

# DEEP LEARNING-BASED METHOD FOR PREDICTION OF SOFTWARE ENGINEERING PROJECT TEAMWORK ASSESSMENT IN HIGHER EDUCATION

<sup>1</sup> LAMYAA ZAED ALJUAID, and <sup>2</sup> MAR YAH SAID

<sup>1</sup>Faculty of Computer Science and Information Technology, University Putra Malaysia, 43400 UPM

Serdang, Selangor, Malaysia

E-mail: <sup>1</sup>aljuaid.lamiaa@gmail.com, <sup>2</sup>maryah@upm.edu.my

## ABSTRACT

One of the very recent applications of machine learning and deep learning is to predict student's performance for a specific task. Particularly in software engineering projects in higher educational institutions, one of the major challenges is how to improve the overall software team performance with the ultimate goal of increasing software team productivity. This problem can be termed as Software Engineering Teamwork Assessment and Prediction (SETAP). To address this, we proposed a SETAP method that uses an effective deep-learning-based method for assessment and early prediction of student learning effectiveness in software engineering teamwork. Further, we identified critical features by using feature importance, which affects the performance of the student's team. For the experimental purpose, we used the freely available dataset from the UCI data repository for student learning effectiveness in software engineering teamwork. In the proposed work, we used Feed-forward deep neural network classifier for this purpose. Deep neural networks can produce high accuracy with complex data very quickly. This made them a perfect choice for our work. To compare and evaluate the performance of the proposed prediction method with others (RF, GLM, GBM), we used accuracy percentage, sensitivity, and specificity as performance measuring benchmarks. Also, we evaluated the method on our university datasets. Results proved that the proposed method performed better than the above-mentioned state-of-the-art classifiers.

**Keywords:** *Teamwork Assessment, Software Engineering, Deep Learning, Stacking.*

## 1. INTRODUCTION

Teamwork is a key ingredient of any successful work in which people of different skill sets are involved at different levels. Particularly, in a higher educational setup, where the team members are students, teamwork becomes more critical as students are not as serious as working individuals. With respect to the domain software engineering projects in higher education institutions such as colleges and universities, one of the major hurdles students and academicians face is how to improve the overall students' software team performance. The goal is to increase students' software project productivity. From the perspective of software engineering, it is important how well we are analyzing student learning effectiveness in software engineering teamwork, which will eventually result in the enhanced overall productivity of the software project team. One classical approach is to empirically analyze data of several projects and then reach a theoretical conclusion. However, this way is

pretty time consuming and the possibility of human bias is very high as the ultimate decision is taken by individual experts based on their thinking. Another way is to use modern-day technologies to predict the above-given quality measures with any human interference. Machine learning and deep learning algorithms are the topmost contenders for doing the job of prediction of early prediction of student learning effectiveness. Therefore in this work, we designed a deep-learning-based method for assessment and early prediction of student learning effectiveness in software engineering teamwork.

One problem in software engineering projects is that it is to determine the project team performance in the initial phase [1]. Until the software project suffers delays and quality issues then the students' project supervisor knows that the project team is not performing as per the project needs. However, if the students' project supervisor can assess and predict team's performance in the initial stages of the software project then, the software team can work on

deficiencies to minimize any software quality issues and delays. Thus, this will increase overall team productivity too. In the proposed work a software engineering teamwork assessment and prediction method is developed which uses a deep-learning-based method for assessment and early prediction of student learning effectiveness in software engineering teamwork.

The objective of this paper is to develop an effective deep-learning-based method for assessment and early prediction of student learning effectiveness in software engineering teamwork. Further, to improve the prediction accuracy by using feature importance to identify critical features (columns), which affects the performance of the student's team the most. By feature importance we mean, which features hold most of the information. This research paper will be helpful for university faculties, students' supervisors, and students for early performance assessment. In addition, computer science researchers can study the domain of software engineering from various perspectives using the proposed method. In addition, students can study and understand how software team performance affects the project and evolves. The target users of the tool will be university faculties, students' supervisors, and students. They can utilize this tool to predict the outcome of their current performance. In case the outcome shows that the current student performance project will suffer, the student and supervisor can take measures in very early stages to enhance students' performance to have a better outcome.

The contribution of this work is twofold. Firstly, the proposed method can enhance the prediction accuracy of failed and successful projects. Secondly, the proposed method is integrated into a GUI based tool, which is user friendly and can be used by supervisors, researchers, and students who are doing software engineering projects to analyze what can be the outcome of their software project.

The paper is divided into five sections. In Section 2, we discussed the performed literature review, whereas the proposed method is introduced in Section 3, where we explained the proposed method in detail. Further, in Section 4, we critically discussed the results and compared them with other algorithms. Finally, the paper is concluded in Section 5.

## 2. BACKGROUND

In this section, we critically examined the present work, which helped us to get the idea of what has been done and what are the research gap in this field.

Teamwork in higher education plays a very important role. Student performance has serious implications on teamwork outcomes. In work [2], focused on developing certain project-specific skills in the team members which are focused on learning outcomes based on quality. The findings of this work tell that quality skill development plays an important role in course planning, content design, and evaluation around the globe. Cohesiveness plays a very important role in enhancing teamwork. Some empirical work is based on antecedents and consequences of team members' cohesiveness. The work [3] analyzed how important role team cohesiveness plays in the outcome of the project. It examines the impact of cohesiveness in individuals. It found that cohesiveness is dependent on quality, satisfaction, and perceived learning. Works [2,3] also establish the fact that quality teamwork can have positive impacts on higher education institutions. In a higher educational setting, effective teamwork is critical to software engineering projects because every team member has a duty and needs to effectively collaborate for the desired project outcome.

A lot depends on teamwork in software engineering. It helps to shape how smoothly and profitably a software development project is carried out [4]. Software project success in educational institutions depends a lot on how well the software engineering team which consists of students performed their tasks [5]. A poorly planned and managed software development results in critical quality issues and hurts the overall reputation of the department, and supervisor [6]. The primary factor of any mismanagement of the software development process is the issue of lack of task-based expertise of software development team members (students) and lack of training. Another issue that hurts the quality of software development and results in delays is the lack of effective communication between the software team. To identify students in a software team who are not experts, not well trained, and have learning issues in a software project development team is a critical task [7]. In this section, we further discuss machine learning, deep learning, and feature selection in detail. The favorable results of the project depend on many aspects such as the quality and reliability of the project. Data mining and machine learning techniques can be applied to find anomalies and shortcomings of these projects to make them more robust. The prediction of failure of the project can reduce the efforts of the team and the efficiency of the project or software [8]. Nowadays many projects have been started but the success rate is very lower. This is a general scenario that almost

1 project becomes fail out of 3 planned projects. So predictive analysis of project measurement is very necessary. Although still there are very few models for the predictive success rate. There one more model is suggested based on the project metric before the initiation of a project [9].

## 2.1 Machine Learning

Machine Learning is the branch of computer science, which is used to develop artificial intelligence. In this technique, the respective machine learning algorithm gets trained to perform a particular task. Once the algorithm gets trained, it acquires the ability to perform a particular task without any human interference. Classically there are two branches of machine learning which are Supervised and Unsupervised learning. Supervised learning can further be broken down into two sub-branches which are classification and regression [9].

Supervised learning as two types of variables which independent and dependent variables. In Supervised learning, the algorithm takes independent variables as input and predicts a dependent. It is a type of machine learning where we predict a categorical variable. It is also known as classification, where gender can only be of two types which are male or female. The algorithm is trained to predict gender, as an input, it will only take an independent variable and predict the dependent variable. By categorical variable, we mean the variables which can be identified into a certain group based on its name. Such as gender (male or female), true or false, and yes or no. In classification, our algorithm trained to predict a categorical variable, as an input algorithm will take independent variables ( $x$ ,  $y$ ,  $z$ ) and categorical variables. After this, it acquires the ability to predict dependent variables which is a categorical variable. In the proposed project, we have fail (F) or pass (A) as a categorical variable. Here algorithm gets trained to predict gender, as an input it will only take independent variables and predict the dependent variable. Regression is a type of machine learning where we predict non-categorical variables, where average can be any numeric value. Here the algorithm trained to predict average, as an input algorithm will only take independent variables ( $x$ ,  $y$ ,  $z$ ) and predict dependent variable which is average. Unsupervised machine learning is a branch of machine learning where we group data items in 'n' number of groups. It is also known as clustering. The data items in a particular group are closely related to each other. These groups are also known as clusters [9]. Unsupervised machine learning is similar to classification, with just one major difference which is in Unsupervised machine learning we do not have any categorical

variable prior to data. We need to find data items that can be grouped based on some similarity matrix such as Euclidean distance, and Manhattan distances, etc.

Machine learning today is used in every domain and solving some of the most critical problems very efficiently [10]. Aljuaid et. al. [11] critically examined what is done in machine learning, what areas need attention, and several open research problem where machine learning can play a key role to develop solutions. Random Forest [12], Support Vector Machine [13], Naïve Bayes [14], K-Nearest Neighbors [15], and Neural Networks [16] are some of the popular choices. For software development, effort estimation in a software project is a critical and important task. Effort can be termed as any work that is done in a software project. Such as coding is an effort, writing a project report is an effort, giving a project presentation is an effort. More efforts mean more human resources, which adds to the cost of the project. Through this project, the leader knows how much total effort is needed and can plan and schedule the software project delivery by computing the cost needed. Monica and Sangwan [17] analyzed and reviewed several machine learning algorithms that are used to predict the efforts using several state-of-the-art algorithms such as neural networks, fuzzy logic, etc. There exist several other works also which use machine learning for this purpose such as [18], [19] where novel machine learning schemes are introduced. Also, machine learning is successfully used for software release time prediction [20], [21], software bug prediction [22], and software cost prediction [23]. Some of the machine-learning algorithms are discussed in the following paragraphs.

Support Vector Machine (SVM) is one of the most widely used supervised machine learning algorithms and it is highly accurate. It is used for both classification and regression problems. The hyperplane is used to separate the classes [13]. The hyperplane is a line which distinguishes between data point. SVM can use linear and as well as nonlinear hyperplane to distinguish between classes which makes it a really good choice for binary as well as multi-class problems. The advantage of SVM is it is accurate. However, the biggest drawbacks are slow and have a longer training period.

Another machine learning classifier is Random Forest (RF) which is one of the most widely used supervised machine learning algorithms and it can give very high accuracies. It uses the technique of bagging, where data is divided into several subsets

and algorithms get trained on these datasets. Finally, all models are merged to form a single comprehensive model [12]. The advantage of RF is that it is accurate. However, the biggest drawback is that its complexities increase as several trees grow and have a longer training period. Naïve Bayes is one of the simplest and easy to use supervised learning algorithms. It is also called probabilistic classifiers where Bayes' theorem is applied [14]. It is a highly scalable algorithm. However, it's not as highly accurate as SVM and RF.

The Generalized linear model (GLM) is a classification algorithm that consists of three parts: the systematic part represents the dependent variable; the random part represents the independent variables and the link function. GLM allows the linear model to relate with the response variable with the help of link function and it allows variance magnitude of each measurement so they can be a function of the corresponding predicted values [28]. Gradient Boosting Machine (GBM) is a supervised machine-learning algorithm used for classification. It is used for stage-wise model development based on the boosting principle [27]. The classification is done using a large number of decision trees, where the first tree is built based on the weighted average of the dependent variables in the dataset. The main difference between RF and GBM is that the decision trees in RF are built in parallel, while decision trees in GBM are built sequentially based on the error from the previous three.

## 2.2 Ensemble Learning Techniques

Ensemble methods are learning algorithms used to build a set of classifiers and then classify new inputs by taking a weighted probability of the prediction result. The main goal of using ensemble learning is to achieve highly accurate predictions [25]. Boosting is a technique used in the machine learning domain in which several classifiers (weak learner) are combined, which can result in better classification accuracy than using a single classifier [24].

Bagging is also known as bootstrap aggregating, is a technique used in machine learning in a domain that aims to enhance stability and improve the accuracy of a classifier. It is a technique that helps to minimize variance and is effective to deal with overfitting. Although it is usually applied to decision tree methods, it can be used with any type of method [25]. Another technique is Stacking which is used in the machine learning domain that explores different classifiers for the same problem. The main principle is used in stacking is that learning problem with

different classifiers. However, in stacking, the problem is simply divided into parts and each classifier only solves the assigned part. This intermediate prediction is used as input from a new level classifier [26].

## 2.3 Deep Learning

The world has witnessed tremendous growth in the use of Deep Learning algorithms in the last 5 years. Today they are the most widely used algorithms [29]. Deep learning is used in every field to give us a better understanding of the things around us [30]. Deep learning algorithms mimic the human neural network and are now used in almost every domain. Such as natural language processing [31], biomedical [32], robotics [33], chemistry [34], physics [35], Elearning [36], social media [37], etc. These are some areas where deep learning is breaking all previous records in terms of accuracy. Within deep learning, there are several algorithms. Deep learning can also be seen as the extension of classical neural networks where we have an input layer, hidden layer, and output layer. Input layer contains input data, the hidden layer where random weights are assigned to input data values and then the kernel is used to understand those values. Further, the output layer consists of the labeled output. However, in classical neural networks, we have a very low limit of layers and neurons per layer. Whereas in deep learning, we can assign several layers with hundreds of neurons each layer [38].

The major advantages of deep learning are [29,30]:

- **Parallel computing:** These algorithms can process and compute in parallel that means a task will be broken down into several subtasks and the system will process each subtask in parallel with other tasks. In the end, all the results of the subtask will be combined to have a final result. This speeds up the processing too.
- **Distributive:** This means deep learning can acquire data from various locations for the same task. It need not be at the same location. Also, it can process at several locations and can have a unified result in the end.
- **Highly accurate:** This deep learning uses matrix algebra to compute and is highly accurate. However, no one knows, how it concludes as it's a black box technique by nature.

- Understand complex data: Deep learning can understand complex relations between the data which other classes of algorithms failed to understand.

With several advantages, deep learning has few disadvantages too, such as they are resources exhausted in nature. This means, need a lot of memory and processing power which directly corresponds to the cost. Parameter tuning is very complex in deep learning. A single wrong assumption in parameter settings can increase errors. From the application domain perspective, recently deep learning makes way into the software engineering domain [39,40]. However, it is challenging how the researcher will map software engineering problems to take benefits from deep learning as stated by Li [40]. One of the very recent applications of them is to predict student's performance for specific tasks [1]. Particularly in software engineering projects, one of the major challenges is how to improve the overall software team performance with the ultimate goal of increasing the software team productivity is important for overall project success. From a software engineering perspective, we need to do a highly accurate assessment and early prediction of student learning effectiveness in software engineering teamwork. This can also greatly help in enhancing the overall productivity of the software project team as individual members will be able to know, earlier than actual his/her performance. As the team will be more focused now and over goal-oriented. In work [41] relates various machine learning methods to software engineering problems. In the eLearning domain, the use of deep learning is an emerging trend. Muniyasamy and Alasiry [42], analyzed how deep learning is now impacting the eLearning domain. They also discussed how deep learning will impact eLearning in the future. Alam et al [43], used a Multilayer perceptron deep learning algorithm for providing ubiquitous teaching, Learning, and dynamic educational content delivery. Another work [44] focused on the use of deep learning for eLearning. In [44], the system learns to deliver educational content to learners based on learner educational profiles using deep learning. This work highlights the importance of learner-centric education. With reference to the above literature, we can see that deep learning algorithms have the ability to affect the eLearning domain positively.

A feedforward deep neural network is one type of deep learning algorithm [45]. A feedforward deep neural network uses multi-layer perception

architecture. It is successfully used in several commercial deep learning packages such as H2O which is used by Amazon [46]. A classical Feedforward deep neural network consists of an input layer, hidden layers, and the output layer. The input layer consists of data that needs to feed to the algorithm. The hidden layer consists of the kernel which has the logic of how to differentiate between different data instances. Whereas, the output layer consists of the output produced by the hidden layer. As the name suggests it is feedforward, which means it can only move in forward, it cannot move backward. Alam et al. [47], proposed a feed-forward deep neural network-based road detection method for autonomous vehicles. Alam et al. [47], they used 9 feature to predict road or no road class. To improve prediction accuracy, they used decision fusion which can be defined as the best possible combination of predictions about the same input data from multiple classifiers. For evaluating the results, in [47] authors used accuracy, sensitivity, specificity, and area-under-the-curve as performance evaluation benchmarks. A convolutional deep neural network (CNN) is a type of deep neural network particularly used for image recognition problems. CNN is highly complex as compared to Feedforward deep neural network. Parameter tuning of CNN is a difficult task. CNN particularly are used for image recognition problem as they produce a very high detection rate.

## 2.4 Feature Selection

Feature selection is another area where machine learning is successfully used. Feature selection is a technique by which we reduce high dimensional data and only keep features that hold most of the information [48]. Miao and Niu et al [48], defined Feature selection, as a dimensionality reduction technique, aims to choose a small subset of the relevant features from the original features by removing irrelevant, redundant, or noisy features. Feature selection usually can lead to better learning performance, i.e., higher learning accuracy, lower computational cost, and better model interpretability. Recently, researchers from computer vision, text mining, and so on have proposed a variety of feature selection algorithms and in terms of theory and experiment, show the effectiveness of their works.

There are several machine learning-based feature selection techniques that are today used to detect faults in software such as [49,50]. This helps greatly to predict software faults in the early stages. Thus, is far easier to rectify the faults and it will also be cost-effective. Most of the feature selection techniques are static. However, there can be scenarios where

static technique will not be the best choice where feature importance changes (not constant). In these cases, it is possible that over a period-of-time the static feature selection technique will not give the best accuracy. Therefore we need an adaptive feature selection technique that adapts as per changing feature importance as proposed in [51–53]. The various categories in which we can group feature selection methods are discussed in proceeding sub-subsections.

#### 2.4.1 Filter methods

In Filter methods, statistical measurements are used to score every feature which means each feature is ranked based on the score. Further, cut even point can decide to select top-ranked features. A few examples are correlation coefficient scores and information gain. One of the works [54], proposed a filter-based method for feature importance for improving the Classification of Portable Executable Files.

#### 2.4.2 Wrapper methods

In Wrapper methods, a search problem is used to select the best features. Here various combinations of features are formed and then the predictive model is used to evaluate each combination based on accuracy achieved. The search methods such as best-first search, and a random hill-climbing algorithm, and any brute force method. One such method has been proposed in [55] where the ensemble is used for feature selection which is the combination of sequential backward selection (SBS), sequential forward selection (SFS), and evolutionary selection.

#### 2.4.3 Machine learning-based

In Machine Learning based feature selection methods, some machine learning-based methods are used to predict feature importance. In one such work in [56], the feature selection method is proposed which uses Random Forest. These selected features are further used to predict heart diseases. In [57], a 2-Step based feature selection method is proposed for improving Cancerlectins prediction. Similar to [56], [57] also uses the Random forest for finding feature importance. Another important and popular method that is used in Principle Component Analysis (PCA) to know feature importance. Such as in [58], for improving the prediction of neurodegenerative disorders PCA has been used.

### 2.5 Research Gaps

Petkovic et al [1], proposed a Software Engineering Teamwork Assessment and Prediction (SETAP)

project. The aim is to develop effective machine-learning-based methods for assessment and early prediction of student learning effectiveness in software engineering teams. Specifically, in [1] authors use the Random Forest (RF) machine learning (ML) algorithm to predict the effectiveness of software engineering teamwork learning based on data collected during student team project development. These data include over 100 objective and quantitative Team Activity Measures (TAM) obtained from monitoring and measuring activities of student teams during the creation of their final class (outcome) of the project in the joint software engineering classes which ran concurrently at San Francisco State University (SFSU), Fulda University (Fulda) and Florida Atlantic University (FAU). In work [1], the authors provided the first RF analysis results done at SFSU of their full data set covering four years of joint SE classes. These data include 74 student teams with over 380 students, totaling over 30000 discrete data points.

Two major research gaps are identified by us with respect to SETAP and related literature in this section which are:

- Only a few pieces of literature exist in the area of SETAP, which also pretty limited work.
- In [1], introduced SETAP method which uses classical RF simply to predict. No further result comparisons are given with other state-of-the-art classifiers. However, we believe the result of SETAP can be improved further if the more specialized and tuned technique are used.

In this work, we address the above-mentioned research gaps. The proposed method uses stacking for SETAP, where we used classifiers such as Gradient Boosting Machine (GBM), generalized linear model (GLM), and Random Forest (RF) and Deep Learning (DL) as level-I learners. The Generalized linear model (GLM) allows the linear model to relate with the response variable with the help of link function and it allows variance magnitude of each measurement so they can be a function of the corresponding predicted values [25]. Gradient Boosting Machine (GBM) is a supervised machine-learning algorithm used for classification. It is used for stage-wise model development based on the boosting principle [26]. RF algorithm is a supervised machine learning algorithm that is used to do classification [27]. It selects random samples then for every sample data it constructs a decision tree. After this, it collects predicted output for every decision tree. Then it used the concept of voting and

class, which has the most votes will be selected as the output.

### 3. THE PROPOSED METHOD

In this section, we discuss the methodology of the proposed work. This section is divided into 4 subsections. In subsection 3.1, we discuss the platform, tools, programming languages we used for this work. In subsection 3.2, we discuss the dataset. In subsection 3.3, we introduce our method. Finally, in subsection 3.4, we introduce the prediction tool.

As depicted in Figure 1, the methodology of developing the proposed method consists of three phases: (1) data processing, (2) designing the prediction method, training the proposed method, tuning the proposed method, and (3) testing the proposed method, and method evaluation.

**Phase-1:** This phase is about data processing; we started with downloading the data from the UCI repository. The downloaded data has multiple data files. Therefore, we need to merge them as a single dataset file. We found out some issues in the dataset that need to be processed. These issues are:

**Unnecessary details:** In all “.csv” dataset files, the top few rows contain data information. However, we need these files clean and in proper table format. So, this extra information should be deleted. For this purpose, we manually removed unwanted data from the header to each file structure properly.

**Missing values:** In the dataset, several cells contain missing values. We have two options to deal with this issue first completely remove or to impute the

missing values. We decided to impute the missing values because their number is small and may contain important information. We used a data imputation algorithm is used to predict the missing values.

The feature selection technique works by assigning importance scores to the features, and based on these scores, we decided which features are considered. After the feature selection technique is performed, and the most important features are selected. Now our final dataset is ready to be divided into training and testing.

**Phase-2:** In this phase, we designed our method, which classifies the project outcome. After designing our method. We were training the prediction method. The training process is about providing the prediction method with the seen dataset to learn the patterns of the data. For training purposes, we used 70% of our dataset. Further, we have to tune the parameters by choosing the best values. As depicted in Figure 1, training and tuning are an iterative process until we achieve the best performance and desired prediction accuracy.

**Phase-3:** In this phase, we test the proposed method on the testing dataset (unseen data). We compute the confusion matrix to describe the performance of the prediction method and record the results. Further, these results are compared with results of other state-of-the-art algorithms.

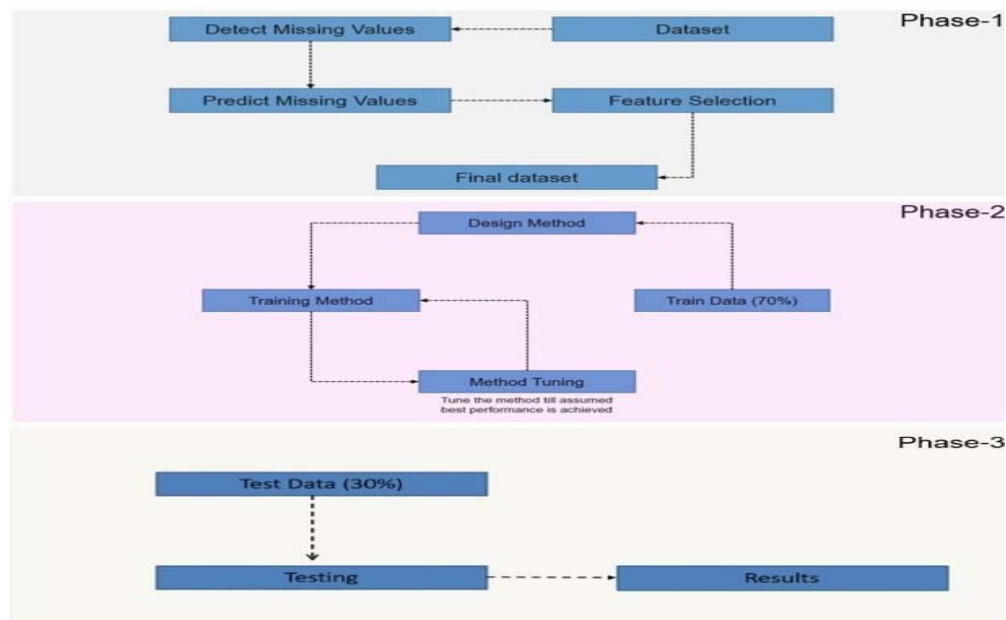


Figure 1. Phase-Wise Block Diagram Of The Proposed Method.

### 3.1 Experimental Setup

In this section, we gave details for which programming language was used for the proposed work, and what the major packages that are used. To simulate the proposed method, we used the R statistical programming platform. We used the H2O package in R for deep learning feed-forward neural networks. H2O package is the product from H2O.ai [58]. It is an open-source tool and widely used today for several commercials' purposes in the domain of finance, insurance, social media, and healthcare. Today around 9000 business and 80000 machine learning experts depend on the H2O package for performing prediction which further helps in decision making and acquiring intelligence. Using

H2O one can take advantage of parallel and distributed computing. For feature selection, we used the FSelector package [59]. It tells which feature holds most of the information. For dividing the dataset into training and testing, we used the caTools package [60]. We used R packages such as caret [61] for designing a confusion matrix, which helps us in measuring the performance of the proposed method and for comparing the proposed method with other algorithms. All the experiments are performed on RStudio which is IDE for R programming [62]. Finally, for data visualization, we used ggplot2 package [63]. All experiments are performed on Windows OS with 1 GB RAM. Various R packages used for experimentation in this work as stated in Figure 2.

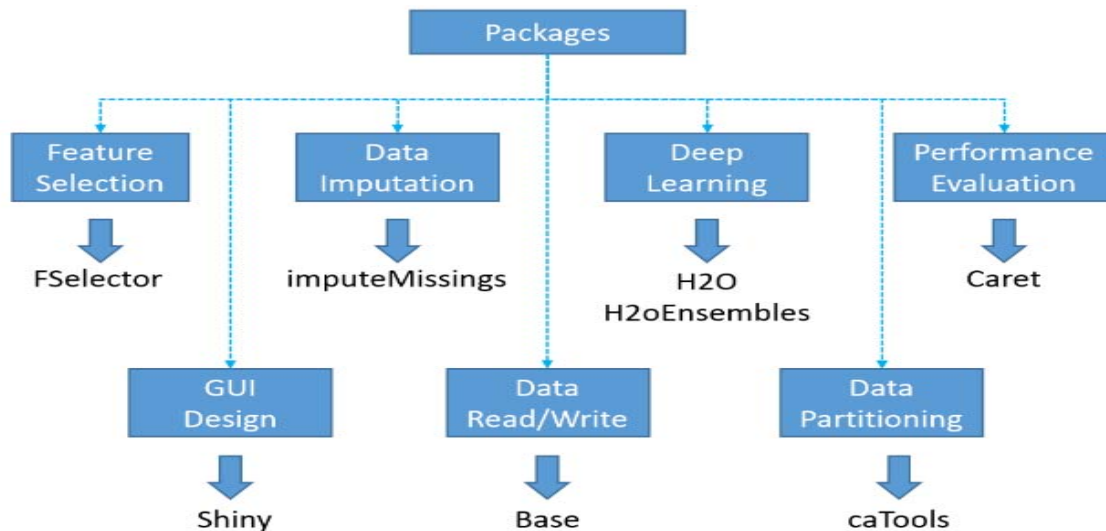


Figure 2. Various Packages Used In This Project.

### 3.2 Dataset

We first downloaded the dataset from the UCI data repository [64]. We selected product-wise prediction; files are stored in form of “.csv” file format. The Dataset includes 116 features and outcomes grades fail (F), and pass (A). We need to predict failure or pass. Once we downloaded all the 11 files are merged in one “.csv” file together to form a single dataset. Then we checked the dataset for missing values. As the dataset contains missing values, we predicted all rows, which have missing attributes in them.

Once we predicted all missing data instances. Then we saved our final dataset which can be used for training and testing. After preparing the final dataset, we computed the feature importance of each feature and selected the top most features. We used 95% information as the cut-off criterion for selecting the

features. Once we prepared our final dataset, we divided the dataset into two parts that are training data which is 70% of the original data, and testing data which is 30% of the original data. Data is divided using a package called caTools. Dataset is divided in such a way both training and test dataset does not contain any duplicates. Some of the important features are stated in Table 1, with their explanation.

Further, university data is collected from Web Engineering Subject SSE5305. This data is collected from 3 teams at different time intervals. This data is used for further evaluation of the method. Dataset is freely available on UCI data repository and data collected have no personal and sensitive information in the dataset is collected which in any sense affects the privacy of any individual and organization and all law of the land are followed. Dataset used and data collected is not at all of any sensitive nature.



Table 1. Description Of Some Independent Features.

Independent Features	Description
Year	During which year, these data were collected.
Time interval	It refers to the software development stages, the semester divided into 5 phases which are requirement gathering, design, development, testing, and deployment.
Team number	Each team has a specific ID.
Semester ID	During which semester, these data were collected.
Team Member Count	The number of members in the team.
Female Team Members Percent	This value is obtained by divide the number of female members by the number of male members.
Leader Gender	It can be Female or Male.
Team distribution	It can be "Local" if the team members within the same university. Or "Global" if the team members from different universities.
Member Response	How many times a team member responded.
Leader Response	The time that the leader spent on management tasks.
Meeting hours	It refers to the time members spent on meetings.
Help hours	It about the time that a team member spent helping each other.
Coding tasks	The time members spent working on coding deliverable tasks.
Non-coding tasks	The time members spent working on non-coding deliverable tasks such as Planning.

Commit Message	It is about committing to the code repository.
Issue	Issues observed by the lecture.

### 3.3 Proposed Method

We believe that if we combine the prediction of multiple algorithms in a way that we couple all results together we can obtain better accuracy than the single algorithm. Therefore, we used the concept of Stacked Generalization where we need to explore a use variety of models (classifiers) to address the problem. The idea is to use multiple classifiers to solve a prediction problem and later combine the results by using a weighted average of all the classifiers at level one classifiers. We explain the whole process in Figure 3. For level-I classifiers, we used Gradient Boosting Machine (GBM), generalized linear model (GLM), Random Forest (RF), and Deep Learning (DL). There is no rule, that specifies which classifier is to be used at level-I. It depends on personal choice, available packages, and computation resource availability. Generally at Level-I, classifiers from different families are used. For example, it is not a good idea to use different (more than one) decision tree classifiers at Level-I because not much improvement will be expected, if classifiers that use the same logic are used.

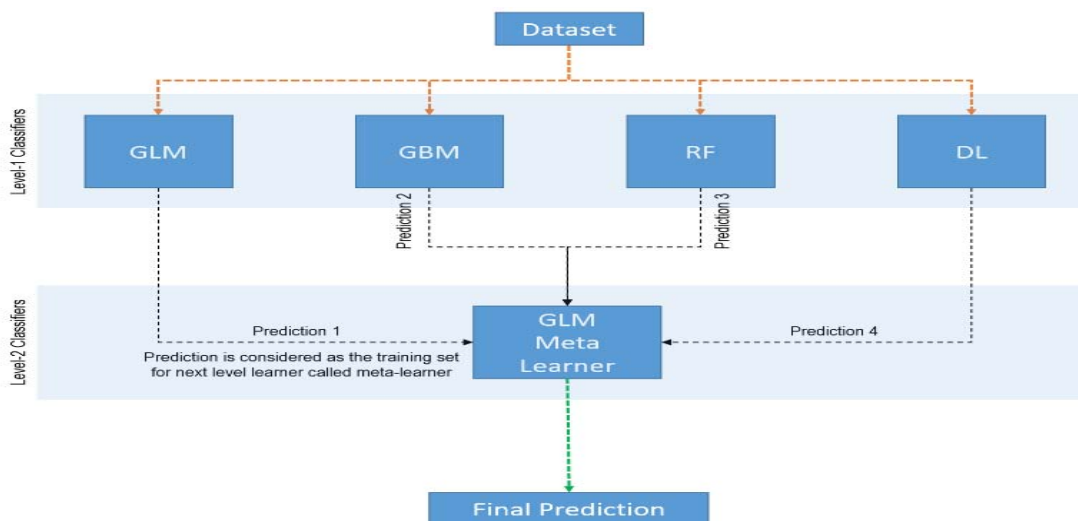


Figure 3. Integration Of Various Classifiers As Stacking For The Proposed Work.

### 3.4 Proposed Tool

After developing the proposed method, we integrated it in a tool that has a graphical user interface, the same is depicted in Figure 5. For developing the tool, we used the Shiny app development package in R. Shiny package makes it easy to build interactive stand-alone desktop and web applications from R.

Shiny applications have two components, which are:

- Server Component (Server.R): This file contains all the logic that is needed to convert the input to the desired output.
- User Interface Component (UI.R): All the code that is needed for developing the user interface belongs UI.R file. It gives interactivity to our tool by dynamically managing the input and the output on the screen.

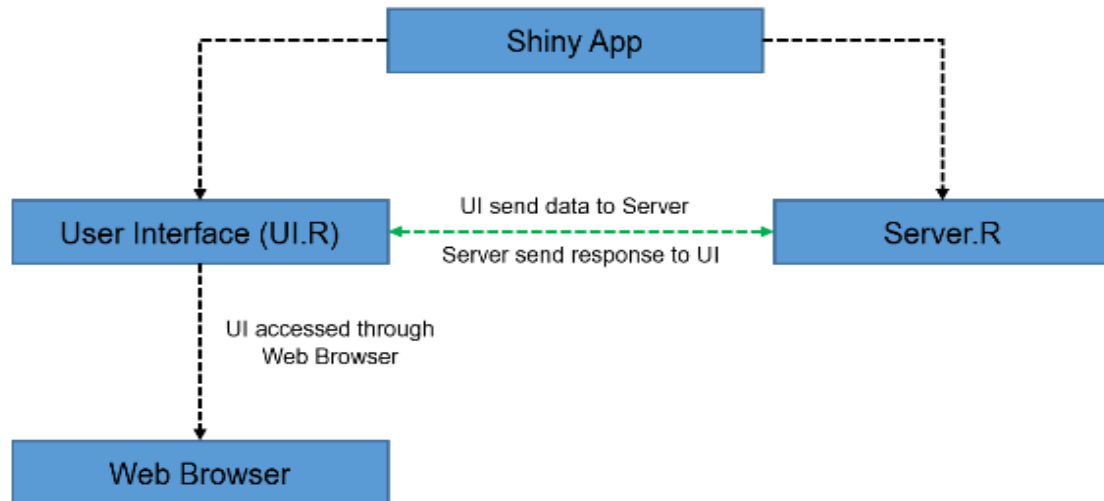


Figure 4. Shiny Application Components.

We can save UI.R and Server.R separately. Once UI.R will run, it will call Server.R too. Also, we have the option to code UI.R and Server.R in files in a single R file by naming two separate functions namely UI and Server in case we are making standalone desktop applications where UI and Server component lie resides in the same system. To call the application from the command line we can use the shiny app (ui, server) command. Also, a button is given at the top right corner to run the application. The proposed tool GUI is depicted in Figure 5. The GUI contains data entry text boxes. Once the user will enter data, he or she can select a machine learning algorithm from the given buttons on the right top corner. Also, a button is given for the proposed method.

## 4. RESULT

For evaluating the effectiveness of the proposed method, we compared our result with the other three state-of-the-art algorithms, which are the Generalized linear model, Gradient boosted model and Random Forest. We used accuracy percentage, confusion matrix, sensitivity, and specificity as performance measuring benchmarks [65]. A confusion matrix is a table that shows actual versus

predicted data labels. The sum of the diagonal of the confusion matrix represents the correctly classified data label, thus can be used to compute classifier accuracy. Sensitivity can be defined as the proportion of actual class labels, which are correctly predicted by the classifier. Specificity is the ability of the classifier to identify negative results. Important terms used to calculate sensitivity and specificity are a number of true positives (TP), number of true negatives (TN), number of false-positive (FP), and number of false negatives (FN) respectively. We evaluated the proposed work in three ways. Firstly, we evaluated the proposed method for testing the benchmark dataset. Secondly, we integrated the proposed method in form of an application and evaluated that application. As an output prediction must be either “A” or “F”. Here class “A” means project successful and class “F” means the project is a failure. These grades are measured based on the 100 inputs of a particular team. Finally, we tested the proposed method using our university data which we collected from Web Engineering Subject SSE5305. For comparing the results of the proposed method, three state-of-the-art algorithms are selected which are GBM, GLM, and RF.

## Software Engineering Teamwork Prediction System

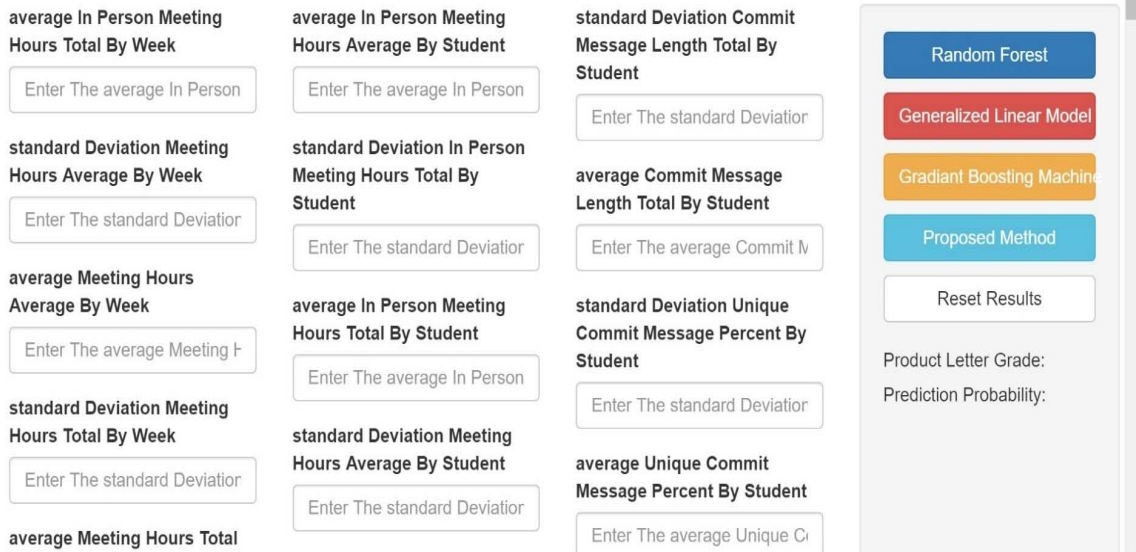


Figure 5. GUI Of The Proposed Tool.

### 4.1 Feature Selection

In the proposed method feature importance is computed to select the most important features in this dataset. The top 109 features are used and six features are not used as they hold no or very little information.

### 4.2 Evaluation-I

In Evaluation-I, the performance of the proposed method on the benchmark testing dataset is evaluated. The performance evaluation benchmarks such as prediction accuracy percentage, sensitivity, and specificity, are depicted from Figure 6 to Figure 8 of GBM, GLM, RF, and the proposed method. In

Figure 6, we depicted the prediction accuracy percentage. We see that the proposed method (yellow bar) is more than the other three algorithms. RF performed worst in terms of prediction accuracy percentage. In Figure 7, we depicted the sensitivity. In terms of sensitivity, the proposed method outperformed the other three algorithms. Whereas in Figure 8, we depicted specificity, here also proposed method outperformed the other three algorithms. In all cases, the performance of RF is the worst. Figure 6 to Figure 8, proves that the proposed method outperformed other algorithms in terms of accuracy, sensitivity, and specificity.

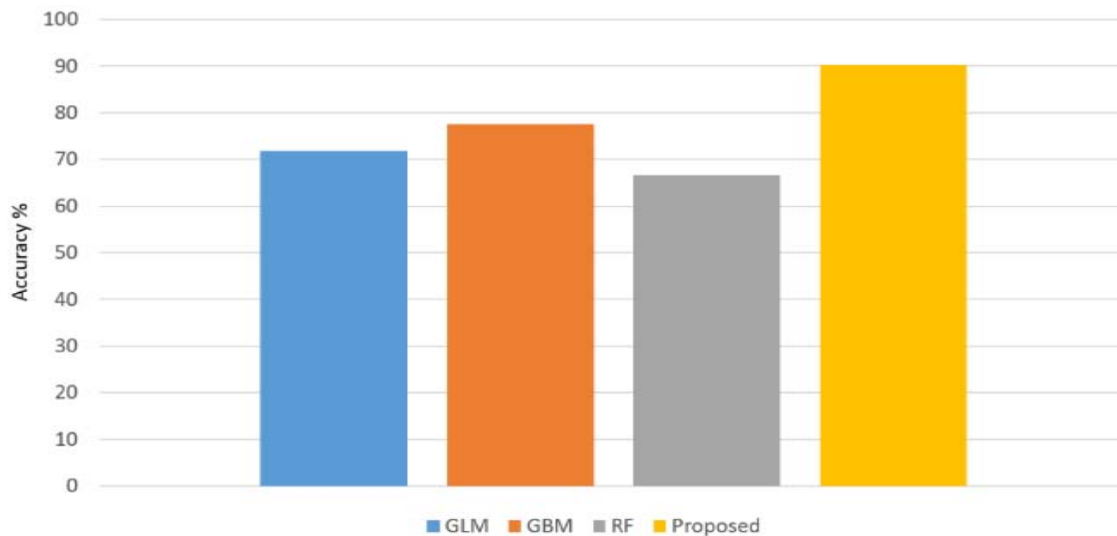


Figure 6. Predictions Accuracy Of GLM, GBM, RF, And Proposed Method.

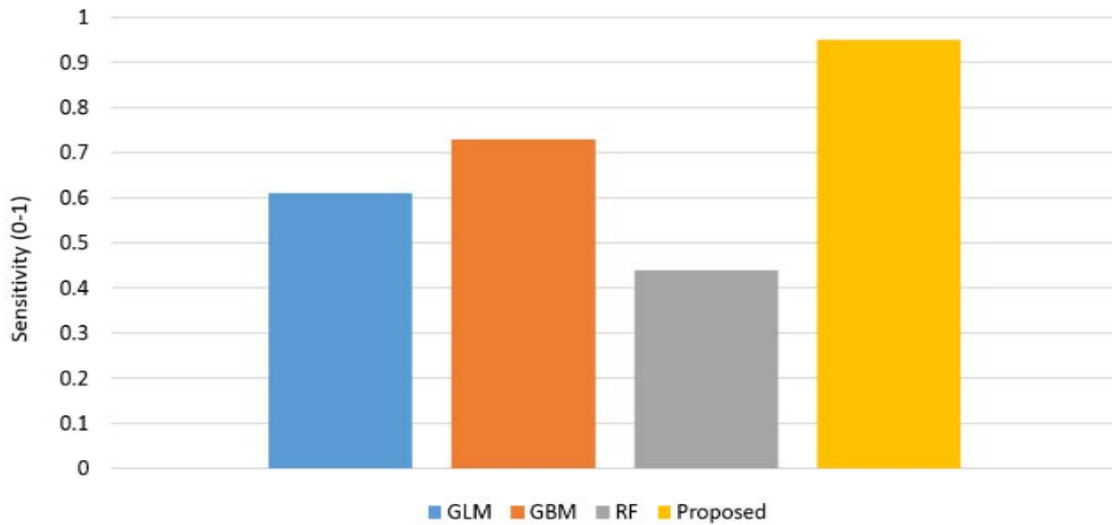


Figure 7. The Sensitivity Of GLM, GBM, RF, And Proposed Method.

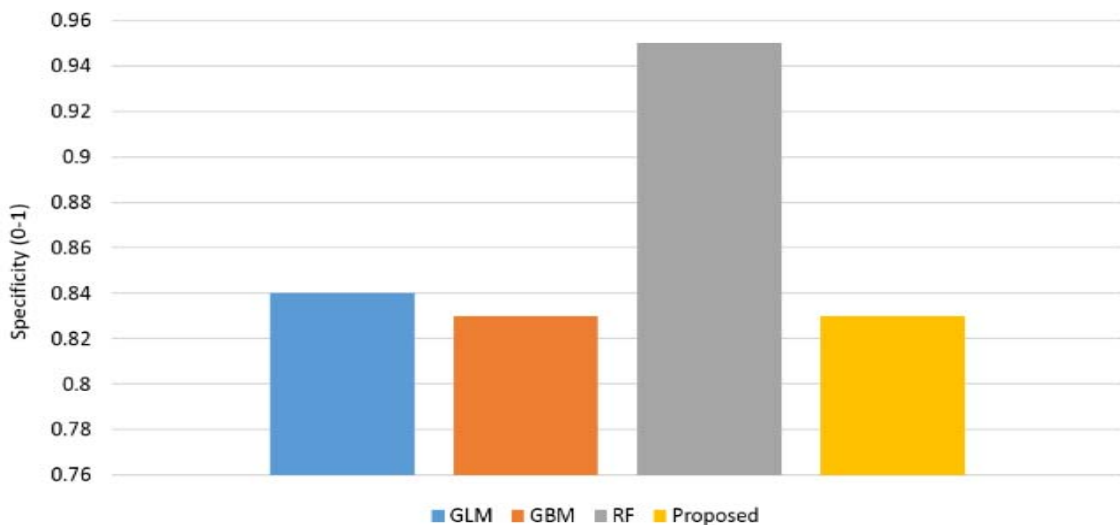


Figure 8. The Specificity Of GLM, GBM, RF, And Proposed Method.

### 4.3 Evaluation-II

In evaluation-II, the proposed tool is integrated with the proposed method and tested which is illustrated in Figure 9 to Figure 12. After filling the details, the user can click on one of the algorithms. As a result, prediction probability will be displayed. Prediction probability lies between 0 to 1. If it is more than 0.50 for a particular class, that class will be considered as output. Prediction probability nearer to 1 means, very strong prediction, and if closer to 0.50, then it means weak prediction. For our test data, all algorithms and proposed methods predicted the correct class which is A. However, the prediction probability of GBM, GLM, and RF are respectively 0.63, 0.70, and 0.60. Whereas the prediction

probability for the proposed method is 0.91, this shows the proposed method predicts very strongly as compared to the others.

As the data entered in the text boxes are the same for all methods, we showed the full GUI in Figure 9. Whereas for Figure 10-12, we have only shown the right top corner of the GUI that presents the results.

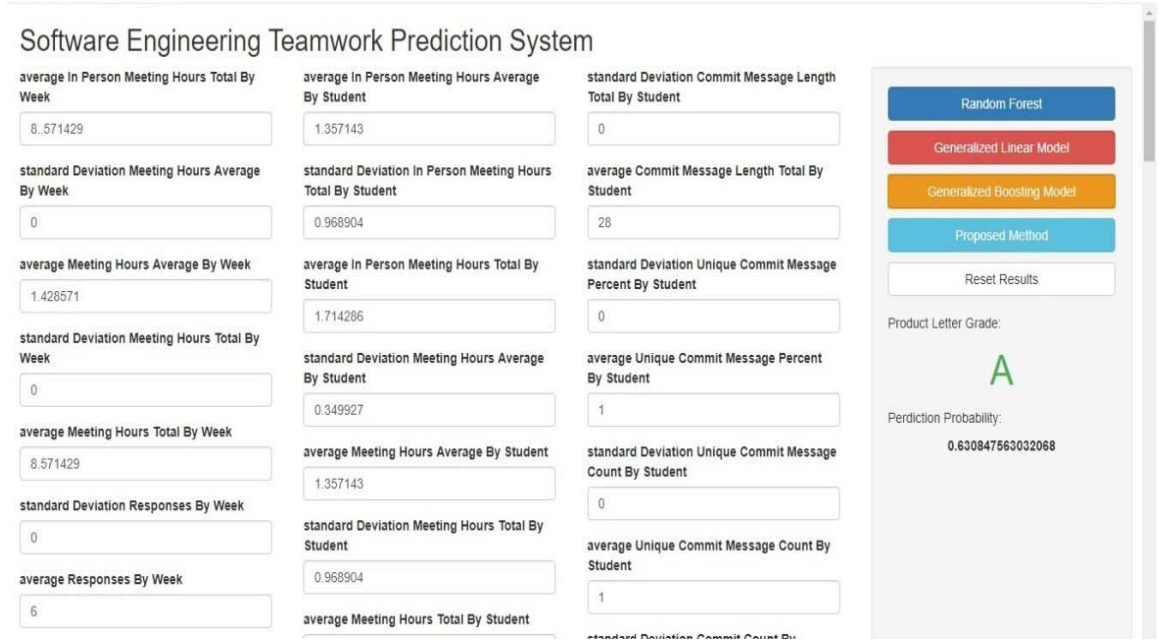


Figure 9. Prediction Using GBM.

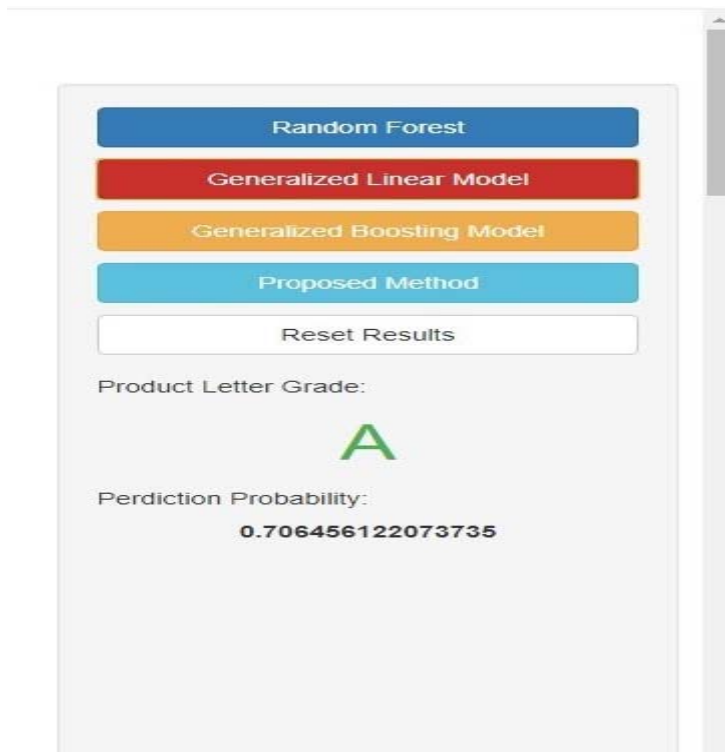


Figure 10. Prediction Using GLM.

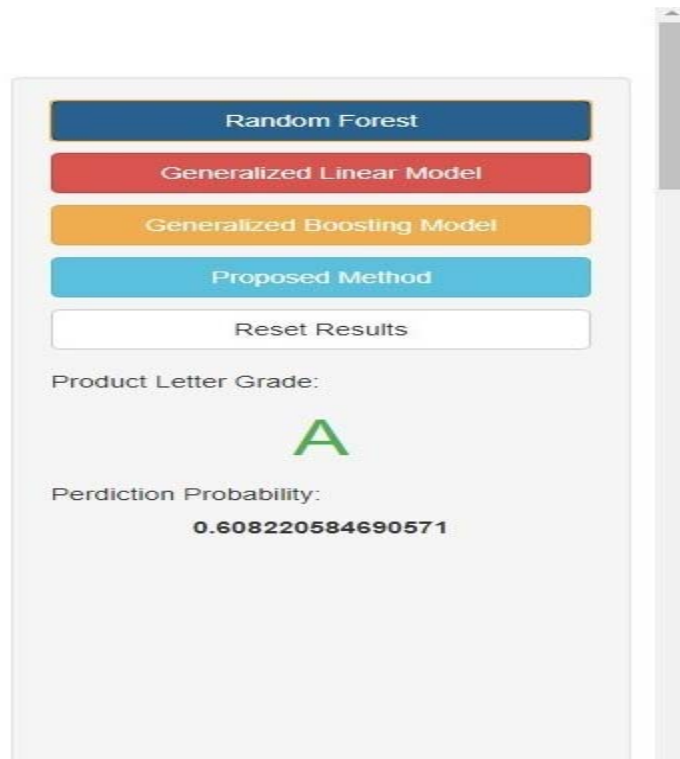


Figure 11. Prediction Using Random Forest (RF).

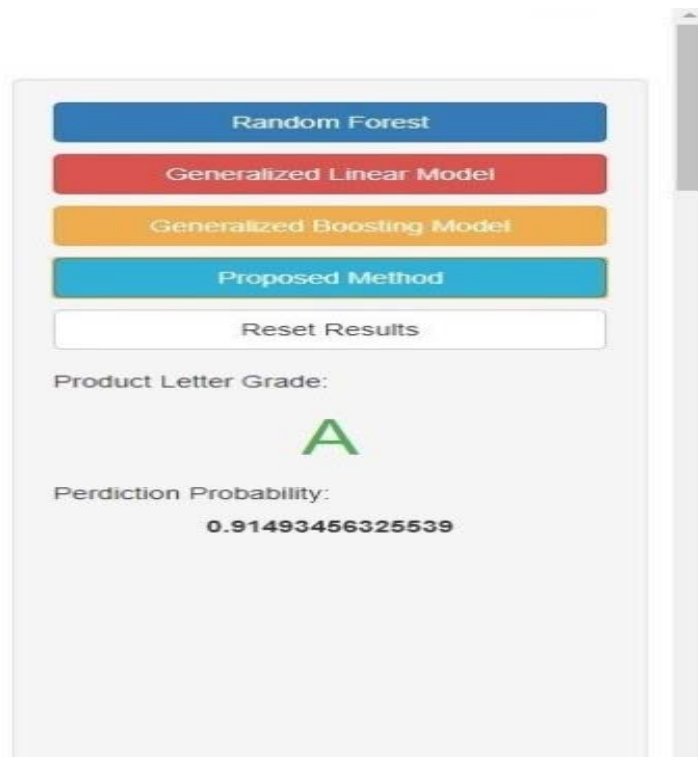


Figure 12..Prediction Using The Proposed Method.

#### 4.4 Evaluation-III

For evaluating the prediction tool using our university data which we collected from Web Engineering Subject SSE5305. After collecting the data, we evaluated the proposed method by predicting using Web Engineering Subject SSE5305 data. Further, we compared the predicted results from RF, GLM, GBM, to prove that the proposed

method performed better than the others, the same is stated in Table 2, Table 3 and Table 4. For all cases that we see in the given Tables, the prediction probability of the proposed method is better than the RF, GLM, and GBM. This comparison shows that the proposed method is a good option to predict even with data from different universities.

Table 2. Prediction Results Of Teams At Requirement Gathering Phase.

		Methods			
Team 1	Product Grade	RF	GLM	GBM	PROPOSED
		Prediction Probability	0.60	0.70	0.60
Team 3	Product Grade	RF	GLM	GBM	PROPOSED
		Prediction Probability	0.60	0.69	0.55

Table 3. Prediction Results Of Teams At The Design Phase.

		Methods			
Team 1	Product Grade	RF	GLM	GBM	PROPOSED
		Prediction Probability	0.74	0.60	0.60
Team 2	Product Grade	RF	GLM	GBM	PROPOSED
		Prediction Probability	0.66	0.61	0.53
Team 3	Product Grade	RF	GLM	GBM	PROPOSED
		Prediction Probability	0.50	0.56	0.61

Table 4. Prediction Results Of Teams At The Development Phase.

		Methods			
Team 1	Product Grade	RF	GLM	GBM	PROPOSED
		Prediction Probability	0.56	0.73	0.55
Team 2	Product Grade	RF	GLM	GBM	PROPOSED
		Prediction Probability	0.46	0.73	0.64
Team 3	Product Grade	RF	GLM	GBM	PROPOSED
		Prediction Probability	0.72	0.77	0.66

## 5. CONCLUSION

Today deep learning algorithms are successfully used in almost every field. Such as natural language processing, biomedical, robotics, physics, etc. One of the very recent applications of them is to predict student's performance for a specific task. Particularly in software engineering projects, one of the major challenges is how to improve the overall software team performance with the ultimate goal of increasing software team productivity. From the software engineering perspective, we need to do a highly accurate assessment and early prediction of student learning effectiveness in software engineering teamwork. The idea here is to use the power of deep learning in the domain of software engineering.

For the experimental purpose, we used the freely available dataset from the UCI data repository for student learning effectiveness in software engineering teamwork [1]. Data consist of hundred independent features and one dependent feature. We predicted the dependent feature using the proposed method. Some of the previous works used the Random Forest classifier to predict student team performance [1]. In the proposed work, we used the Feed-forward deep neural network classifier and ensemble technique. Deep neural networks can produce high accuracy with complex data very quickly. This made them a perfect choice for our work. We also, compute the feature importance and used features that are most important only. The development process of the proposed method consisted of three phases: (1) data processing, (2) designing the prediction method, training the proposed method, tuning the proposed method, and testing the proposed method, and (3) recording the results.

All the stated objectives are successfully achieved. The objective of the proposed Software Engineering Teamwork Assessment and Prediction method was to develop an effective deep-learning-based method for assessment and early prediction of student learning effectiveness in software engineering teamwork. Proposed method able to produce better prediction accuracy by using feature importance. As an increase the usability is one of the objectives, we integrated the proposed method with GUI.

To compare and evaluate the performance of the proposed prediction method with others (RF, GLM, GBM), we used accuracy percentage, confusion matrix, sensitivity, and specificity as performance measuring benchmarks. With the help of the confusion matrix, we can see that the proposed method outperformed the other three state-of-the-art machine learning algorithms which are GLM, GBM,

and RF. With all performance measures such as accuracy, sensitivity, and specificity, the proposed performed better than the other.

In future work, we expect to enhance our method further and collect more comprehensive data from multiple institutions and multiple projects. This prediction tool will be extended to automate the data collection process by allowing teams to insert their weekly progress. This tool will provide recommendations for the team that failed to improve their performance.

## ACKNOWLEDGMENT

The authors gratefully acknowledge the financial assistance from the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia.

## REFERENCES:

- [1] Petkovic, D.; Sosnick-Perez, M.; Okada, K.; Todtenhoefer, R.; Huang, S.; Miglani, N.; Vigil, A. Using the Random Forest Classifier to Assess and Predict Student Learning of Software Engineering Teamwork. *2016 IEEE Front. Educ. Conf.* **2016**, doi:10.1109/FIE.2016.7757406.
- [2] Volkov, A.; Volkov, M. Teamwork benefits in tertiary education: Student perceptions that lead to best practice assessment design. *Educ. + Train.* **2015**, *57*, doi:10.1108/ET-02-2013-0025.
- [3] Gil, B.; Gil, C.; Pérez, P.; Miguela, J. Analysing Teamwork in Higher Education: An Empirical Study on the Antecedents and Consequences of Team Conhesiveness. *Congr. Mark. AEMARK* **2017**.
- [4] Mello, R.M. de; Travassos, G.H. Surveys in Software Engineering: Identifying Representative Samples. *ESEM '16 Proc. 10th ACM/IEEE Int. Symp. Empir. Softw. Eng. Meas.* **2016**.
- [5] Ghazi, A.N.; Petersen, K.; Reddy, S.S.V.R.; Nekkanti, H. Survey Research in Software Engineering: Problems and Strategies. *Cornell Univ. Libr.* **2017**.
- [6] Krasner, H. *The Cost of Poor Quality Software in the US: A 2018 Report*; 2018;
- [7] Ghazi, A.N.; Petersen, K.; Reddy, S.S.V.R.; Nekkanti, H. Survey Research in Software Engineering: Problems and Mitigation Strategies. *24703 - 24718* **2018**, *7*, 24703–24718.
- [8] Suma, V.; Pushphavathi, T.P.; Ramaswamy, V. An Approach to Predict Software Project Success Based on Random Forest Classifier. *ICT Crit. Infrastruct. Proc. 48th Annu. Conv.*



- Comput. Soc. India* **2014**, 2, 329–336.
- [9] Janssen, N.E. A Machine Learning Proposal for Predicting the Success Rate of IT-Projects Based on Project Metrics Before Initiation. *Open-access Artic. under terms Creat. Commons Attrib.* **2019**.
- [10] Obulesu, O.; Mahendra, M.; ThrilokReddy, M. Machine Learning Techniques and Tools: A Survey. *2018 Int. Conf. Inven. Res. Comput. Appl.* **2018**.
- [11] Aljuaid, L.Z.; Wei, K.T.; Sharif, K.Y. Machine Learning: Tasks, Modern Day Applications and Challenges. *Int. J. Adv. Sci. Technol.* **2019**, 28.
- [12] Babu, A.; Srinivasan, S. A Brief Survey on Random Forest Ensembles in Classification Model. *Int. Conf. Innov. Comput. Commun.* **2018**, 56, 253–260.
- [13] Wang, H.; Xiong, J.; Yao, Z.; Lin, M.; Ren, J. Research Survey on Support Vector Machine. *MOBIMEDIA'17 Proc. 10th EAI Int. Conf. Mob. Multimed. Commun.* **2017**, 95–103.
- [14] Kaviani, P.; Dhotre, S. Short Survey on Naive Bayes Algorithm. *Int. J. Adv. Res. Comput. Sci. Manag.* **2018**, 4.
- [15] Lamba, A.; Kumar, D. Survey on KNN and Its Variants. *Int. J. Adv. Res. Comput. Commun. Eng.* **2016**, 5.
- [16] Abiodunab, O.I.; Abiodun, A.J.; OmolaracKemi, E.; Nachaat, V.D.; Elatif, A.; Arshadf, M.H. State-of-the-art in artificial neural network applications: A survey. *Heylion* **2018**, 4.
- [17] Monika; Sangwan, O.P. Software effort estimation using machine learning techniques. *2017 7th Int. Conf. Cloud Comput. Data Sci. Eng. - Conflu.* **2017**.
- [18] Pospieszny, P.; Czarnacka-Chrobot, B.; Kobylinski, A. An effective approach for software project effort and duration estimation with machine learning algorithms. *J. Syst. Softw.* **2018**, 137, 184–196.
- [19] Rijwani, P.; SonalJain Enhanced Software Effort Estimation Using Multi Layered Feed Forward Artificial Neural Network Technique. *Procedia Comput. Sci.* **2016**, 89, 307–312.
- [20] Washizaki, H.; Honda, K.; Fukazawa, Y. Predicting Release Time for Open Source Software Based on the Generalized Software Reliability Model. *Agil. Conf.* **2015**.
- [21] Han, W.; Jiang, L.; Lu, T.; Zhang, X. Comparison of Machine Learning Algorithms for Software Project Time Prediction. *Int. J. Multimed. Ubiquitous Eng.* **2015**, 10, 1–8.
- [22] Immaculate, S.D.; Begam, M.F.; Floramary, M. Software Bug Prediction Using Supervised Machine Learning Algorithms. *2019 Int. Conf. Data Sci. Commun.* **2019**.
- [23] Tayyab, M.R.; Muhammad, M.U.; Ahmad, W. A Machine Learning Based Model for Software Cost Estimation. *Proc. SAI Intell. Syst. Conf.* **2018**.
- [24] Mayr, A.; Binder, H.; Gefeller, O.; Schmid, M. The Evolution of Boosting Algorithms. Cornell University Library **2014**.
- [25] Zhu, M. When is the majority-vote classifier beneficial? **2013**, 1–17.
- [26] Dzeroski, S.; Zenko, B. Is Combining Classifiers with Stacking Better than Selecting the Best One? *Machine Learning* **2004**, 54, 255–273.
- [27] Hastie; Trevor; Tibshirani Generalized Additive Models. New York Chapman Hall **1990**.
- [28] Friedman Stochastic Gradient Boosting. Stanford Univ. **1990**.
- [29] Hatcher, W.G.; Yu, W. A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends. *IEEE Access* **2018**, 6, 24411–24432.
- [30] Schmidhuber, J. Deep Learning in neural networks: An overview. *Neural Networks* **2015**, 61, 85–117, doi:10.1016/j.neunet.2014.09.003.
- [31] M, I.P.; Srikanth, G.U. Survey of Sentiment Analysis Using Deep Learning Techniques. 2019 1st International Conference Innovation Information Communication Technology **2019**.
- [32] Shanthamallu, U.S.; Spanias, A.; Tepedelenlioglu, C.; Stanley, M. A brief survey of machine learning methods and their sensor and IoT applications. In Proceedings of the 2017 8th International Conference on Information, Intelligence, Systems Applications (IISA); 2017; pp. 1–8.
- [33] A. I. Karoly, P. Galambos, J. Kuti, and I. J. Rudas. Deep Learning in Robotics: Survey on Model Structures and Training Strategies. *IEEE Trans. Syst. Man, Cybern. Syst.*, pp. 1–14, 2020, doi: 10.1109/tsmc.2020.3018325.
- [34] Mater, A.C.; Coote, M.L. Deep Learning in Chemistry. *Journal Chemistry Information Model* **2019**, 59.
- [35] Carleo, G.; Cirac, I.; Cranmer, K.; Daudet, L.; Schuld, M.; Tishby, N.; Vogt-Maranto, L.; Zdeborová, L. Machine learning and the physical sciences. Cornell University Library **2019**.
- [36] P. V. Kulkarni, R. Phatak, B. Bhate, R. Deshpande, and S. Rai. Recommendation System for Enhancing eLearning using Deep Learning. *2019 IEEE Pune Sect. Int. Conf. PuneCon 2019*, pp. 8–11, 2019, doi: 10.1109/PuneCon46936.2019.9105685.
- [37] Suma, S.; Mehmood, R.; Albeshri, A. Automatic Event Detection in Smart Cities

- Using Big Data Analytics. Autom. Event Detect. Smart Cities Using Big Data Anal. Mehmood R., Bhaduri B., Katib I., Chlamtac I. Smart Soc. Infrastructure, Technology Application SCITA 2017. Lecter Notes of the Institue Computer Science S **2018**, 224, 111–122, doi:doi.org/10.1007/978-3-319-94180-6\_13.
- [38] Alam, F.; Mehmood, R.; Katib, I. Comparison of Decision Trees and Deep Learning for Object Classification in Autonomous Driving. In Smart Infrastructure and Applications: Foundations for Smarter Cities and Societies; Mehmood, R., See, S., Katib, I., Chlamtac, I., Eds.; Springer International Publishing: Cham, 2020; pp. 135–158 ISBN 978-3-030-13705-2.
- [39] Basu, T.; Bhatia, A.; Joseph, D.; La, P.R. A Survey on the Role of Artificial Intelligence in Software Engineering. International Journal Innovation Research Computer Communication Engineering **2017**, 5.
- [40] Li, X.; Jiang, H.; Ren, Z.; Li, G.; Zhang, J. Deep Learning in Software Engineering. Cornell Univ. Libr. **2018**.
- [41] Meinke, K.; Bennaceur, A. Machine Learning for Software Engineering: Models, Methods, and Applications. Proc. 40th International Conference Software Engineering Companion Proceedings **2018**.
- [42] Muniasamy, A.; Alasiry, A. Deep Learning: The Impact on Future eLearning. International Journal Emergineering Technology Learning **2015**, 15.
- [43] Mehmood, R.; Alam, F.; Albogami, N.N.; Katib, I.; Albeshri, A.; Altowajjri, S.M. UTiLearn: A Personalised Ubiquitous Teaching and Learning System for Smart Societies. IEEE Access **2017**, 3536, 1–22, doi:10.1109/ACCESS.2017.2668840.
- [44] Faddouli, A.C.N. El Deep learning for a smart e-learning system. 2018 4th International Conference Cloud Computing Technology Application **2018**, doi:10.1109/CloudTech.2018.8713335.
- [45] Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. University Montr ´ eal **2010**.
- [46] Candel, A.; LeDell, E.; Parmar, V.; Arora, A. Deep Learning With H2O. H2O.ai Inc **2018**.
- [47] Alam, F.; Mehmood, R.; Katib, I.; Altowajjri, S.; Albeshri, A. TAAWUN: A Decision Fusion and Feature Specific Road Detection Approach for Connected Autonomous Vehicles. Mobile Networks Application **2019**, doi:10.1007/s11036-019-01319-2.
- [48] Miaoa, J.; Niu, L. A Survey on Feature Selection. Procedia Computer Science **2016**, 91, 919–926.
- [49] Balogun, A.O.; Basri, S.; Abdulkadir, S.J.; Hashim, A.S. Performance Analysis of Feature Selection Methods in Software Defect Prediction: A Search Method Approach. Application Science **2019**, 9.
- [50] Chen, X.; Shen, Y.; Cui, Z.; Ju, X. Applying Feature Selection to Software Defect Prediction Using Multi-objective Optimization. 2017 IEEE 41st Annual Computer Software Application Conference **2017**.
- [51] Kale, S.; Karnin, Z.; Liang, T.; Pál, D. Adaptive Feature Selection: Computationally Efficient Online Sparse Linear Regression under RIP. Cornell University Library **2017**.
- [52] Du, W.; Phlypo, R.; Adali, T. Adaptive Feature Selection and Feature Fusion for Semi-supervised Classification. Journal of Signal Processing System **2019**, 91.
- [53] Cui, L.; Ma, B. Adaptive feature selection for kinship verification. 2017 IEEE International Conference Multimedia Expo **2017**.
- [54] Darshan, S.L.S.; C.D.Jaidhar Performance Evaluation of Filter-based Feature Selection Techniques in Classifying Portable Executable Files. Procedia Computer Science **2018**.
- [55] Panthong, R.; Srivihok, A. Wrapper Feature Subset Selection for Dimension Reduction Based on Ensemble Learning Algorithm. Procedia Computer Science **2015**, 72, 162–169.
- [56] Bashir, S.; Khan, Z.S.; Hassan Khan, F.; Anjum, A.; Bashir, K. Improving Heart Disease Prediction Using Feature Selection Approaches. In Proceedings of the 2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST); 2019; pp. 619–623.
- [57] Yang, R.; Zhang, C.; Zhang, L.; Gao, R. A Two-Step Feature Selection Method to Predict Cancerlectins by Multiview Features and Synthetic Minority Oversampling Technique. Biomed Research International **2018**, 2018, 10.
- [58] Candel, A.; Lanford, J.; LeDell, E.; Parmar, V.; Arora, A. Deep Learning with H2O Deep Learning with H2O. In: H2O.ai, Inc, 2015.
- [59] Romanski, P.; Kotthoff, L.; Kotthoff, M.L. Package FSelector: Selecting Attributes. CRAN **2016**, 18.
- [60] Tuszynski, J. Package ‘caTools.’ CRAN **2020**.
- [61] Kuhn, M.; Wing, J.; Weston, S.; Williams, A.; Keefer, C.; Engelhardt, A.; Cooper, T.; Mayer, Z.; Brenton Kenkel, the R Core Team, Michael Benesty, R.L.; Andrew Ziem, Luca Scrucca, Yuan Tang, Can Candan, and T.H. Classification and Regression Training. CRAN **2017**.
- [62] Nalband, S.; Sundar, A.; Prince, A.A.;

- 
- Agarwal, A. Feature selection and classification methodology for the detection of knee-joint disorders. *Comput. Methods Programs Biomed.* **2016**, *127*, 94–104, doi:10.1016/j.cmpb.2016.01.020.
- [63] Wickham, H. *ggplot2: Elegant Graphics for Data Analysis*; Springer-Verlag New York, 20015; ISBN 978-0-387-98140-6.
- [64] UCI UC Irvine Machine Learning Repository Available online: <https://archive.ics.uci.edu/ml/index.php>.
- [65] Sokolova, M.; Lapalme, G. A systematic analysis of performance measures for classification tasks. *Information Processing Management* **2009**, *45*, 427–437, doi:10.1016/j.ipm.2009.03.002.