*International Journal on Recent and Innovation Trends in Computing and Communication*
***ISSN: 2321-8169 Volume: 11 Issue: 10***
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

# A Framework to Automate Requirements Specification Task

**Pratvina Talele**
Department of Computer Engineering and Technology,
Dr. Vishwanath Karad MIT World Peace University,
Pune, India
pratvina.talele@mitwpu.edu.in

**Rashmi Phalnikar**
Department of Computer Engineering and Technology,
Dr. Vishwanath Karad MIT World Peace University,
Pune, India
rashmi.phalnikar@mitwpu.edu.in

**Abstract**— Requirement identification and prioritisation are two principal activities of the requirement engineering process in the Software Development Life cycle. There are several approaches to prioritization of requirements identified by the stakeholders. However, there is a need for a deeper understanding of the optimal approach. Much study has been done and machine learning has proven to help automate requirement engineering tasks. A framework that identifies the types of requirements and assigns the priority to requirements does not exist. This study examines the behaviour of the different machine learning algorithms used for software requirements identification and prioritisation. Due to variations in research methodologies and datasets, the results of various studies are inherently contradictory. A framework that identifies the types of requirements and assigns the priority to requirements does not exist. This paper further discusses a framework for text preparation of requirements stated in natural language, type identification and requirements prioritisation has been proposed and implemented. After analysing the ML algorithms that are now in use, it can be concluded that it is necessary to take into account the various types of requirements when dealing with the identification and classification of requirements. A Multiple Correlation Coefficient-based Decision Tree (MCC-based DT) algorithm considers multiple features to map to a requirement and hence overcomes the limitations of the existing machine learning algorithms. The results demonstrated that the MCC-based DT algorithm has enhanced type identification performance compared to existing ML methods. The MCC-based DT algorithm is 4.42% more accurate than the Decision Tree algorithm. This study also tries to determine an optimisation algorithm that is likely to prioritise software requirements and further evaluate the performance. The sparse matrix produced for the text dataset indicates that Adam optimisation method must be modified to assign the requirement a more precise priority. To address the limitations of the Adam Algorithm, the Automated Requirement Prioritisation Technique, an innovative algorithm, is implemented in this work. Testing the ARPT on 43 projects reveals that the mean squared error is reduced to 1.34 and the error cost is reduced to 0.0001. The results indicate an 84% improvement in the prioritisation of requirements compared to the Adam algorithm.

**Keywords**- Requirements Classification, Requirements Prioritisation, Optimisation Algorithm, Machine Learning

## I. Introduction

Understanding the customer requirements, analysing them, determining feasibility, checking for the practical solution, clearly specifying the needs and a solution, validating the documents and managing the requirements as they are transformed into a working system are all covered by Requirement Engineering (RE). Requirement engineering is the systematic use of well-established ideas, methodologies, tools, and notation to specify a proposed system's expected behaviour and restrictions [1]. This phase mainly elicits, specifies, validates and manages requirements activities to be followed sequentially. The perspective to interpret the user's requirement changes from developer to developer due to the ambiguous nature of the requirements. This creates several flaws that are considered during the software development process. For this purpose, identifying types of software requirements specified in a natural language plays a vital role in the success or failure of a software product [2]. Classification of software requirements should be done systematically, so that developers can make decisions about the sequence of software attributes to be implemented. However, there are many stakeholders with different requirements involved in the development process of a software product. For this reason, the software requirements are to be automatically classified into functional and non-functional requirements.

Recent research work has shown that these requirements described in software requirements specification (SRS) can be classified automatically using Natural Language Processing (NLP) techniques and Machine Learning (ML) algorithms [3]. The first step of the classification of requirements stated in a natural language is the text preparation and the second step is to apply classification ML algorithms. The text preparation involves two stages, that is, text preprocessing and feature extraction. Text preprocessing involves text segmentation that describes the frontiers for a word by removing stop words, punctuations and white spaces and converting the text into tokens [4]. Text preprocessing also involves text normalization, which uses stemming or lemmatization steps. Feature extraction can be done by converting words into vectors as a machine circuitry that only understands 0s and 1s [5]. Different classification ML algorithms include Decision Tree, Support Vector Machine, Neural Network, Naïve Bayes, Logistic Regression, K-Nearest Neighbor and Random Forest. Decision Tree and Support Vector Machine are the most popular algorithms for classifying software requirements [6]. Decision tree uses criteria such as Information Gain, Gini Index, chi-square and gain ratio to split an attribute [7]. Using these criteria, the attribute will be selected to split on that has a significant number of dissimilar values. For example, a dataset has an attribute with a maximum number of unique values. As this attribute will be considered to split, the number of levels in the tree will be increased. This increases the biasness.

Because of this reason, these criteria are unable to provide satisfactory accuracy for some datasets where the dependency should be considered. To overcome this drawback, a new method is proposed in [8] that uses the concept of multiple correlation coefficient as a splitting criterion. This approach considers two attributes for classification mapped to a single target value.

There are a few constraints during software development, such as cost and time, due to which different software versions can be released in succession. Hence, it is important to decide which requirements should be considered in the first release of the software. Requirement Prioritisation (RP) is the process where the stakeholders prioritise the requirements [9]. The prioritisation of requirements to be implemented in successive software releases is determined by stakeholders.

Requirements Prioritisation has proven incredibly difficult, with scalability being one of the most significant challenges [10] [11]. The number of stakeholders in the case of large-scale projects is enormous. These stakeholders may have different requirements, resulting in disagreements over which criteria should be prioritised. The major problem facing today's companies is meeting stakeholder's requirements and raising potential expectations in a timely, secure, cost-effective and relevant manner. Due to budget constraints and production time constraints, it may be difficult for requirements analysts to decide which requirements should be prioritised, resulting in excellent user satisfaction. Inaccuracy in prioritising requirements can lead to the software being rejected by end-users due to a lack of standards and the product being declared a failure sooner or later. To increase the cost advantages and fulfill the timeframes of software, high-priority requirements must be considered first, followed by low-priority requirements. Methods that prioritise requirements are becoming increasingly necessary. Most research is done in this area, but finding the correct technique or framework at the right time takes much work. Since the beginning of software, that is, the requirement phase, it has been indicated to implement security features [10].

There are many requirements prioritisation algorithms used by stakeholders like Analytic Hierarchy Process (AHP), Must have, Should have, Could have, Won't have (MoScoW), Bubble sort, Binary Search Tree, Hundred Dollar and Priority group [12]. Recently researchers have proposed different prioritisation techniques such as Adaptive Fuzzy Hierarchical Cumulative Voting [13], Gradient Descent Rank [14], Apriori [15] and DRank [16]. The study by Talele and Phalnikar [17] shows that existing prioritisation techniques have limitations in terms of complexity and scalability.

This work makes three primary contributions. Initially, it employs a method for classifying software requirements by type. Secondly, it utilizes an automated procedure to prioritize these requirements. Thirdly, the study assesses the effectiveness of the proposed method by comparing existing machine learning algorithms in terms of their accuracy and mean squared error for identifying requirement types and priorities. The paper is structured as follows: Section II presents the literature survey, Section III outlines the framework, Section IV presents the results, and Section V concludes the study and suggests avenues for future research.

## II. Literature Survey

Requirement Engineering is an important Software Development Life Cycle (SDLC) step. Significant research and empirical studies have been conducted on the first two phases of requirement engineering: requirements classification [18] and requirements prioritization [19].

Several attempts have been made to automate the process of identifying requirements [18] [20] and designating requirements' priorities using machine learning algorithms [14] and various prioritisation techniques [19]. Natural Language Processing (NLP) techniques prepare the text, which is the initial phase in identifying requirement types. The second stage uses ML algorithms to determine the types of requirements. Prioritisation strategies can be used to assign priority values to requirements once they have been categorised.

### *A. Text Preparation*

A text-based requirement document is provided as input for the text preparation step. Various NLP techniques are used to pre-process the text of these documents. The most popular NLP pre-processing techniques include stop word removal, part of speech, tokenization, N-grams, lemmatization and stemming. When the input is text, tokenization is an important step. Tokenization breaks down a sentence into tokens [19] [21] [22]. The Stemming phase reduces terms to their stem, base, or root form. For instance, provided and providing have the same root word, "provide" [19] [22] [23]. The Part of Speech (POS) system allocates tags to nouns, verbs, etc. [24] [16]. This technique eliminates conjunctions, auxiliary verbs and articles from sentences. For instance, a, an, the, and, be, etc. It is the most prevalent method. [19] [22] [23] [24] [25]. Lemmatization identifies the lemma word, the infinitive form of verbs, and the singular noun and adjective forms for each term. For instance, saves, 'saved' and 'saving' correspond to 'save'. This method is employed in [23] [24].

The model can be trained using features as inputs. This phase converts a previously processed requirements document into features. Feature selection techniques include Term Frequency-Inverse Document Frequency (TF-IDF) and Bag of Words (BoW), which are two-word frequency analysis techniques. The BoW technique calculates the frequency of each word in a sentence or document [23] [24]. The Term Frequency-Inverse Document Frequency (TF-IDF) technique is used to calculate the frequency with which a word appears in a document or sentence by calculating the inverse of the number of times the word appears in the corpus [19] [23] [26].

### *B. Identification of Type of Requirements Methods*

Various studies have classified software requirements using machine learning algorithms. Following feature selection, the model's learning phase begins. Supervised, unsupervised, and semi-supervised learning techniques are employed to identify Functional Requirements (FR) and Non- Functional Requirements (NFR).

Supervised learning algorithms require a set of requirements that have been correctly identified into FRs and NFRs. The number of requirements used to train datasets varies in the selected studies. 58 requirements are used to train its data, focusing on security NFRs [19]. In this area of research, numerous supervised learning algorithms have been applied, including Support Vector Machine (SVM) [24] [26] [27], Naïve Bayes (NB) [23] [26] [28], Adaptive Boost and Random Forest [24], Decision Tree (DT) [22] [23] [24] [26] [29] and K-Nearest Neighbour (KNN) [26] [27].

Researchers have also applied unsupervised learning algorithms such as LDA [28] [29] [30] [31], Hierarchical Agglomerative Clustering algorithm [28] [32], singlelink clustering algorithm [25], K-means [28] [32] and Bi-term [29]. The accuracy performance of these algorithms could be enhanced [20].

Self-training, RAndom Subspace Method for Co-training (RASCO), Relevant RAndom Subspace Method for Co-training (Rel-RASCO) [26] and active learning [27] are semi-supervised learning algorithms used in this research domain. RASCO, Rel-RASCO and Self-training algorithms are combined with supervised learning algorithms such as KNN, DT, NB and SVM to identify the types of requirements. When classifying requirements using these algorithms, not all NFR categories are evaluated. Only three SRS are considered as a dataset [27] and app store user evaluations are used as a dataset [26]. Using these algorithms, labelling training instances requires less human effort. But recall levels ranged between 54.6% and 81.9%, precision levels ranged between 73.9% and 92.4% on average for distinct categories of NFRs from three projects [27], and transductive and inductive accuracy levels were below 70% [26].

Numerous additional ML algorithms, such as Stochastic Gradient Descent (SGD) [25] and Goal Oriented Requirements Elicitation (GORE) [33] [34] are also utilised. CNN is also utilised to categorise the requirements [25]. Applying CNN and SGD to various documents to test the learned model yields inferior results [25].

### C. *Requirements Prioritisation Methods*

Research and empirical studies are conducted in the domain of requirements prioritisation [19]. There are two types of approaches to prioritising requirements. The first set of fundamental methods includes priority classification, numerical assignment, Cost-Value approach, Analytic Hierarchy Process (AHP) and cumulative voting, while the second group incorporates the fundamental methods. It is challenging to determine if the second group has been validated and applied in practice. Applying a specific set of approaches to similar criteria that have been pretty well studied and are at the same level of abstraction has been a challenging step in research.

Prioritising software requirements, AHP is a methodical statistical technique utilized [14] [16] [33] [35] in the software community that focuses on relative evaluation. When qualitative and quantitative factors must be considered in the decision-making process, the AHP proves to be a resilient and flexible method that aids individuals in establishing priorities and selecting the optimal option. AHP facilitates the selection of the most advantageous option by simplifying difficult decisions into a sequence of direct comparisons. AHP facilitates the synthesis of results, thereby offering a justification for selecting prospective requirements. Based on n requirements, n (n – 1) / 2 comparisons must be performed at each hierarchy level throughout the procedure. This is the disadvantage of this procedure, as the number of comparisons increases by a factor of O(n2) as the number of requirements increases.

100-Dollar Test or Cumulative Voting (CV) is a simple method in which each stakeholder is given 100 fictitious units. This method utilises the ratio scale. The sophistication and granularity of this prioritisation method are both high. These fictitious units could represent various aspects (for example

, hours, penalty, importance, implementation cost, etc.). Adaptive Fuzzy Hierarchical Cumulative Voting (AFHCV) [13] is a novel prioritisation technique that is built using the foundational algorithm Fuzzy Hierarchical Cumulatives (FHCV). AFHCV is of the opinion that requirements are subject to change throughout the software development process. As a result, modifications to requirements are incorporated at runtime. These requirements are reassessed and new priorities are assigned. The system's efficiency decreases with each iteration of the analysis and prioritisation of requirements, particularly with regard to time complexity and scalability.

Similar prioritisation techniques employ machine learning algorithms such as Gradient Descent Rank (GDRank) [14] and apriori [15]. Case Base Rank (CBRank) [14] algorithm is evaluated to compare these new prioritisation techniques. The MoSCoW (Must have, Should have, Could have, and Won't) method is an approach to prioritisation utilised to ascertain which requirements are essential, desirable, feasible, and not worthy of consideration. The fuzzy-based MoSCoW method [34] utilises the fundamental MoSCoW prioritisation algorithm. This method is not evaluated. It is necessary to validate it against an extensive range of requirements.

Continuous domain adaptation is necessary [36]. Novel approaches that were developed were not assessed on datasets encompassing diverse domains. Furthermore, the classification process relied solely on a restricted set of software requirements rather than sub-classes of non-functional requirements [37]. Prioritisation approaches require professional participation and are time-consuming and complex. To overcome such challenges, existing machine learning algorithms must be evaluated for scalability and accuracy.

## III. METHODOLOGY

The software development life cycle (SDLC) refers to the systematic approach employed in planning, modifying, and maintaining software. Software quality ensures that software products are engineered to meet the user requirements. An important aspect is developing a framework that defines the requirements. The requirements are gathered, classified and prioritised in the RE process. Based on the research gap identified, a new framework is proposed for identifying the types of requirements and prioritising requirements, as shown in Fig. 1. The function for each module –

Module 1 – The text preparation
Module 2 – Identification of types of requirements
Module 3 – Prioritising requirements

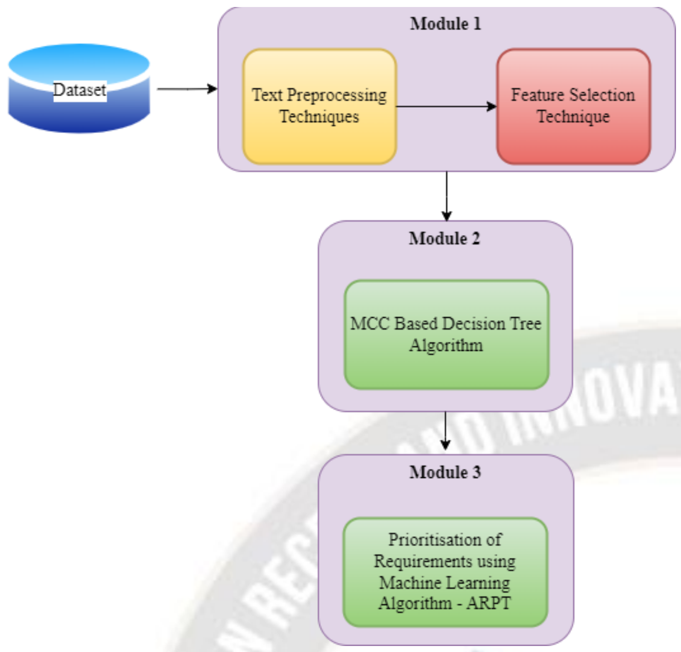

Dataset
Module 1
Text Preprocessing Techniques
Feature Selection Technique
Module 2
MCC Based Decision Tree Algorithm
Module 3
Prioritisation of Requirements using Machine Learning Algorithm - ARPT

Figure 1. A Framework

*A. The Text Preparation*

Natural Language Processing is a part of ML technologies that trains computers to process a large amount of data specified in natural language. It provides the facility to understand human language. The text preprocessing and feature extraction methods clean the raw text and provide the features as input to the ML algorithms. It will be easy to train models using supervised learning algorithms after the raw text is preprocessed using natural language processing methods. Based on the literature survey, text preprocessing methods such as lowercase, removal of white space, removal of stop words, tokenization, and lemmatization are used, and the Term Frequency-Inverse Document Frequency (TF-IDF) method is used to extract the features [38] as shown in Fig. 2.

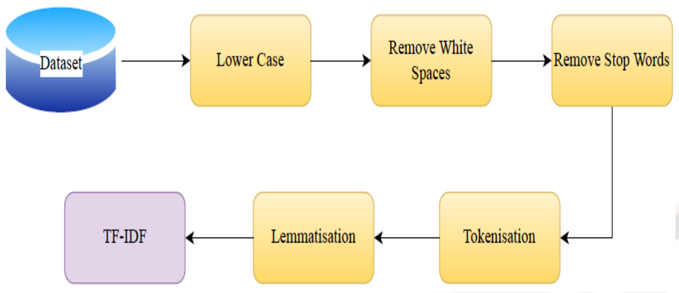


Figure 2. Text Preparation Module

*B. Identification of Types of Requirements*

As the requirements are stated in a natural language, the extracted features are words. The relationship between these words must be considered during the type identification of requirements. Multiple Correlation Coefficient (MCC) [39] considers the relationship between two features and a class. To consider more features at a time to split on and should be mapped to a class, MCC based decision tree algorithm is used in this study [8] as shown in Fig. 3.

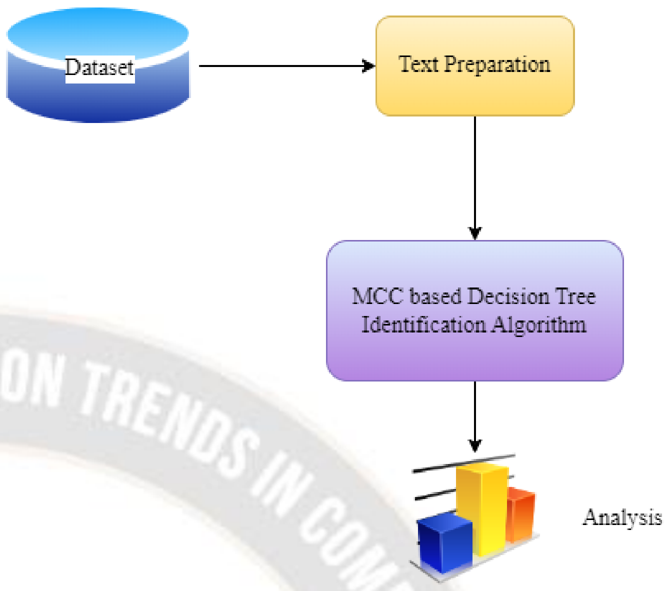


Figure 3. MCC based Decision Tree Algorithm

*C. Prioritising Requirements*

Prioritising software requirements is an essential phase in requirement engineering. Many researchers have presented various approaches for prioritising requirements. Prioritising requirements is an iterative process that identifies the most significant requirements for effective software or system deployment.

Based on the literature study, various ML algorithms, such as GDRank and CBRank, are presented for prioritising requirements. Existing optimization algorithms such as stochastic gradient descent (SGD), mini-batch SGD, SGD with Momentum, and Root Mean Squared Propagation (RMSProp) [40] [41] [42] are used to compare with updated Adaptive Movement Estimation (Adam), which finds the priority for requirements.

To find the priority, the Automatic Requirement Prioritisation Technique (ARPT) uses the Updated Adaptive Movement Estimation (Updated Adam SGD) optimisation algorithm [43] as shown in Fig. 4, which is compared to the SGD, mini-batch SGD, SGD with Momentum, and RMSProp algorithms.

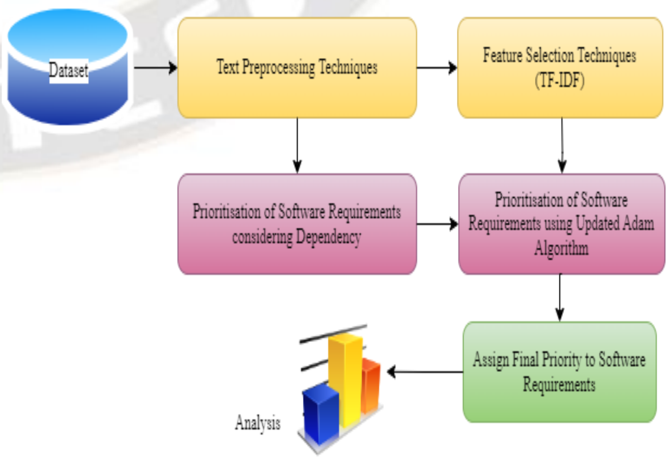

Dataset
Text Preprocessing Techniques
Feature Selection Techniques (TF-IDF)
Prioritisation of Software Requirements considering Dependency
Prioritisation of Software Requirements using Updated Adam Algorithm
Assign Final Priority to Software Requirements
Analysis

Figure 4. ARPT

## IV. RESULTS

To execute the proposed approach, we employed the user-friendly Python 3.8.5 programming language and tkinter, which offers support for various dataset file formats, including CSV. Additionally, we opted for Scikit-learn due to its array of modules facilitating the creation of machine learning classifiers and computation of evaluation metrics. The computational infrastructure consisted of a DELL Laptop featuring an Intel(R) Core(TM) i5-10300H processor clocked at 2.50GHz, 8 GB of RAM, and a 64-bit Windows operating system.

The dataset underwent a split, dividing it into two parts using an 8:2 ratio. Specifically, 80% of the dataset was allocated for training the model, while the remaining 20% originating from the same domain was designated for model testing purposes. This study compares the existing machine learning algorithms to identify the type of software requirements in terms of accuracy, as shown in Table 1 and Fig. 5.

TABLE I. COMPARISON OF ALGORITHMS USED TO IDENTIFY THE TYPES OF REQUIREMENTS

| Algorithm | 561 | 701 | 912 | 1670 | 2153 |
|---|---|---|---|---|---|
| Decision Tree | 95.24 | 96.43 | 79.49 | 95.06 | 91.21 |
| Random Forest | 95.24 | 92.86 | 84.62 | 91.97 | 87.91 |
| Multinomial Naive Bayes | 95.24 | 89.29 | 79.49 | 89.51 | 87.36 |
| Logistic Regression | 95.24 | 89.29 | 79.49 | 90.12 | 87.36 |
| Neural Network | 95.24 | 92.86 | 82.05 | 88.27 | 78.57 |
| KNN | 90.48 | 100 | 82.05 | 90.74 | 88.46 |
| SVM | 95.24 | 92.86 | 82.05 | 90.12 | 89.56 |
| MCC based Decision Tree | 95.24 | 96.43 | 94.87 | 96.3 | 96.7 |

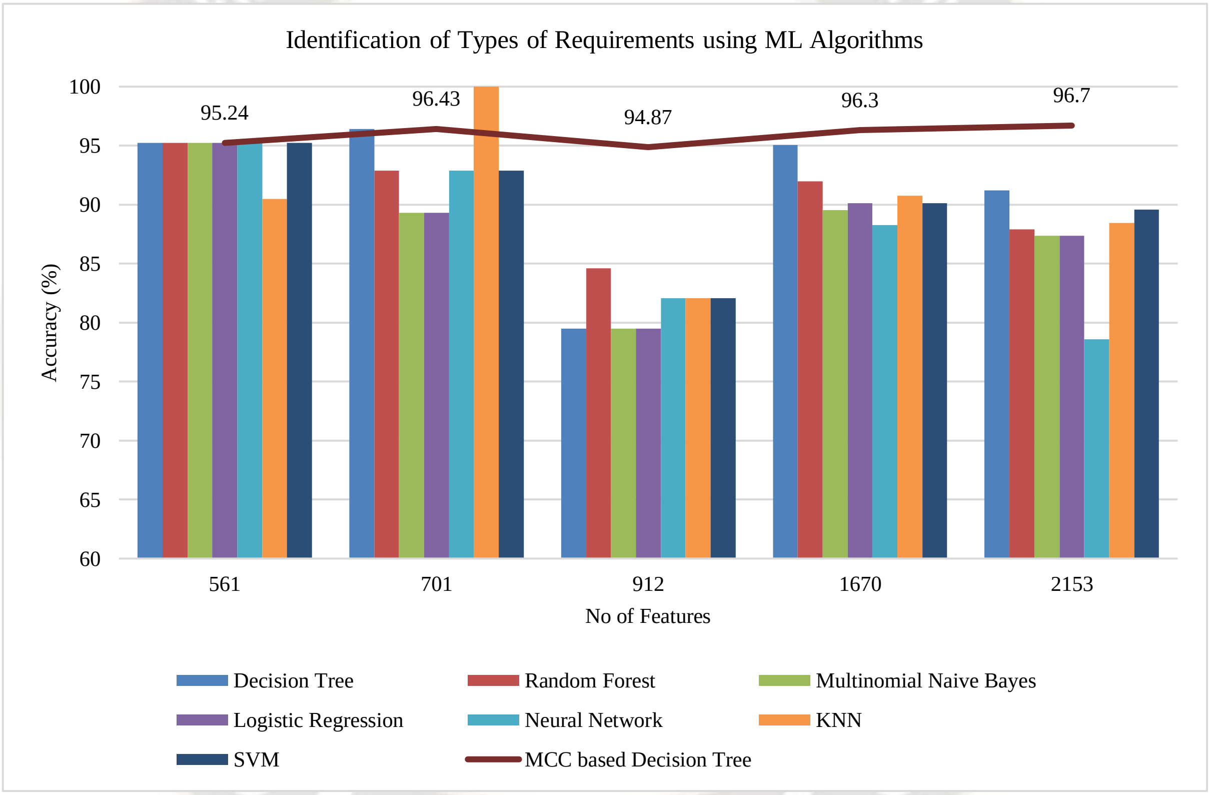


Figure 5. Comparison of Identification of Types of Requirements using ML Algorithms

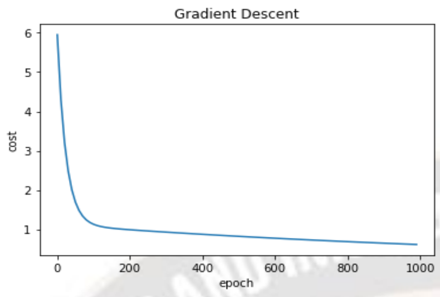
From Table 1 and Fig. 5 it is observed that MCC based Decision Tree consistently outperforms the other existing machine learning algorithms even if the number of features are changed.

Stakeholders frame the same type of requirement using different terms. Because of the high level of variance in requirements elicitation, automated classification or type identification is more prone to errors. Therefore, the challenge is to determine the best classification algorithm. Furthermore, such classification is required since manually identifying types of software requirements takes time, particularly for large projects with a significant number of requirements in the industry [28].

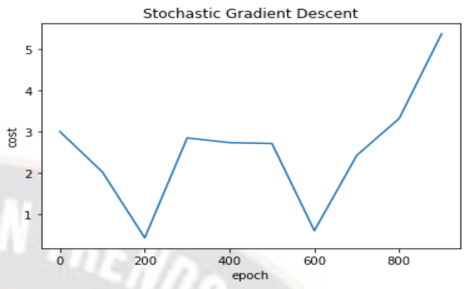
A comparison of existing optimization algorithms to assign priority to software requirements Gradient Descent (GD), Stochastic Gradient Descent (SGD), Mini-batch SGD, SGD with momentum, RMSProp, Adam with the updated Adam algorithm is shown in Fig. 6 and 7 in terms of error cost vs epochs and mean squared error respectively.

It is observed that error cost reduces as the number of epochs increases in the case of GD, Mini-batch SGD, SGD with momentum, RMSProp, Updated Adam as shown in Fig. 6 (a, c, d, e, g). At the commencement of the iteration, the error cost for GD is 5.94, while for SGD it is 3. The distorted waveform is formed in the case of the Adam algorithm, as the huge sparse data is created for the text requirements. Hence, the Adam algorithm necessitates an extended duration to mitigate the error cost. The convergence rate of the ARPT is greater than that of the Adam algorithm.

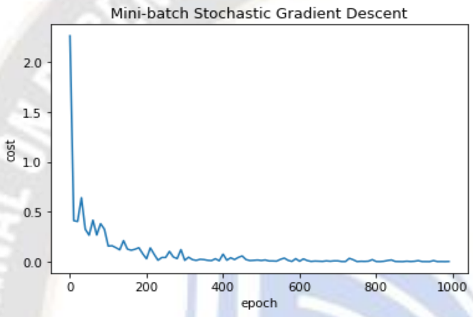
The results presented in Figure 6 indicate that the mean squared error, which represents the discrepancy between the predicted and original priorities, is significantly smaller when utilising Updated Adam compared to other optimisation algorithms currently in use. Adam was ultimately outperformed by 84.44% by ARPT. The proposed method can be used in industry to identify the type of requirements and assign priority to requirements automatically, so manual efforts and time can be reduced, as well as errors that occurred during the type identification of requirements and finding the priority of requirements manually.



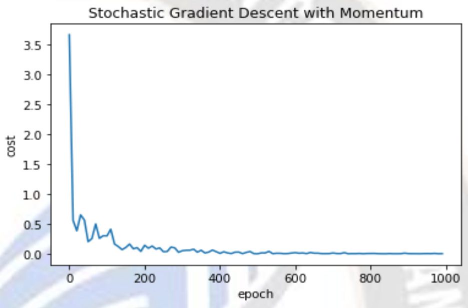
(a)



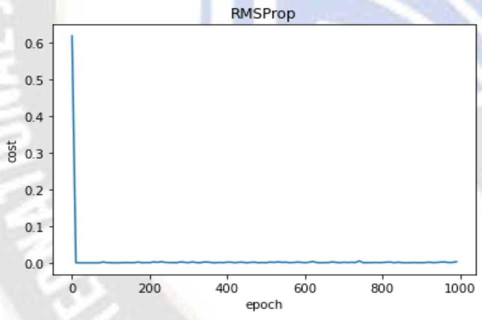
(b)



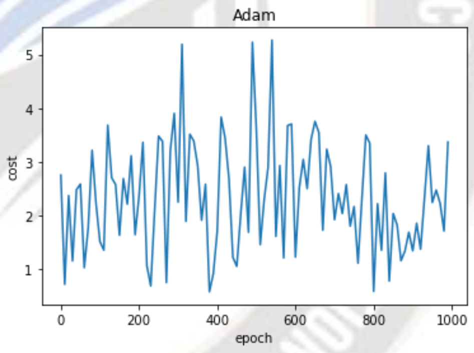
(c)



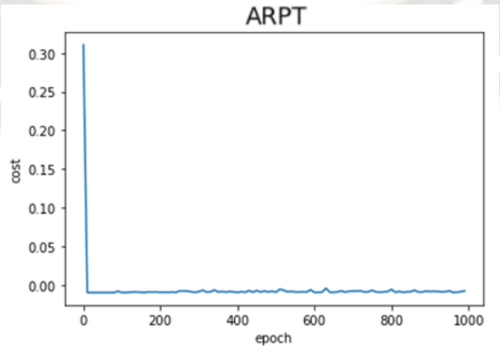
(d)



(e)



(f)



(g)

Figure 6. Training cost of Optimization Algorithms

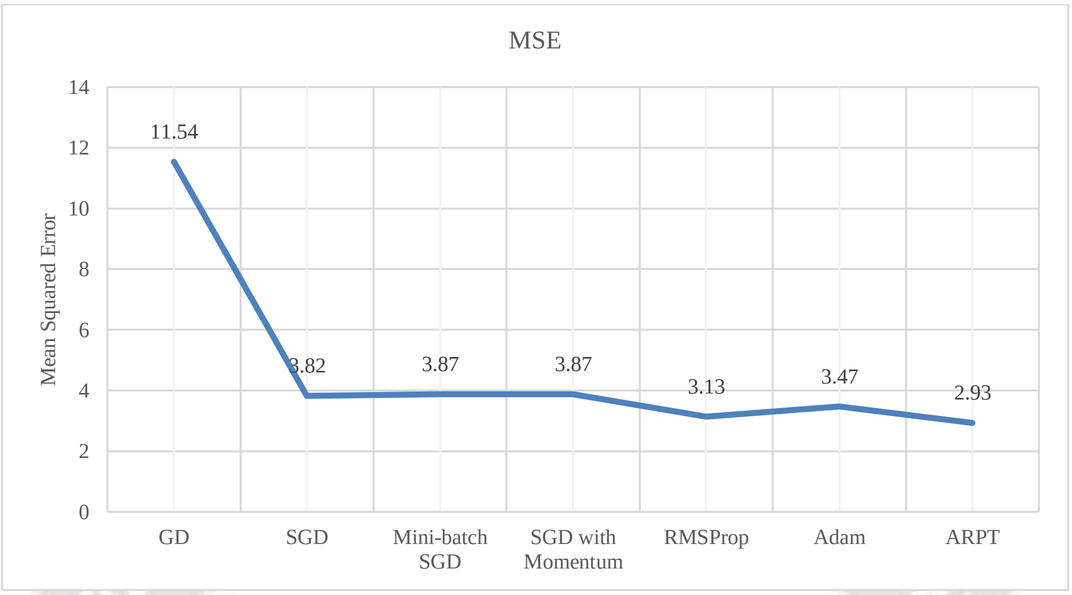


Figure 7. Comparison of Optimization Algorithms

## V. Conclusion and Future Scope

This study discusses the framework for identification of types of requirements and prioritisation of requirements. This framework can bridge the gap between the research and practice.

The experimental findings so conducted showed that the Multiple Correlation Coefficient based decision tree algorithm increased the type identification performance, outperformed the decision tree algorithm by 4.42%, and achieved 96.7% accuracy for identifying the requirements. The Adam Algorithm exhibits a lack of convergence in the context of text features, specifically when the feature frequency within the dataset is excessively low. In order to circumvent this constraint, the Automated Requirement Prioritisation Technique is suggested as a means of prioritising software requirements. One notable benefit of the Automated Requirement Prioritisation Technique is its superior performance on text datasets compared to Adam. In terms of error cost and mean squared error, this study also systematically contrasts different optimisation methods with the proposed method for prioritising requirements. On 43 projects dataset containing functional and non-functional requirements for software projects, including usability, performance, and so forth, the Automated Requirement Prioritisation Technique is evaluated. The experimental findings revealed that the Automated Requirement Prioritisation Technique exhibited superior performance compared to the Adam algorithms. At epoch 1000, requirement prioritisation was executed with a mean square error of 2.93 and a cost error of 0.0001. One drawback of the proposed algorithm is its failure to consider the interdependencies among requirements. This study proves that MCC based decision tree and ARPT algorithms performs better than the existing methods used to identify the types of requirements and assign the priorities to requirements respectively.

Future evaluations of the proposed method will utilise a wide range of datasets from various domains in order to circumvent this limitation. Future work will also involve designing and developing an improved algorithm to consider requirement dependencies and conflicts.

## References

[1] J. Dick, E. Hull and K. Jackson, "Introduction," in Requirements Engineering, Springer Cham, 2017.

[2] A. Khelifa, M. Haoues and A. Sellami, "Towards a Software Requirements Change Classification using Support Vector Machine," in CEUR Workshop Proceedings, 2018.

[3] L. V. Rooijen, F. S. Bäumer, M. C. Platenius, M. Geierhos, H. Hamann and G. Engels, "From user demand to software service: using machine learning to automate the requirements specification process," in IEEE 25Th international requirements engineering conference workshops (REW), 2017.

[4] V. Fong, "Software requirements classification using word embeddings and convolutional neural networks," California Polytechnic State University, 2018.

[5] K. A. Memon and X. Xiaoling, "Deciphering and analyzing software requirements employing the techniques of natural language processing," in 4th International Conference on Mathematics and Artificial Intelligence, 2019.

[6] P. Talele and R. Phalnikar, "Software requirements classification and prioritisation using machine learning," in Machine Learning for Predictive Analysis: Proceedings of ICTIS 2020, 2021.

[7] B. Lantz, "Divide and Conquer-Classification Using Decision Trees and Rules," Machine Learning with R, 2015.

[8] P. Talele and R. Phalnikar, "Multiple correlation based decision tree model for classification of software requirements," International Journal of Computational Science and Engineering, vol. 26, no. 3, pp. 305-315, 2023.

[9] B. NUSEIBEH and S. EASTERBROOK, "Requirements engineering: a roadmap," in The Future of Software Engineering, Ireland, 2000.

[10] M. Batra and A. Bhatnagar, "Requirements Prioritization: A Review," International Journal of Advanced Research in Science, Engineering and Technology, vol. 3, no. 11, pp. 2899-2904, 2016.

[11] R. Devadas and N. G. Cholli, "Multi aspects based requirements prioritization for large scale software using deep neural lagrange multiplier," in International Conference on Smart Technologies and Systems for Next Generation Computing (ICSTSN), 2022.

[12] A. Hudaib, R. Masadeh, M. H. Qasem and A. Alzaqebah, "Requirements prioritization techniques comparison," Modern Applied Science, vol. 12, no. 2, 2018.

[13] B. B. Jawale, G. K. Patnaik and A. T. Bhole, "Requirement prioritization using adaptive fuzzy hierarchical cumulative voting," in IEEE 7th International Advance Computing Conference (IACC), 2017.

[14] D. Singh and A. Sharma, "Software Requirement Prioritization using Machine Learning," in Software Engineering and Knowledge Engineering, SEKE, 2014.

[15] R. V. Anand and M. Dinakaran, "Handling stakeholder conflict by agile requirement prioritization using Apriori technique," Computers & Electrical Engineering, vol. 61, pp. 126--136, 2017.

[16] F. Shao, R. Peng, H. Lai and B. Wang, "DRank: A semi-automated requirements prioritization method based on preferences and dependencies," Journal of Systems and Software, vol. 126, pp. 141--156, 2017.

[17] P. Talele and R. Phalnikar, "Classification and prioritisation of software requirements using machine learning--A systematic review," in 11th International Conference on Cloud Computing, Data Science \& Engineering (Confluence), 2021.

[18] T. Iqbal, P. Elahidoost and L. Lucio, "A bird's eye view on requirements engineering and machine learning," in 25th Asia-Pacific Software Engineering Conference (APSEC), 2018.

[19] F. Hujainah, R. B. A. Bakar, M. A. Abdulgabber and K. Z. Zamli, "Software requirements prioritisation: a systematic literature review on significance, stakeholders, techniques and challenges," IEEE Access, vol. 6, pp. 71497--71523, 2018.

[20] M. Binkhonain and L. Zhao, "A review of machine learning algorithms for identification and classification of non-functional requirements," Expert Systems with Applications: X, vol. 1, p. 100001, 2019.

[21] A. Khan, B. Baharudin, L. H. Lee and K. Khan, "A review of machine learning algorithms for text-documents classification," Journal of advances in information technology, vol. 1, no. 1, pp. 4-20, 2010.

[22] R. Malhotra, A. Chug, A. Hayrapetian and R. Raje, "Analyzing and evaluating security features in software requirements," in International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH), 2016.

[23] M. Lu and P. Liang, "Automatic classification of non-functional requirements from augmented app user reviews," in 21st International Conference on Evaluation and Assessment in Software Engineering, 2017.

[24] Z. Kurtanović and W. Maalej, "Automatically classifying functional and non-functional requirements using supervised machine learning," in 25th International Requirements Engineering Conference (RE), 2017.

[25] J. Winkler and A. Vogelsang, "Automatic classification of requirements based on convolutional neural networks," in 24th International Requirements Engineering Conference Workshops (REW), 2016.

[26] R. Deocadez, R. Harrison and D. Rodriguez, "Automatically classifying requirements from app stores: A preliminary study," in 25th international requirements engineering conference workshops (REW), 2017.

[27] C. Li, L. Huang, J. Ge, B. Luo and V. Ng, "Automatically classifying user requests in crowdsourcing requirements engineering," Journal of Systems and Software, vol. 138, pp. 108-123, 2018.

[28] Z. S. H. Abad, O. Karras, P. Ghazi, M. Glinz, G. Ruhe and K. Schneider, "What works better? a study of classifying requirements," in 25th International Requirements Engineering Conference (RE), 2017.

[29] R. Jindal, R. Malhotra and A. Jain, "Automated classification of security requirements," in International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2016.

[30] J. Zou, L. Xu, M. Yang, X. Zhang and D. Yang, "Towards comprehending the non-functional requirements through developers' eyes: An exploration of stack overflow using topic analysis," Information and Software Technology, vol. 84, pp. 19-32, 2017.

[31] I. Morales-Ramirez, D. Munante, F. Kifetew, A. Perini, A. Susi and A. Siena, "Exploiting user feedback in tool-supported multi-criteria requirements prioritization," in 25th International Requirements Engineering Conference (RE), 2017.

[32] A. Mahmoud and G. Williams, "Detecting, classifying, and tracing non-functional software requirements," Requirements Engineering, vol. 21, pp. 357-381, 2016.

[33] M. Sadiq, T. Hassan and S. Nazneen, "AHP_GORE_PSR: Applying analytic hierarchy process in goal oriented requirements elicitation method for the prioritization of software requirements," in 3rd International Conference on Computational Intelligence & Communication Technology (CICT), 2017.

[34] K. S. Ahmad, N. Ahmad, H. Tahir and S. Khan, "Fuzzy_MoSCoW: A fuzzy based MoSCoW method for the prioritization of software requirements," in International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), 2017.

[35] L. Alawneh, "Requirements prioritization using hierarchical dependencies," in Information Technology-New Generations: 14th International Conference on Information Technolog, 2018.

[36] F. Dalpiaz, D. Davide, A. F. Basak and Ç. Sercan, "Requirements classification with interpretable machine learning and dependency parsing," in 27th International Requirements Engineering Conference (RE), 2019.

[37] C. S. R. K. Surana, D. B. Gupta and S. P. Shankar, "Intelligent chatbot for requirements elicitation and classification," in 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT), 2019.

[38] P. Talele, S. Apte, R. Phalnikar and H. Talele, "Semi-automated Software Requirements Categorisation using Machine Learning Algorithms," International Journal of Electrical and Computer Engineering Systems, vol. 14, no. 10, pp. 1107--1114, 2023.

[39] [Online]. Available: https://real-statistics.com/correlation/multiple-correlation/.

[40] "PURE dataset," [Online]. Available: http://nlreqdataset.isti.cnr.it/.

[41] "RMSProp," [Online]. Available: https://www.coursera.org/lecture/deep-neural-network/rmsprop-BhJlm.

[42] P. Talele and R. Phalnikar, "Machine learning-based software requirements identification for a large number of features," International Journal of Computational Systems Engineering, vol. 6, no. 6, pp. 255--260, 2021.

[43] P. Talele and R. Phalnikar, "Automated Requirement Prioritisation Technique Using an Updated Adam Optimisation Algorithm," International Journal of Intelligent Systems and Applications in Engineering, vol. 11, no. 3, pp. 1211-1221, 2023.

[44] P. Talele and R. Phalnikar. [Online]. Available: https://ieee-dataport.org/documents/requirements-classification-and-prioritisation.

[45] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in International Conference on Learning Representations, 2014.

[46] [Online]. Available: https://www.opendoorerp.com/the-standish-group-report-83-9-of-it-projects-partially-or-completely-fail/