for this problem turns out to be compelling learners to follow application domain constraints and not considering their needs, learning features and preferences. Most of the time learners are dependent on complex practices, theoretical models, system complexity and low-level details. According to literature, each learner's feature is equally important in defining his/her exact learning behavior [54], [55]. In other words, existing learner modeling methods ascribe equal weights to each feature in the learner modeling process. Not considering features weights and their association is the main reason for misclassification in the learner modeling process. The modern DL algorithms have enabled the development of a comprehensive learner model that can identify and represent a broader range of learner features which were not possible previously. DL algorithms such as deep Artificial Neural Networks (deep ANNs) with several hidden layers are capable of determining significant features along with their weights i.e. importance in classifying learners in different categories. Assigning a weight to each feature is called the weight-tuning process. The weight-tuning process improves learner modeling prediction, classification, and estimation results. M-learning system that can properly identify M-learners' needs and features will enable them to easily customize learning resources at a micro-level to meet their demands in real-time.

III. DATASET

Unlike online web-based learning systems and static classroom settings, the M-learning system faces more challenges in collecting features dataset. There is a lot of distraction, ambient noise, and instability for M-learners while they use mobile devices. M-learning occurs without temporal and spatial constraints. Therefore, it is important to know exactly what features influence M-learners more and how these features can be used for making the M-learning process easy and adaptive. M-learning system shown in figure 1 collects features data such as learning content accessed, learning location, study time duration, navigation paths, and learners' responses, etc. 374 M-learners participated in using our

TABLE 1. Dataset Features for M-Learning Model.

Features	Description
AST	Average Study Time, (numeric: 1 $\Longrightarrow$ time between 1 and 10 mins, 2 $\Longrightarrow$ time between 11 and 20 mins, 3 $\Longrightarrow$ time between 21 to 30 mins, 4 $\Longrightarrow$ time between 31 to 40 mins, 5 $\Longrightarrow$ time between 41 to 50 mins and 6 $\Longrightarrow$ time > 50 mins)
NTRA	Number of times Text Resource Accessed, (numeric: 0 $\Longrightarrow$ text resource accessed 0 times, 1 $\Longrightarrow$ text resource accessed 1 times, 2 $\Longrightarrow$ text resource accessed 2 times, etc.)
NTVA	Number of times Video Resource Accessed, (numeric: 0 $\Longrightarrow$ video resource accessed 0 times, 1 $\Longrightarrow$ video resource accessed 1 times, 2 $\Longrightarrow$ video resource accessed 2 times, etc.)
MP1	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)
MP2	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)
MP3	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)
APV	Number of times Academic Places Visited, (numeric: 0 $\Longrightarrow$ 0 times academic places visited, 1 $\Longrightarrow$ 1 times academic places visited, 2 $\Longrightarrow$ 2 times academic places visited, etc.)
SPV	Number of times Social Places Visited, (numeric: 0 $\Longrightarrow$ 0 times social places visited, 1 $\Longrightarrow$ 1 times social places visited, 2 $\Longrightarrow$ 2 times social places visited, etc.)
ODGP	Online Discussion Group Participation, (numeric: 0 $\Longrightarrow$ 0 times participated, 1 $\Longrightarrow$ 1 times participated, 2 $\Longrightarrow$ 2 times participated, etc.)
NPP	No of times problem posted, (numeric: 0 $\Longrightarrow$ 0 times problem posted, 1 $\Longrightarrow$ 1 times problem posted, 2 $\Longrightarrow$ 2 times problem posted, etc.)
NPS	No of times problem solved, (numeric: 0 $\Longrightarrow$ 0 times problem solved, 1 $\Longrightarrow$ 1 times problem solved, 2 $\Longrightarrow$ 2 times problem solved, etc.)
NTAQ	No of times attempted quiz, (numeric: 0 $\Longrightarrow$ 0 times attempted quiz, 1 $\Longrightarrow$ 1 times attempted quiz, 2 $\Longrightarrow$ 2 times attempted quiz, etc.)
TRR	Topic Repetition Rate, (numeric: 0 $\Longrightarrow$ 0 times topic repeated, 1 $\Longrightarrow$ 1 times topic repeated, 2 $\Longrightarrow$ 2 times topic repeated, etc.)
FG	Final Grades derived from the final performance score, dependent feature, categorical: (A,B,C,D,F)

proposed M-learning system to enhance their programming skills. A programming course each for JAVA and Python language was presented to M-learners on their Android-based mobile devices which they had to complete in 2 months. Each of these courses was further divided into three modules and after completion of each module, a quiz was conducted. After completion of each course, a final quiz was conducted to determine the final grades of M-learners.

Table 1 contains the features of our dataset along with their corresponding datatype domain values. The features are divided into three categories namely 'behavioral features', 'context features' and 'final grade'. The behavioral features are concerned with mobile learners' interaction during a study process such as participation in an online discussion group, posting problems, solving problems posted, number of times quiz was attempted, and topic repetition rate. The context features contains the learning context information of M-learners which includes features such as learning location, types of learning content accessed, average study time in daily routine, background knowledge, and modules performance, etc. The final grade is derived feature acquired from the final performance score. The final grade is a categorical feature representing the grades (A, B, C, D, F) of M-learners. The 13 behavioral and context features are independent features that are given as input to the Deep ANN model to predict the dependent final grades. The task of prediction consists of obtaining an M-learning model that relates the values of independent features with the values of dependent feature i.e. final grade. The actual values of independent features and their weights describe the mapping between the independent predictor features and the dependent target feature.

For the purpose of comparison, , classification and effectiveness of learning performance, five baseline multi-class classification ML algorithms are used which include Support Vector Machines (SVM), Random Forest (RF), K-Nearest Neighbors (K-NN), Multi-class Logistic Regression i.e. softmax regression, and Decision Trees (DT) along with Deep Artificial Neural Network (DANN).

IV. ELEMENTS OF ADAPTIVE M-LEARNING MODEL

The three major elements of our proposed adaptive M-learning model includes M-learner model, domain model, and adaptation model. An understanding of these elements is essential in knowing how the procedure of adaptiveness is carried out in the M-learning process.

A. M-LEARNER MODEL

M-learner model is the main source of personalization and adaptation in the M-learning process [56]. The features of M-learner define her/his needs in the M-learning process. The features of M-learners define a strong association between the M-learner model and the domain model in the M-learning environment. M-learner model stores feature such as background knowledge, performance states, and preferences, etc. that are used by the adaptation model to predict M-learners' knowledge about target learning object. The domain-dependent features of learners corresponding to target learning objects in the M-learning environment are represented by the set such as $< \mathrm{MLO}_1, \mathrm{MLO}_2, \mathrm{MLO}_3, \mathrm{MLO}_n >$ where each element $< \mathrm{MLO}_c >$ denotes M-learner context features such as average study time, type of learning content accessed, performance and places visited. The feature set also encompasses behavioral features such as online discussion group participation, posting problems, solving problems, topic repetition rate, and several quiz attempts corresponding to target learning object O_c.

It is obvious that each feature in the set $<$ MLOc$>$ will have a different effect on the knowledge and performance level of M-learner. Considering this fact, the current challenge is to explore the weight/importance of each feature. The aim of weight assignment is to find the real-values of each feature in the set $< \mathrm{MLO}_c >$ and model them on the learning behavior of M-learners. The weighted feature set represents the weight of each feature in the M-learner model composition.

B. M-LEARNER DOMAIN MODEL

The M-learner domain model comprises the learning objects in the application domain. In adaptive learning settings, the domain model represents learning objects that are in the interest of M-learners [57]. The domain model is designed to reflect the learner's goals, topics, and objectives. At a generic level, the goals and objectives of learning objects are defined independently of any domain whereas, at a detailed level, the goals, topics, and objectives are defined at a granular level. Because of the domain model, the ordered relationship among different learning objects can easily be defined. The instruction requirements for different learning objects in the domain model are also defined and stored in the M-learning system. The basic relationships in the domain model are the prerequisite connections among different learning objects. The prerequisite connections define the instruction requirements for different learning objects, which are to be fulfilled by the M-learners.

According to the domain model, the set of learning objects is represented by the set $\boldsymbol{O}$, $<$ O1, O2, O3…Om $>$. 'm' represents the total number of learning objects in the domain model. The learning objects in the set $\boldsymbol{O}$ can be represented in textual, video, audio, and animation form. The individual features of a learning object O_c can be represented by the set FOc $<$ FOc1, FOc2, FOc3…FOcn$>$. 'n' denotes the total number of features of learning object Oc. According to the generic domain model, the features of learning objects should be defined accurately to represent the environment where the learning occurs.

Some of the features of learning object O_c are difficulty level, learning duration, questions, and repetition number, etc. The instructor can state the real-values of learning object O_c features according to a measure of belief of the learner understanding about the difficulty level of individual learning objects.

C. M-LEARNER ADAPTATION MODEL

The purpose of the adaptation model is to deliver learning objects and activities to M-learners according to their learning features defined in the M-learner model [58], [59]. In our proposed model, the adaptation model consists of a Deep ANN algorithm that takes M-learner's features as input, processes them and based on their values, classify M-learners into different performance categories. The adaptation model generates adapted learning objects, objectives, and goals that are according to the learning behavior of M-learner. The customary e-learning and M-learning model uses a hired-wired implementation which follows the one-size-fits-all approach. As a result, the hard-wired adaptation model cannot differentiate among varying learners in providing them with more accurate and appropriate educational content. Furthermore, hard-wired adaptation models limit their potential to be scalable and applied to new types of learners. In stark contrast, our proposed M-learner adaptation model, which is based on the M-learning model, adapts to learning content in real-time according to individual learner's features and their corresponding weights.

The goal of the adaptation model is to assist learners in finding tailored learning objects from a large pool of learning content (text, video, audio, etc.). For example, the adaptation model can adaptatively select, sort, annotate, or partly hide the target learning objects to make it easier for the learner to choose where to go next. The adaptation model delivers learning objects to the learner in such a way where a learner can find an ''optimal path'' through the learning process. Furthermore, the adaptation model tries to be more cooperative and less directive as opposed to models used in the traditional learning systems: It leaves learners in a state from where they can choose which next knowledge item to learn or which problem to solve. In an M-learning environment where there is a lot of distraction, adaptive support becomes both natural and efficient. In the M-learning context, where there is no human teacher, tutor, or even peer nearby, the adaptation model has to provide a one-stop solution for all the learner's needs. Together with adaptive learning objects and adaptive information filtering processes, the adaptation model should be more attractive than interactive due to its natural fit to small screen size, low memory, and processing capabilities.

V. PROPOSED M-LEARNING SYSTEM ARCHITECTURE

Figure 1 shows the architecture of the proposed M-learning system. It is based on the M-learning model. The modeling process of M-learners is created and updated in five steps. The first step collects and stores M-learners' features data on the online Google Firebase cloud. M-learning system tracks and collects M-learners' features such as learners' participation in problem-solving, learning activities, navigation paths, performance scores, study time duration, and topic repetition rate about target learning objects, etc. Initially, the online data represent a generic profile of the M-learners as they are not processed, classified and weighted by the Deep ANN model. In the second step, the stored data is preprocessed, encoded, converted and normalized to useful data that becomes suitable to be further accepted and processed by the Deep ANN model. In the third step of the proposed M-learning system workflow, M-learners are classified by the deep ANN model depending on real-values of features about the target learning object. In the fourth step, the features of M-learners are weighted by the weight-tuning process. After that optimum weights are assigned to each feature, the M-learners are further classified based on weighted values of their features. In the fifth step, the M-learning model developed for each M-learner is deployed on their mobile devices for adaptive assistance and recommendations. Each M-learner has a particular M-learning model that represents his/her knowledge state, learning behavior and M-learning interests.

A. EXPERIMENT RESULTS OF DEEP ARTIFICIAL NEURAL NETWORK (DEEP ANN)

Deep Artificial Neural Network (Deep ANN) is a form of DL/deep neural network (DNN) algorithm that we have selected for M-learners' classification and performance prediction tasks [60]. Deep ANN relays on proper learners' datasets for its processing and prediction result generation. In our study, the dataset includes the features records of learners' study behavior stored on the online Google Firebase cloud.

Figure 2 shows deep ANN pictorially along with its input layer, hidden layers, and an output layer. The deep ANN has 13 input neurons (for 13 input features instances), 2 hidden layers having 6 neurons each and an output layer having 5 neurons. The 5 output neurons in output layers contain learner's performance grades i.e. A, B, C, D, F. During deep ANN implementation process we used Python Sequential class to map the 13 input features to input layer neurons and Python Dense class was used for randomly initializing weights to deep ANN synapses (edges). The output layer neurons yield learners' performance represented by five-level grades.

1) DEEP ANN LEARNING PROCESS

The deep ANN learning process i.e. M-learning model development process comprises the following steps.

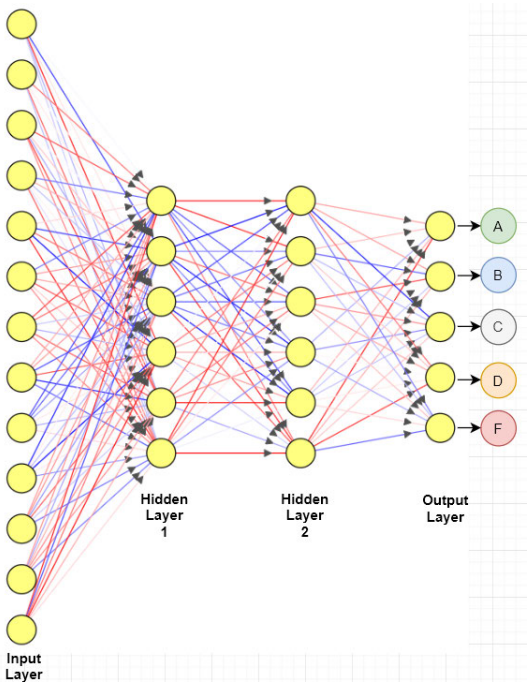


FIGURE 2. Deep ANN Processing M-learners' Features.

- Forward Propagation: for passing features instance through deep ANN.
- ReLU: using rectifier activation function for neuron activation in deep ANN hidden layers.
- Softmax function: to apportion learners' performance into final grades at the output layer.
- Back-propagation: feeding output error back to the ANN to mitigate the output-generated error.

2) FORWARD PROPAGATION

In the forward propagation technique, the M-learners' data flows from input through hidden layers towards the output layer. For four-layered deep ANN (1 input layer, 2 hidden layers, and 1 output layer) the learned function would be:

$$f(x) = f1(f2(f3(x))) \tag{1}$$

where:

$f1(x)$ = learning process occurred at hidden layer 1
$f2(x)$ = learning process occurred at hidden layer 2
$f3(x)$ = learning process occurred at output layer

At each layer, the deep ANN learns different representation and weights of input features that gets more complex with later hidden layers. Initially, the features instances are of the form $n * 13$, where n is the total number of feature instances and 13 are M-learners' features. To speed up the input process and to feed multiple inputs features records at one time to the deep ANN input layer we used matrix multiplication techniques. For performing matrix multiplication, first, we defined two matrices namely X and W_1. The input features are represented by matrix X having an $N * M$ dimension. N is the number of records in features dataset

whereas M represents a total number of M-learners' features, which in our case are 13. At the input layer, the weights of synapses are represented by the matrix W_1 matrix having $13 * 6$ dimensions. 13 denote the input neurons whereas 6 are the values of synapses weights attached to each neuron (6 synapses per neuron) at hidden layer 1. At the start of deep ANN operation, the Python Dense class randomly initialized the values of the weights on synapses. Initially, the values of the weights chosen by Python Dense class are close to zero. Mathematically, the matrix X and W_1 are represented as:

$$X = \begin{bmatrix} X1_1, & X2_1, X3_1, X4_1, \ldots X13_1 \\ X1_2, & X2_2, X3_2, X4_2, \ldots X13_2 \\ X1_3, & X2_3, X3_3, X4_3, \ldots X13_3 \\ \ldots\ldots\ldots & \\ X1_n, & X2_n, X3_n, X4_n, \ldots X13_n \end{bmatrix}$$

$$W_1 = \begin{bmatrix} W1_1, & W1_2, & W1_3, & W1_4, \ldots & W1_6 \\ W2_1, & W2_2, & W2_3, & W2_4, \ldots & W2_6 \\ W3_1, & W3_2, & W3_3, & W3_4, \ldots & W3_6 \\ \ldots\ldots\ldots & & & & \\ W13_1, & W13_2, & W13_3, & W13_4, \ldots & W13_6 \end{bmatrix}$$

Multiplication of matrix X with W_1 produces matrix *c2* having dimension $n * 6$ as shown in equation 2. This matrix multiplication technique would enable multiple features instances to pass through deep ANN at the same time.

$$c_2 = XW_1 \tag{2}$$

During the deep ANN forward propagation process, the neurons at hidden layer 1 perform two operations. First, the input matrix X representing the features instances is multiplied by the weights of the corresponding synapses and then multiplication result is added with other multiplication results at a neuron where the synapses are connected to it. Secondly, neurons at hidden layer one perform activation function. We used Rectifier activation Function (ReLU) to perform activation function on each entry in matrix c_2. Applying the ReLU activation function on c_2 deduce new equation as shown below.

$$e_2 = f(c_2) \tag{3}$$

The result of the ReLU activation function is stored in a new matrix e_2. The ReLU activation function is applied at each neuron in deep ANN hidden layer 1. To complete the forward propagation process, deep ANN propagates the values of e_2 all the way to the output layer. The result at the output layer is represented by $\hat{y}$ which is deep ANN predicted grades values for M-learners' performance. The operation at hidden layer 2 is the same as that of hidden layer 1. First, the result generated from hidden layer 1 neurons i.e. e_2 is multiplied by hidden layer 1 synapse weights using matrix multiplication technique. The dimension of matrix e_2 is $n * 6$ whereas the dimension of the W_2 matrix is $6 * 6.6 * 6$ represents synapses weights and neurons at hidden layer 2. The matrix e_2, when multiplied by matrix W_2, yields matrix c_3 having size $n * 6$ and can be denoted by the subsequent equation 4

$$c_3 = e_2 * W_2 \tag{4}$$

At the hidden layer 2, the ReLU activation function is applied on each entry of the c_3 matrix resulting in new matrix e_3. The dimension of e_3 is same as c_3 and can be written as:

$$e_3 = f(c_3) \tag{5}$$

The resultant matrix e_3 is further multiplied by layer 2 weights and the multiplication results are further added up at the output layer. The multiplication and addition process at the output layer produces matrix c_4. At the output layer, the Softmax function is applied to the entries of the c_4 matrix that generates the e_4 matrix which represented by equation 6.

$$e_4 = f(c_4) \tag{6}$$

Here e_4 is the predicted final grades representing learners' performance. The predicted final grades can also be represented $\hat{y}$. For improving the accuracy of the deep ANN, we must minimize the difference between predicted final grades $\hat{y}$ and actual final grades y. The difference between $\hat{y}$ and y can also be measured by cost function *C*.

3) TRAINING DEEP ANN-BASED M-LEARNING MODEL: BACK-PROPAGATION

The goal of the back-propagation technique is to optimize synapses weights so as to minimize the difference between predicted result $\hat{y}$ and actual result y. Cost function *C* tells us how wrong the predicted result was when compared to the actual result. The cost function can be expressed by the following equation 7.

$$C = \Sigma\, 1/2(y - \hat{y})^2 \tag{7}$$

In the back-propagation technique, the weights on the deep ANN synapses get updated causing the predicted result to come closer to the actual result. There are only two possibilities for minimizing the value of cost function; 1) changing the values of input features, 2) changing weights of deep ANN synapses. We do not have control over changing the values of input features, therefore, the only choice left for us is to adjust synapses weights values. To lessen time and computation resources, we used a technique called Stochastic Gradient Descent (SGD) to find optimal values for synapses weights. Assigning optimal values for synapses weights ensures minimum error in the predicted results. SGD updates synapses weights after every single record propagation through deep ANN, therefore, they have much higher fluctuation and ability to find global minimum values for synapses weights. SGD works well on higher dimension data and training models where the weights of the synapses have to be updated after each training sample. The following steps were carried out in back-propagation technique:

- Initialized synapses weights with random values and calculated the error in the predicted result.

- Compared to the predicted result with an actual result and measured the generated error.
- Generated error back-propagated from the output layer to the input layer through hidden layers.
- Updated synapses weights according to how much they are responsible for the generated error.
- Repeated the steps from 1 to 4 and updated synapses weights after each observation.
- The whole training dataset is passed through deep ANN which is an epoch.
- Redo more epochs until deep ANN gets suitable synapses weights values that generate a minimum error in the predicted result.

During the initial phases of the deep ANN learning process, it may not find the proper association between independent features and dependent feature. Therefore, deep ANN has to be train with a back-propagation technique where if the predicted result is not closer to the actual result, the error is back-propagated into the entire deep ANN. The lower the value of cost function C is, the lower will be the difference between y and $\hat{y}$.

VI. EXPERIMENTAL RESULTS OF BASELINE MULTI-CLASS CLASSIFICATION ALGORITHMS

In this section, we will discuss and apply baseline multi-class classification algorithms to our problem dataset and later will compare their prediction accuracy results with the deep ANN results.

A. SUPPORT VECTOR MACHINES (SVM)

SVM can produce significant classification accuracy with less computation power [61]. In practice, the SVM multi-class classification tasks (k > 2) are disintegrated into a series of binary tasks where the normal SVM technique is directly applied. Two popular SVM ensemble schemes are one-versus-all and one-versus-one [62].

1) ONE-VERSUS-ALL STRATEGY

In the SVM one-versus-all (OVA) strategy, a single model is trained for one class. The samples of the class selected are labeled as positive samples whereas other class samples are labeled as negatives. The following pseudo-code demonstrates how we used SVM OVA in learners' grades classification.

1) Inputs: M, a model (SVM OVA algorithm for binary classifiers)
2) Samples: (N: M-learner grades)
3) Labels y where yi belong to 1, ... K is the label for Ni learner grade

Although SVM OVA is a popular strategy, its implementation suffers from several problems. Firstly, the accuracy of predicted value may differ between different binary classifiers. Secondly, if equal numbers of the class exist in a problem set, the OVA see unbalanced distributions because typically the negative classes it observes are much larger than the positive classes.

2) ONE-VERSUS-ONE STRATEGY

In our dataset, the M-learners' performance is categorized into five classes (A, B, C, D, F), and thus OVO will create n(n-1)/2 = 10 binary classifiers i.e. (A, B), (A, C), (A, D), (A, F), (B, C), (B, D), (B, F), (C, D), (C, F), (D, F). If a learner's performance is to be classified, the obtained performance grade is presented to each binary classifier of the ensemble to create an array of individual classification, e.g. (A, A, A, A, B, B, B, C, C, D). Finally, a win for one class is the number of votes for that class. The class that has most votes wins. In our scenario, A class has most votes, therefore, the learner performance is classified into A class.

B. DECISION TREES (DT)

DT is a very common, simple and powerful technique for multi-class classification [63]. The working of decision trees is based on IF/ELSE conditional statements where if the condition is true, a direction in tree construction is followed else if the condition is false, an opposite direction is followed.

C. RANDOM FOREST (RF) ENSEMBLE METHOD

RF is considered as the most popular, simple, and flexible multi-class classification algorithm [64]. RF develop a forest consisting of several decision trees. The robustness and accuracy of RF increase with increasing the number of decision trees. The RF creates an ensemble of decision trees learning models which increases the overall accuracy result. As the number of trees increases, randomness is increased in the model which enables RF to select the most important feature while deciding at the node.

We have used RF in the M-learning model to measure the weights or the relative importance of each feature on M-learners' performance grades prediction. By looking at features' weights and importance, the M-learning model decides which learning path to recommend to M-learners thus making their learning interesting and adaptive. The M-learning model may also drop those features that do not contribute enough to the prediction process.

D. K-NEAREST NEIGHBORS (KNN)

KNN is a supervised ML algorithm used commonly both for classification and regression problems prediction [65]. When used for classification problems, KNN learning is based on "how similar" is object features to neighbor objects features. Initially, KNN chooses the number K of neighbors. Upon receiving the unclassified data, the KNN algorithm measures the distance (Manhattan, Euclidean, Minkowski, or Weighted) from the new data point to all the other data that has already been classified. Because KNN is based on features similarity, the KNN model classifies a new M-learner's performance grade based on how much her/his learning features are like already classified M-learners.

E. MULTI-CLASS LOGISTIC REGRESSION I.E. SOFTMAX REGRESSION

Like SVM, multi-class logistic regression can also be used for multi-class classification using two approaches: one-vs-rest also known as one-vs-all and one-vs-one [66]. In this study, we have 5 output classes (A, B, C, D, and F) therefore, multi-class logistic regression will train 5 classifiers. For the classification task, the probability of each class is predicted and the class with maximum probability is selected. For example, we have five model classifiers namely classifier_A, classifier_B, classifier_C, classifier_D, classifier_F and the probability we get during prediction/training phase is classifier_A = 40%, classifier_B = 45%, classifier_C = 50%, classifier_D = 35% and classifier_F = 37%. As the probability of class C in classifier_C is the highest therefore we predicted class C and class C generated in the output result.

In one-vs-one approach, a total of n*(n-1)/2 classes are trained, so if we have 5 classes, we train $5*(5-1)/2 = 10$ classifiers. During the training process, binary pairs of classes are considered, and the model classifier is trained on a subset of data containing those pairs of classes. As a contrast to the one-vs-rest approach, where each classifier predicts probability, in a one-vs-one approach, each classifier predicts one class during the classification phase. The class has been predicted the most in the output class. 10 classifiers trained for 5 grades could be classifier_AB, classifier_AC, classifier_AD, classifier_AF, classifier_BC, classifier_BD, classifier_BF, classifier_CD, classifier_CF, and classifier_DF. During classification, let's say the output of each classifier is: classifier_AB assign A, classifier_AC assign A, classifier_AD assign D, classifier_AF assign A, classifier_BC assign C, classifier_BD assign B, classifier_BF assign B, classifier_CD assign C, classifier_CF assign F, and classifier_DF assign F. As class A is predicted the most, therefore class A is predicted.

F. BASELINE MULTI-CLASS CLASSIFICATION ALGORITHMS PREDICTIVE ACCURACY RESULTS

The parameters adjusted for the six ML models were RF (e.g. T = 500), deep ANN (E = 150 epochs using forward-propagation and back-propagation algorithm), KNN (manhattan_distance (l1), K = 3), SVM (kernel = RBF, C = 1.0, degree = 3, gamma = 0.0, random_state = none), MCLR (One-vs-all, Softmax, Optimizer = stochastic gradient descent (SGD)). All DM models were evaluated using the following four configurations:

Model 1: This model accepts all features as input except the final grades (the output to be predicted);

Model 2: This model is similar to Model 1 except module 3 performance;

Model 3: This model is similar to Model 2 except module 2 performance; and

Model 4: This model is similar to Model 3 except module 1 performance.

TABLE 2. Multi-Class Classification Models Prediction Accuracy Results of Final Grades in JAVA Course.

Models	JAVA					
	SVM	DT	RF	KNN	MCLR	Deep ANN
Model 1	81.89	83.44	84.65	80.56	80.25	85.96
Model 2	80.45	82.11	82.76	78.34	76.17	83.54
Model 3	76.61	77.48	78.34	75.45	72.82	80.23
Model 4	68.93	73.65	75.76	70.61	69.41	77.65

TABLE 3. Multi-Class Classification Models Prediction Accuracy Results of Final Grades in Python Course.

Models	PYTHON					
	SVM	DT	RF	KNN	MCLR	Deep ANN
Model 1	85.55	87.54	88.45	83.64	81.65	89.47
Model 2	82.29	84.78	86.87	81.46	78.54	87.87
Model 3	72.76	80.56	83.67	74.12	75.65	85.45
Model 4	64.61	73.92	76.49	67.34	71.68	80.34

To produce optimal predictive models, 10 runs of 10-cross validation (a total of 100 simulations) were applied to each configuration. Under the 10-cross validation scheme, a dataset is shuffled randomly and is split into 10 equal groups. At a time, each group is taken as a test group whereas the rest of the nine groups are fitted into the model (acts as training data). This way each group is assigned to testing set once whereas it is assigned to training set 9 times. In the end, the results of 10 rounds were averaged to estimate the predictive accuracy of each model. The prediction accuracy results are shown for each DM algorithm with four configurations in the JAVA and Python courses in table 2 and table 3. Looking at the results, we observed that the accuracy of ANN and RF models was the highest in both the courses whereas the MCLR model showed inferior accuracy. As expected, the Model 1 in both courses achieved the highest accuracy. The predictive accuracy of all the model decreases as we remove the module 3 performance score (Model 2) module 2 performance score (Model 3), and module 1 performance score (Model 4). These results revealed that the intermediate performance score plays an important role in increasing the final grades and are directly correlated to it.

Besides module scores, it is important to know how much other learning features affect the final grades i.e. what is the weight/importance of other features in increasing the final grades of m-learners. We used Random Forest (RF) ensemble model to determine the weight of each feature in predicting the final grades of M-learners. As compared to other ML/DM models, RF gives better accuracy, robustness, and control over under-fitting and over-fitting problems. Looking at feature weights and importance helps in understanding the strength/weaknesses of M-learners during their interaction with the M-learning system. Table 4 presents the relative importance of independent features in percentage in increasing the final grades of M-learners in the JAVA and Python course. The result analysis revealed that modules performance scores i.e. MP1, MP2, and MP3 overall have 38% (in JAVA course) and 43.4% (in Python course) impact on learning outcomes of M-learners which indicates that individually these features are the most important and relevant ones in increasing the final grades. Moreover, behavioral and

TABLE 4. Features Relative Importance in the JAVA and Python Course in Percentage.

Courses	Features Relative Importance in Percentage												
	MP1	MP2	MP3	NTAQ	NTRA	AST	NPP	APV	NVRA	NPS	ODGP	NPS	TRR
JAVA	12.50	14.02	11.45	7.35	7.25	6.99	6.58	6.28	6.03	5.97	5.60	5.12	4.80
Python	14.31	15.47	13.66	7.16	6.41	6.41	5.85	5.80	5.59	5.25	5.05	4.89	4.08

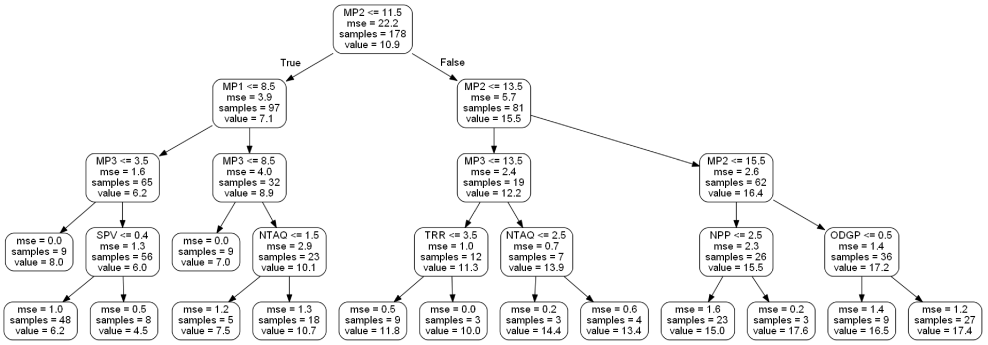


FIGURE 3. Short RF Tree for JAVA Course Important Features.

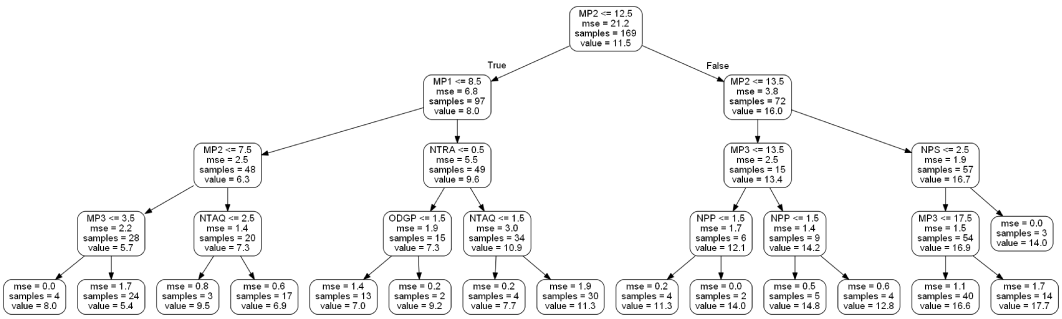


FIGURE 4. Short RF Tree for Python Course Important Features.

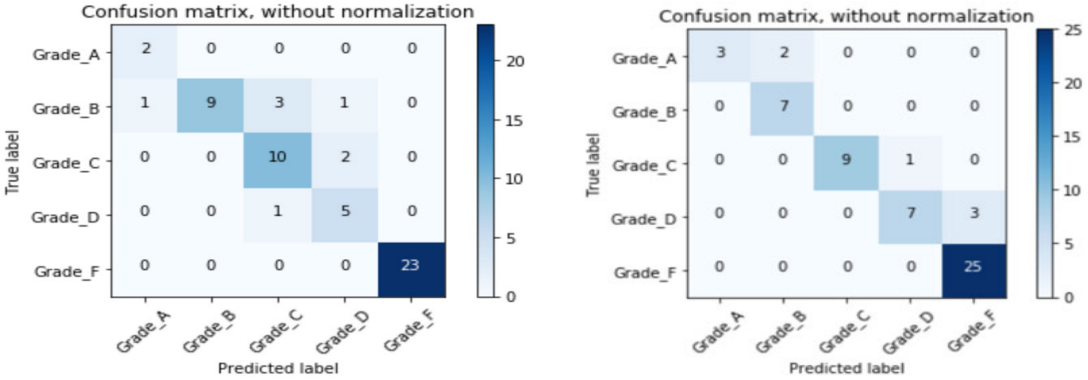
context features overall contribute 62.01% (in JAVA course) and 56.53% (in Python course) in increasing the learning behavior of M-learners. For instance, NTAQ (Number of times attempted quiz) feature has 7.35% and 7.16% impact on final grades in JAVA and Python course. Similarly, NTRA (Number of times text resource accessed) feature has 7.25% and 6.41% impact on final grades in JAVA and Python course. We also noticed that TRR (Topic repetition rate) feature has the lowest impact with 4.80% for the JAVA course and 4.08% for the Python course which indicates that M-learners give less importance to revising topic while they are using mobile devices.

Figure 3 and 4 plot the best decision trees for the RF algorithm. Again, the modules' performance MP1, MP2, and MP3 are the most important features appearing at the root of the trees whereas less important features such as TRR, SPV, NPP, ODGP, and NTAQ appears at the bottom of the trees. ML models that identify the most important features has three benefits. First, the ML model is easy to understand. Second, the overfitting of the model is reduced with the reduction of the variance of the model. Finally, the computational cost and time are reduced when we are training the model.

VII. DEEP ANN MODEL EVALUATION

As the deep ANN model was deployed on mobile devices, the next task that we performed was evaluating the performance of the deep ANN model via accuracy, precision, recall, and F1 score metrics [67]. Model evaluation delineates how well is the model doing? Is it a useful model? How the model performs on new data? How good the model predictions are? Moreover, these measures help models in providing help and adaptive content to the right person. For example, if the deep ANN model is helping a low average M-learner, the model must be sure that the M-learners it is helping has a low average performance. Further, the model also wants assist to all low average M-learners. The model is making sure that no low average M-learner is ignored/missed while guiding low average M-learners.

Figure 5a and 5b presents the confusion matrices generated for the JAVA and Python course datasets using numpy, sklearn, and seaborn ML libraries. The confusion matrices were generated after the deep ANN model training process. The deep ANN model was fitted on 85% training set in both cases whereas 15% data was allotted for the test set. The FG (final grades) predictions were compared to test data and each



(a) M-learning Model Confusion matrix Generated for JAVA Course

(b) M-learning Model Confusion matrix Generated for Python Course

FIGURE 5. M-learning Model Confusion Matrices.

prediction was identified as one of the 25 possible outcomes of the confusion matrix.

The three main metrics selected for deep ANN model evaluation are accuracy, precision, and recall. Accuracy is the percentage of correct grade predictions made by the model on the test data. Accuracy is calculated by dividing the number of correct grades predictions by the total number of grades predictions.

$$Accuracy = \frac{Correct\ Grades\ Predictions}{Total\ number\ of\ Grades\ predictions}$$

Calculating the accuracy of the deep ANN model for the JAVA and Python course confusion matrices gives the results as shown at the bottom of the page.

Putting values from confusion matrix A into the above equation yields

$$Accuracy\ (matrix\ A) = \frac{2+9+10+5+23}{57}$$

Accuracy (matrix A) = 85.96

Now calculating accuracy for matrix B

$$Accuracy\ (matrix\ B) = \frac{(3+7+9+7+25)}{57}$$

Accuracy (matrix B) = 89.47

For the JAVA and Python course datasets, we cannot solely rely on accuracy metrics as the data is not balanced which means that final grades (FG) are not distributed equally. Increasing only the model accuracy is not enough, it should also be useful, reliable, and valuable. If a small percentage of M-learners (let's say 1%) are getting F grade, we could build a model that almost always accurately predicts whether M-learners are getting passing grades or not, we would have designed a model that is 99% accurate but 0% reliable and useful. Therefore, we increase the performance of a model by introducing other metrics such as precision and recall which are discussed in the following section.

Precision is obtained by dividing true positive predicted upon true positive predicted and false-positive predicted.

$$Precision = \frac{True\ Positives}{True\ Positives\ +\ False\ Positives}$$

While calculating precision, first individual grade precision is obtained and then we calculate the average precision of a model.

$$\begin{aligned} Precision\ (matrix\ A) &= \frac{TP}{TP+FP} \\ P(A) &= .66\ P(B) = 1\ P(C) \\ &= .71P(D) = .62\ P(F) = 1 \\ &AveragePrecision(matrix\ A) \\ &= \frac{P(A)+P(B)+P(C)+P(D)+P(F)}{5} \\ &Average\ Precision\ (matrix\ A) \\ &= \frac{.66+1+.71+.62+1}{5} \\ &Average\ Precision\ (matrix\ A) \\ &= .80 \end{aligned}$$

Similarly, we calculate the precision of matrix B.

$$\begin{aligned} Precision\ (matrix\ B) &= \frac{TP}{TP+FP} \\ &Average\ Precision\ (matrix\ B) \\ &= \frac{P(A)+P(B)+P(C)+P(D)+P(F)}{5} \end{aligned}$$

$$Accuracy\ (matrix\ A) = \frac{True\ Positives + True\ Negatives}{True\ Positives + False\ Positives + False\ Negatives + True\ Negatives}$$

$$Accuracy\ (matrix\ A) = \frac{TP+TN}{TP+FP+FN+TN}$$

$$P(A) = 1P(B) = .77P(C)$$
$$= 1P(D) = .87P(F) = .89$$
$$Average\ Precision\ (matrix\ B) = \frac{1+.77+1+.87+.89}{5}$$
$$Average\ Precision\ (matrix\ B) = .90$$

and in percentage it is = **90%**

Calculating the recall of matrix A and matrix B yields the following results.

$$Recall\ (matrix\ A) = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

First, individual grade recall is determined for matrix A and then the average recall value is calculated for the model.

$$Recall\ (matrix\ A) = \frac{TP}{TP + FN}$$
$$R(A)=1,\ R(B)=.64,\ R(C)=.83,$$
$$R(D) = .83,\ R(F)=1$$
$$AverageRecall\ (matrix\ A) = \frac{R(A)+R(B)+R(C)+R(D)+R(F)}{5}$$
$$Average\ Recall\ (matrix\ A) = \frac{1+.64+.83+.83+1}{5}$$
$$Average\ Recall\ (matrix\ A) = .86$$

In percentage the average recall = **86%**

Next, the recall value for matrix B is calculated.

$$Recall\ (matrix\ B) = \frac{TP}{TP + FN}$$

First, individual grade recall for matrix B is determined and then the average recall value is calculated for the model.

$$R(A) = .6,\ R(B) = 1,\ R(C) = .9,$$
$$R(D) = .7,\ R(F) = 1$$
$$AverageRecall\ (matrix\ B) = \frac{R(A)+R(B)+R(C)+R(D)+R(F)}{5}$$
$$Average\ Recall\ (matrix\ B) = \frac{.6+1+.9+.7+1}{5}$$
$$Average\ Recall\ (matrix\ B) = .84$$

In percentage the average recall = **84%**

Recall metric ensures that we are not overlooking few M-learners who are getting low or high-performance grades. Suppose if only 1% of M-learners are getting F grade and 99% are getting A, B, C, D grades then the model would predict the grades of M-learners having A, B, C, and D with 99% accuracy. This means that the accuracy of the model is 99% and it is very likely that M-learners having F grades may be categorized in higher grades. Recall metric ensures that we are not overlooking those 1% M-learners having F grades. On the other hand, the precision metric ensures that we are not misclassifying too many M-learners as having F grade when in fact they don't. Thus it is very important to evaluate the ML model in terms of both precision and recall metrics. The last metric which we used to evaluate our ANN model was the F1 score. F1 score maintains a balance between precision and recall for the M-learning model. The equation for calculating the F1 score is:

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Calculating F1 score for matrix A and matrix B yields:

$$F1\ Score\ (matrix\ A) = 2 * \frac{.80 * .86}{.80 + .86}$$
$$F1\ Score\ (matrix\ A) = .82$$

In percentage, the F1 score for matrix A = **82%**.

Similarly, we calculate the F1 score for matrix B

$$F1\ Score\ (matrix\ B) = 2 * \frac{.90 * .84}{.90 + .84}$$
$$F1\ Score\ (matrix\ B) = .86$$

In percentage the F1 score for matrix B = **86%**

As the F1 score for matrix B is greater than the F1 score of matrix A, this means that the model built on the Python course dataset will give better results and will work well on unbalanced datasets.

VIII. EARLY ENGAGEMENT EXPERIMENT

After training and testing the deep ANN-based M-learning model, an early engagement experiment was performed on those M-learners who achieved grade D and F in the JAVA and Python course. The purpose of the early engagement experiment was to determine whether early engagement in the learning process improves learning performance or not. The total number of M-learners who obtained grades D and F in the JAVA course were D = 52 and F = 146 whereas the total number of M-learners who obtained grades D and F in Python course was D = 46 and F = 168. M-learners obtaining D and F grades in the JAVA course were divided equally into control (the control group for JAVA course M-learners, CJ = 99) and experimental (the experimental group for JAVA course M-learners, EJ = 99) groups. Similarly, M-learners obtaining D and F grades in Python course were divided equally into control (the control group for Python M-learners, CP = 107) and experimental (the experimental group for Python course M-learners, EP = 107) groups. The early engagement experiment lasted for one month where CJ and CP M-learners were independent of early engagement and received normal programming exercises and learning material. On the other hand, the EJ and EP M-learners were intervened during their learning process by providing them adaptive programming content, motivational/adaptive messages, and adaptive navigational paths. The M-learning model can help both new and old M-learners in providing them adaptive help and making

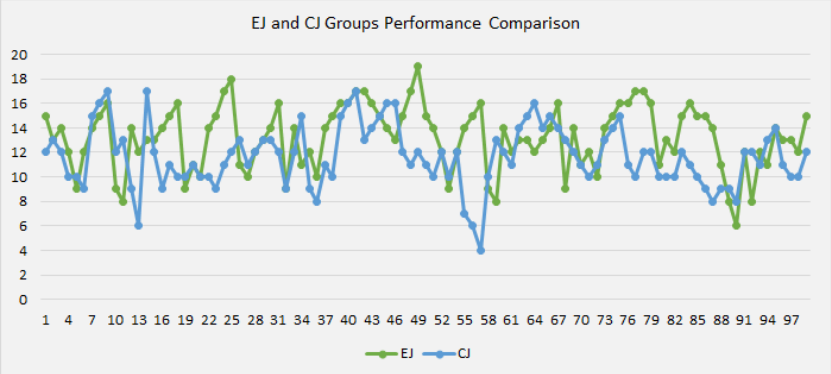


FIGURE 6. Performance Comparison of EJ and CJ Groups.

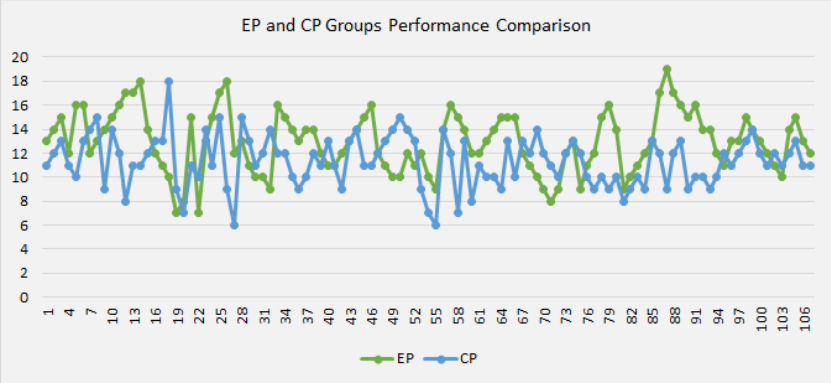


FIGURE 7. Perfomance Comparison of EP and CP Groups.

their learning self-paced. The M-learning model does not have information about new M-learners but it has learned from its experience/training about the features of different M-learners. Based on its experience, the M-learning model can guide new M-learners proactively and adaptively before they give their final examination thus motivating and guiding M-learners to increase their study performance. The EJ and EP group M-learners were engaged in their learning process through the following measures:

- Sending adaptive messages to EJ and EP M-learners according to their M-learning preferences. The purpose of sending adaptive messages to M-learners was to provide them adaptive learning material and support during their M-learning process. Some examples of adaptive messages are 1. ''Please revise the earlier topics if you want to study the new topic''. This adaptive message is provided to those M-learners who do not revise their study. 2. ''Please see chapter 5, 6 of Deital & Deital book to know more about classes and objects in JAVA''. This message is sent to those M-learners who like reading textbooks as oppose to watching educational videos during their study.
- Sending motivational messages to EJ and EP M-learners according to their M-learning performance. The aim of sending motivational messages was to increase M-learners' motivation towards learning. Some examples of motivational messages are 1. ''Continues poor performance will put you in a ceased/relegation state''. 2. ''Programming is learned by doing it. Try to practice programming exercise daily for at least 2 hours''. 3. ''Congratulations! You have improved your programming skills and now you are in the top 10 in your class''. 4. ''Please see the newly uploaded video by your instructor on the Google Groups regarding exception handling''.

In motivational messages, the factors of fear, hope, and suggestions were included in order to increase the M-learners inspiration towards learning [68].

A. EARLY ENGAGEMENT EXPERIMENT RESULTS

After one month, the performance results of the 4 groups were compared in pairs. The performance results of the CJ group was compared to the performance results of the EJ group. Similarly, the performance results of the CP group was compared to the performance result of the EP group. The results in figure 6 and 7 concluded that engaged M-learners (EJ, EP groups) overall showed a better performance than unengaged M-learners. These results revealed that early engagement of M-learners through motivational and adaptive

TABLE 5. End-User Computing Satisfaction (EUCS) Survey Result.

S.No	Dimension	Questions	Mean Value
1	Usefulness	The M-learning system motivated me to improve my programming skills.	4.65
2	Usefulness	After using the M-learning system, I can write computer programs with more confidence.	4.54
3	Engaging	While using the M-learning system, I was curious about learning more and new things.	4.23
4	Engaging	The M-learning system presented programming tasks in an engaging and interesting manner.	4.11
5	Ease of use	The use of M-learning system was very simple and easy	4.51
6	Ease of use	I can use the M-learning system without any expert help.	4.45
7	Timeliness	M-learning system provided learning material and help on time.	4.43
8	Timeliness	The programming exercises and quizzes were conducted on time.	4.55
9	Adaptiveness	M-learning system did not overwhelm me with unnecessary learning material.	4.34
10	Adaptiveness	The programming content provided was tailored and according to my performance state.	4.56
11	Attitude towards M-learning system	I will recommend others to use M-learning system for increasing their programming abilities.	4.14
12	Attitude towards M-learning system	I will use the M-learning system or similar type of systems in the future.	4.23

messages do motivate them in improving their learning performance. Overall it was noticed that EJ group performance was 7.78% higher than the CJ group in the JAVA course whereas in Python course the EP group outperformed the CP group by 8.64%.

B. ANALYZING M-LEARNERS CONTENTMENT THROUGH EUCS INSTRUMENT

End-User Computing Satisfaction (EUCS) is a well-known and frequently used instrument to measure the end-user contentment and experience of using a software system [69]. End-user contentment/experience specifically includes software application usefulness, user-engaging experience, software ease of use, timeliness, software adaptively, and user attitude towards using a software system. Several research studies have introduced modified and customized versions of the EUCS instrument but all versions focus on determining end-users satisfaction about software systems after they have used it [70], [71]. We used a modified version of the EUCS instrument to elicit M-learners' contentment after using the M-learning system supported by the M-learning model. Using the Google Form survey administration app, the EUCS survey was conducted with 206 EJ and EP group M-learners. Total 12 questions covering 6 dimensions of EUCS instrument namely usefulness, engaging, ease of use, timeliness, adaptiveness, and attitude towards using the M-learning system were administered on EJ and EP group M-learners. Five-point Likert-scale was used to measure M-learners' satisfaction toward using the M-learning system where 5 means ''strongly agree'' and 1 means ''strongly disagree''. Considering the assigned five-points on Likert-scale, the mean M-learners contentment was set to 4 (agree) or greater, which implies that overall the M-learners were satisfied with the M-learning system and M-learning system did increase their job performance. Table 5 presents 6 evaluation dimensions of the M-learning system, corresponding questions and mean score.

The response to questions 1 and 2 indicated that the M-learning system along with early engagement measures was successful in increasing the programming skills of M-learners ($m = 4.65$, $m = 4.54$). The response to questions 3 and 4 presented that during the early engagement experiment the M-learners were persuaded to take time to learn computer programming ($m = 4.23$, $m = 4.11$). The answer to questions 5 and 6 revealed that the M-learning system was user-friendly and easy to use during its interaction with M-learners ($m = 4.51$, $m = 4.45$). Similarly, the response to questions 7 and 8 indicated that the M-learning system considered M-learners preferred learning time and delivered help and study material accordingly ($m = 4.43$, $m = 4.55$). Likewise, the riposte to questions 9 and 10 showed that the M-learning system was successful in delivering adaptive and tailored learning content/guidance to M-learners according to their learning behavior and performance ($m = 4.34$, $m = 4.56$). Lastly, the response to questions 11 and 12 specified that the M-learners agreed to use the M-learning system or similar type of systems in the future to increase their programming skills ($m = 4.14$, $m = 4.23$).

IX. CONCLUSION AND FUTURE WORK

In this research study, we developed and proposed the M-learning model which when integrated with the

M-learning system provides an adaptive learning experience to the M-learners. For developing the M-learning model, we first trained it on 85% M-learners features data, generated while M-learners were taking the JAVA and Python course. When tested on 15% test-size data, the M-learning model classified M-learners into A, B, C, D, and F grades with 85.96% accuracy for the JAVA course and 89.47% for Python course. Moreover, the M-learning model achieved 80% precision, 86% recall, and 82% F1 score for the JAVA course whereas it achieved 90% precision, 84% recall and 86% F1 score for the Python course.

For determining the weights of M-learners' features, we used the Random Forest (RF) ensemble method. Results revealed that modules performance score i.e. MP1, MP2, and MP3 contributes significantly in predicting the final performance of M-learners. Moreover, behavioral and context features such as NTAQ, NTRA, AST, and NPP also plays a significant role in performance prediction.

When compared with 5 baseline multi-class classification models, we noticed that the deep ANN-based model outperformed others by predicting M-learners' grades with more accuracy. We noticed that the closest multi-class classification model with the deep ANN model in terms of prediction accuracy was RF.

This study also determines the effectiveness of the M-learning model in early engagement/intervention of M-learners. The early engagement process can help university administration and instructors in providing timely guidance, support, and counseling to the learners. Generally, traditional classroom settings and virtual learning environment (VLE) follows a one-size-fits-all approach where it is very difficult for the institute and instructors to know the exact needs and problems of the individual learners. On the other hand, mobile devices and M-learning features can help institute and instructors in knowing learners' performance state, preferences, needs, and problems. Moreover, M-learning features can help the institute in formulating helping committees for learners' timely support and provision thus increasing their overall productivity and maintaining their decorum.

These results demonstrate the effectiveness of our proposed M-learning system in predicting M-learners' performance and determining significant features with high impact on learning outcomes. Our predictive models are useful for institutions in formulation of a proactive analytics model, that supports their decision-making process. In future, we intend to incorporate additional deep learning algorithms such as Long Short-Term Memory (LSTM), Recurrent Neural Network (RNN), Self-Organizing Maps (SOMs), etc. in training and testing our M-learning model with the aim to increase the accuracy and bring more improvement in the effectiveness of M-learning model. In this research study, 374 M-learners participated in using the M-learning system and took the JAVA and Python course. The number of M-learners was kept low as the programming courses were delivered on their mobile devices. We intend to increase the number of M-learners by integrating the M-learning model with Learning Management System (LMS) in the future. We hope that increasing the number of M-learners and their corresponding features will help in improvement of accuracy and effectiveness of our M-learning model because the deep learning algorithms produce better results on larger dataset containing hundreds of features and dimensions.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
[2] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.
[3] V.-H. Trieu, "Getting value from business intelligence systems: A review and research agenda," *Decis. Support Syst.*, vol. 93, pp. 111–124, Jan. 2017.
[4] P. V. Balachandran, D. Xue, J. Theiler, J. Hogden, J. E. Gubernatis, and T. Lookman, "Importance of feature selection in machine learning and adaptive design for materials," in *Materials Discovery and Design*. Cham, Switzerland: Springer, 2018, pp. 59–79.
[5] H. M. Truong, "Integrating learning styles and adaptive e-learning system: Current developments, problems and opportunities," *Comput. Hum. Behav.*, vol. 55, pp. 1185–1193, Feb. 2016.
[6] S. S. A. Hamid, N. Admodisastro, N. Manshor, A. Kamaruddin, and A. A. A. Ghani, "Dyslexia adaptive learning model: Student engagement prediction using machine learning approach," in *Proc. Int. Conf. Soft Comput. Data Mining*. Johor, Malaysia: Springer, 2018, pp. 372–384.
[7] T. Vafeiadis, K. I. Diamantaras, G. Sarigiannidis, and K. C. Chatzisavvas, "A comparison of machine learning techniques for customer churn prediction," *Simul. Model. Pract. Theory*, vol. 55, pp. 1–9, Jun. 2015.
[8] S. Hassan, H. Waheed, N. R. Aljohani, M. Ali, S. Ventura, and F. Herrera, "Virtual learning environment to predict withdrawal by leveraging deep learning," *Int. J. Intell. Syst.*, vol. 34, no. 8, pp. 1935–1952, Aug. 2019.
[9] T. R. Patil and S. S. Sherekar, "Performance analysis of Naive Bayes and J48 classification algorithm for data classification," *Int. J. Comput. Sci. Appl.*, vol. 6, no. 2, pp. 256–261, 2013.
[10] P. Kaur, M. Singh, and G. S. Josan, "Classification and prediction based data mining algorithms to predict slow learners in education sector," *Procedia Comput. Sci.*, vol. 57, pp. 500–508, 2015.
[11] F. Marbouti, H. A. Diefes-Dux, and K. Madhavan, "Models for early prediction of at-risk students in a course using standards-based grading," *Comput. Edu.*, vol. 103, pp. 1–15, Dec. 2016.
[12] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, "Machine learning on big data: Opportunities and challenges," *Neurocomputing*, vol. 237, pp. 350–361, May 2017.
[13] C. Fan, F. Xiao, and Y. Zhao, "A short-term building cooling load prediction method using deep learning algorithms," *Appl. Energy*, vol. 195, pp. 222–233, Jun. 2017.
[14] I. Jugo, B. Kovacic, and V. Slavuj, "Increasing the adaptivity of an intelligent tutoring system with educational data mining: A system overview," *Int. J. Eng. Technol.*, vol. 11, no. 3, pp. 67–70, 2016.
[15] G. Riahi, "E-learning systems based on cloud computing: A review," *Procedia Comput. Sci.*, vol. 62, pp. 352–359, 2015.
[16] D. Chao, H. T. Chang, and P. P. Tong, "Computer-aided learning method and systems matching students with instructors," U.S. Patent 6 325 632, Dec. 4, 2001.
[17] Y.-M. Cheng, "Towards an understanding of the factors affecting m-learning acceptance: Roles of technological characteristics and compatibility," *Asia–Pacific Manage. Rev.*, vol. 20, no. 3, pp. 109–119, Sep. 2015.
[18] K. S. Hone and G. R. El Said, "Exploring the factors affecting MOOC retention: A survey study," *Comput. Edu.*, vol. 98, pp. 157–168, Jul. 2016.
[19] G. R. El Said, "Understanding how learners use massive open online courses and why they drop out: Thematic analysis of an interview study in a developing country," *J. Educ. Comput. Res.*, vol. 55, no. 5, pp. 724–752, Sep. 2017.
[20] A. Hernández-Blanco, B. Herrera-Flores, D. Tomás, and B. Navarro-Colorado, "A systematic review of deep learning approaches to educational data mining," *Complexity*, vol. 2019, pp. 1–22, May 2019.
[21] M. Wells, A. Wollenschlaeger, D. Lefevre, G. D. Magoulas, and A. Poulovassilis, "Analysing engagement in an online management programme and implications for course design," in *Proc. 6th Int. Conf.*, 2016, pp. 236–240.

[22] M. Hussain, W. Zhu, W. Zhang, and S. M. R. Abidi, "Student engagement predictions in an e-Learning system and their impact on student course assessment scores," *Comput. Intell. Neurosci.*, vol. 2018, pp. 1–21, Oct. 2018.

[23] E. B. Costa, B. Fonseca, M. A. Santana, F. F. de Araújo, and J. Rego, "Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses," *Comput. Hum. Behav.*, vol. 73, pp. 247–256, Aug. 2017.

[24] S. N. Liao, D. Zingaro, K. Thai, C. Alvarado, W. G. Griswold, and L. Porter, "A robust machine learning technique to predict low-performing students," *ACM Trans. Comput. Edu.*, vol. 19, no. 3, p. 18, 2019.

[25] M. F. Marbouti and H. A. Diefes-Dux, "Building course-specific regression-based models to identify at-risk students," *Age*, vol. 26, p. 1, Jun. 2015.

[26] M. Fei and D.-Y. Yeung, "Temporal models for predicting student dropout in massive open online courses," in *Proc. IEEE Int. Conf. Data Mining Workshop (ICDMW)*, Nov. 2015, pp. 256–263.

[27] M. Tan and P. Shao, "Prediction of student dropout in e-learning program through the use of machine learning method," *Int. J. Emerg. Technol. Learn.*, vol. 10, no. 1, pp. 11–17, 2015.

[28] W. Xing, X. Chen, J. Stein, and M. Marcinkowski, "Temporal predication of dropouts in MOOCs: Reaching the low hanging fruit through stacking generalization," *Comput. Hum. Behav.*, vol. 58, pp. 119–129, May 2016.

[29] H. Khalil and M. Ebner, "Moocs completion rates and possible methods to improve retention-a literature review," in *EdMedia+ Innovate Learning*. Jacksonville, FL, USA: Association for the Advancement of Computing in Education (AACE), 2014, pp. 1305–1313.

[30] S. S. Jaggars and D. Xu, "How do online course design features influence student performance?" *Comput. Edu.*, vol. 95, pp. 270–284, Apr. 2016.

[31] A. M. Shahiri, W. Husain, and N. A. Rashid, "A review on predicting student's performance using data mining techniques," *Procedia Comput. Sci.*, vol. 72, pp. 414–422, 2015.

[32] J. Naren, "Application of data mining in educational database for predicting behavioural patterns of the students," *Int. J. Eng. Technol.*, vol. 5, no. 3, pp. 4469–4472, 2014.

[33] A. Daud, N. R. Aljohani, R. A. Abbasi, M. D. Lytras, F. Abbas, and J. S. Alowibdi, "Predicting student performance using advanced learning analytics," in *Proc. 26th Int. Conf. World Wide Web Companion-WWW Companion*, 2017, pp. 415–421.

[34] H. T. Kahraman, S. Sagiroglu, and I. Colak, "The development of intuitive knowledge classifier and the modeling of domain dependent data," *Knowl.-Based Syst.*, vol. 37, pp. 283–295, Jan. 2013.

[35] N. Nordin, M. A. Embi, and M. M. Yunus, "Mobile learning framework for lifelong learning," *Procedia-Social Behav. Sci.*, vol. 7, pp. 130–138, Jan. 2010.

[36] V. P. Dennen, K. J. Burner, and M. L. Cates, "Information and communication technologies, and learning theories: Putting pedagogy into practice," in *Second Handbook of Information Technology in Primary and Secondary Education*. Cham, Switzerland: Springer, 2018, pp. 143–160.

[37] K. C. Manwaring, R. Larsen, C. R. Graham, C. R. Henrie, and L. R. Halverson, "Investigating student engagement in blended learning settings using experience sampling and structural equation modeling," *Internet Higher Edu.*, vol. 35, pp. 21–33, Oct. 2017.

[38] J. Mutahi, A. Kinai, N. Bore, A. Diriye, and K. Weldemariam, "Studying engagement and performance with learning technology in an African classroom," in *Proc. 7th Int. Learn. Analytics Knowl. Conf.* New York, NY, USA: ACM, Mar. 2017, pp. 148–152.

[39] E. Aguiar, G. A. Ambrose, N. V. Chawla, V. Goodrich, and J. Brockman, "Engagement vs performance: Using electronic portfolios to predict first semester engineering student persistence," *J. Learn. Analytics*, vol. 1, no. 3, pp. 7–33, 2014.

[40] M. Atherton, M. Shah, J. Vazquez, Z. Griffiths, B. Jackson, and C. Burgess, "Using learning analytics to assess student engagement and academic outcomes in open access enabling programmes," *Open Learn., J. Open, Distance e-Learn.*, vol. 32, no. 2, pp. 119–136, May 2017.

[41] R. S. Baker, "Stupid tutoring systems, intelligent humans," *Int. J. Artif. Intell. Edu.*, vol. 26, no. 2, pp. 600–614, Jun. 2016.

[42] M. Bezold, "Describing user interactions in adaptive interactive systems," in *Proc. Int. Conf. User Modeling, Adaptation, Personalization*. Trento, Italy: Springer, 2009, pp. 150–161.

[43] C. Abela, C. Staff, and S. Handschuh, "Task-based user modelling for knowledge work support," in *Proc. Int. Conf. User Modeling, Adaptation, Personalization*. Manoa, HI, USA: Springer, 2010, pp. 419–422.

[44] B. Guo, R. Zhang, G. Xu, C. Shi, and L. Yang, "Predicting students performance in educational data mining," in *Proc. Int. Symp. Educ. Technol. (ISET)*, Jul. 2015, pp. 125–128.

[45] Q. Meng, D. Catchpoole, D. Skillicom, and P. J. Kennedy, "Relational autoencoder for feature extraction," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 364–371.

[46] D. Charte, F. Charte, S. García, M. J. del Jesus, and F. Herrera, "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines," *Inf. Fusion*, vol. 44, pp. 78–96, Nov. 2018.

[47] D. Bouneffouf, "Applying machine learning techniques to improve user acceptance on ubiquitous environement," 2013, *arXiv:1301.4351*. [Online]. Available: http://arxiv.org/abs/1301.4351

[48] G. Sun, T. Cui, J. Yong, J. Shen, and S. Chen, "MLaaS: A cloud-based system for delivering adaptive micro learning in mobile MOOC learning," *IEEE Trans. Services Comput.*, vol. 11, no. 2, pp. 292–305, Mar. 2018.

[49] P. Pham and J. Wang, "AttentiveLearner^2: A multimodal approach for improving MOOC learning on mobile devices," in *Proc. Int. Conf. Artif. Intell. Educ.* Springer, 2017, pp. 561–564.

[50] B. Tabuenca, M. Kalz, and A. Löhr, "MoocCast: Evaluating mobile-screencast for online courses," *Universal Access Inf. Soc.*, vol. 17, no. 4, pp. 745–753, Nov. 2018.

[51] L. Ramírez-Donoso, J. S. Rojas-Riethmuller, M. Pérez-Sanagustín, A. Neyem, and C. Alario-Hoyos, "MyMOOCSpace: A cloud-based mobile system to support effective collaboration in higher education online courses," *Comput. Appl. Eng. Edu.*, vol. 25, no. 6, pp. 910–926, Nov. 2017.

[52] M. Aparicio, F. Bacao, and T. Oliveira, "An e-learning theoretical framework," *J. Educ. Technol. Soc.*, vol. 19, no. 1, pp. 292–307, 2016.

[53] P. Beach, "Self-directed online learning: A theoretical model for understanding elementary teachers' online learning experiences," *Teaching Teacher Edu.*, vol. 61, pp. 60–72, Jan. 2017.

[54] A. Razmjoo, P. Xanthopoulos, and Q. P. Zheng, "Online feature importance ranking based on sensitivity analysis," *Expert Syst. Appl.*, vol. 85, pp. 397–406, Nov. 2017.

[55] K. Konyushkova, R. Sznitman, and P. Fua, "Learning active learning from data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4225–4235.

[56] Y. Long and V. Aleven, "Enhancing learning outcomes through self-regulated learning support with an open learner model," *User Model. User-Adapted Interact.*, vol. 27, no. 1, pp. 55–88, Mar. 2017.

[57] R. A. Sottilare, A. C. Graesser, X. Hu, A. Olney, B. Nye, and A. M. Sinatra, *Design Recommendations for Intelligent Tutoring Systems: Domain Modeling*, vol. 4. Adelphi, MD, USA: US Army Research Laboratory, 2016.

[58] A. Battou, "Designing an adaptive learning system based on a balanced combination of agile learner design and learner centered approach," *Amer. Sci. Res. J. Eng. Technol., Sci.*, vol. 37, no. 1, pp. 178–186, Oct. 2017.

[59] M. Tmimi, M. Benslimane, M. Berrada, and K. Ouazzani, "Implemented and tested conception proposal of adaptation model for adaptive hypermedia," *Int. J. Emerg. Technol. Learn. (iJET)*, vol. 14, no. 02, p. 16, Jan. 2019.

[60] I. N. Da Silva, D. H. Spatti, R. A. Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves, *Artificial Neural Networks*. Cham, Switzerland: Springer, 2017.

[61] N. Guenther and M. Schonlau, "Support vector machines," *Stata J.*, vol. 16, no. 4, pp. 917–937, 2016.

[62] Y. Liu, J.-W. Bi, and Z.-P. Fan, "A method for multi-class sentiment classification based on an improved one-vs-one (OVO) strategy and the support vector machine (SVM) algorithm," *Inf. Sci.*, vols. 394–395, pp. 38–52, Jul. 2017.

[63] M. Gates, *Machine Learning: For Beginners—Definitive Guide for Neural Networks, Algorithms, Random Forests and Decision Trees Made Simple*. Scotts Valley, CA, USA: Createspace Independent Publishing Platform, 2017.

[64] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, no. 2, pp. 197–227, Jun. 2016.

[65] J. Maillo, S. Ramírez, I. Triguero, and F. Herrera, "KNN-IS: An iterative spark-based design of the k-Nearest neighbors classifier for big data," *Knowl.-Based Syst.*, vol. 117, pp. 3–15, Feb. 2017.

[66] Y. Ren, P. Zhao, Y. Sheng, D. Yao, and Z. Xu, "Robust softmax regression for multi-class classification with self-paced learning," in *Proc. 26th Int. Joint Conf. Artif. Intell.* Menlo Park, CA, USA: AAAI Press, Aug. 2017, pp. 2641–2647.

[67] S. A. Alvarez, "An exact analytical relation among recall, precision, and classification accuracy in information retrieval," Boston College, Boston, MA, USA, Tech. Rep. BCCS-02-01, 2002, pp. 1–22.

[68] B. Fogg, "A behavior model for persuasive design," in *Proc. 4th Int. Conf. Persuasive Technol.-Persuasive*. New York, NY, USA: ACM, 2009, p. 40.

[69] M. Dastgir and A. S. Mortezaie, "Factors affecting the end-user computing satisfaction," *Bus. Intell. J.*, vol. 5, no. 2, pp. 292–298, 2012.
[70] S. B. MacKenzie, P. M. Podsakoff, and N. P. Podsakoff, "Construct measurement and validation procedures in MIS and behavioral research: Integrating new and existing techniques," *MIS Quart.*, vol. 35, no. 2, pp. 293–334, 2011.
[71] R. Scheepers, H. Scheepers, and O. K. Ngwenyama, "Contextual influences on user satisfaction with mobile computing: Findings from two healthcare organizations," *Eur. J. Inf. Syst.*, vol. 15, no. 3, pp. 261–268, Jun. 2006.

MUHAMMAD ADNAN received the master's degree in information technology from the prestigious School of Electrical Engineering and Computer Science (SEECS), National University of Science and Technology, (NUST), Pakistan. He is currently a Lecturer with the Institute of Computing (IoC), KUST, Pakistan. His research includes mobile learning, adaptive learning, machine learning, deep learning, and ubiquitous systems. He has submitted his Ph.D. dissertation.

ASAD HABIB received the Ph.D. degree in engineering from the globally renowned Nara Institute of Science and Technology (NAIST), Japan. He has served as the Director of the Institute of Computing (IoC), Kohat University of Science and Technology (KUST), Pakistan. His research interests include data science, natural language processing, computational modeling, mobile learning, software engineering, and adaptive interface designs.

JAWAD ASHRAF received the Ph.D. degree from the Department of Computer Science, University of Leicester, U.K. He is currently serving as an Assistant Professor with the Institute of Computing, KUST. He is also active in research on partner-based scheduling algorithms for advanced reservation environment, K-shortest path variant for routing in reservation environment, mobile learning, trajectory optimization, and novel workflow job selection techniques.

BABAR SHAH is currently an Associate Professor with the College of Technological Innovation, Zayed University, Abu Dhabi Campus, United Arab Emirates. His professional services include but are not limited to Guest Editorships, University Services, the Workshops Chair, a Technical Program Committee Member, and a Reviewer of several reputed international journals and conferences. His research interests include WSN, WBAN, the IoT, churn prediction, security, real-time communication mobile P2P networks, and M-learning.

GOHAR ALI received the Ph.D. degree in real-time communication in wireless sensor networks and mobile communication from the distinguished Gyeongsang National University, South Korea. He is currently an Assistant Professor with the Information Systems and Technology Department, Sur University College, Sur, Oman. He is also supervising students in the fields of real-time big data analytics, power-aware mix-criticality systems, routing in the mobile ad-hoc network and delay-tolerant networks, load balancing scheduling algorithm for grid workflow, and mobile learning. He has vast experience both in academia and industry in South Korea, U.K., Pakistan, and Oman.