International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048
Volume 10, Issue 5, pp 316-327, May-2026
https://doi.org/10.47001/IRJIET/2026.105042

A Multiclass Classification of Software Security Requirements: A Deep Learning Approach

Landry Giraud Wandji T.

Ph.D. Student at the Hochschulinstitut Schaffhausen, Switzerland
E-mail: landry.wandjitchatchoua@edu.hochschule-schaffhausen.ch

***Abstract* - One of the first step in the software development process is the requirements analysis. This task plays a relevant role because of his considerable impact on the developed end product. Since the security aspect of software systems are becoming much more critical this recants years, the prioritization of software security requirements in the early stage of the software development process is necessary. For that reason, the development of techniques applicable on the analysis of software security requirements are representing great advancements for the software requirements engineering activities. The literature review on methods to analyze and classify software requirements shows that, there are existing research on machine learning techniques to classify software requirements, but the most of works are working on binary classification of functional and non-functional requirements, so the multiclass classification of software security requirements are not in the focus of the most research activities. This work proposes an approach for classification of software security requirements in multiclass, using classical machine learning techniques and deep learning techniques, filling the gap on the missing activities in this area. Another contribution of this approach it the evaluation of machine learning models on multiclass classification of software security requirements. For this work, a Multinomial Naive Bayes model (MNB), an Artificial Neural Network model (ANN), a Convolutional Neural Network model (CNN) and a Recurrent Neural Network model (RNN) are trained and tested on DOSSPRE dataset. The results show that the Multinomial Naive Bayes model achieved the best performance with 72% accuracy, the worst performer was the Recurrent Neural Network model with 62% accuracy.**

Keywords: Requirement engineering, Deep learning, Machine learning.

I. INTRODUCTION

The agile development of software products requires a prioritization of software requirement, for that reason the analysis of software requirements is playing a central role and is one of the first steps in the SDLC. [1] Consider the requirement analysis phase as a major step as the customer requirements are analyzed and documented in a clear natural language. In this phase several important aspects of the final product are clearly expressed and reviewed. Since the security aspect of software system is becoming further more critical this last year, security requirements are winning importance. [15] Mentioned that the source of security problems is the non-consideration of security requirements of a complete system, but the security in the application itself. For that reason, the prioritization of security requirements in the early stage of the software development process shall be considered.

Security requirements are a subcategory of non-functional requirement (NFR) that can be generated based on established standards or industrial application, which outline the required security characteristics of a software in other to be save from potential attackers [3] - [11]. They can be also obtained through user stories, evaluations of software or documentations supplied by experts or stakeholders involved in system development, where the software will be incorporated. Software security requirements can also be defined by identifying threats and vulnerabilities that describe possible actions of hackers to violate the security integrity of the developed system.

Software security requirements can be binary classified between Security Requirements (SR) and Non-Security Requirements (NSR). [5] Published a works where the binary classification of security requirements has been investigated using different machine learning techniques, presenting software security requirements in two different categories. This shows that machine learning techniques can be applied for software requirements classification on a very successful way. There are different types of methods using Natural Language Processing (NLP) to analyze and classify software requirements. The performances and classification results using classical machine learning or deep learning techniques can be different, depending on the amount of data available and the type of analysis task to be performed.

For a deeper analysis of software security requirements, their classification in multiclass will be necessary. Sub-categories of software security requirements reported by [9]

ISSN (online): 2581-3048
Volume 10, Issue 5, pp 316-327, May-2026
https://doi.org/10.47001/IRJIET/2026.105042

are Availability (AVA), Authentication (THE), Authorization (THO), Immunity (IMM), Integrity (INT), Intrusion detection (IND), confidentiality (CON), Auditing (AUD), Survivability (SUR), Maintainability (MAI). This gives the possibility for the software development process to consider the specificity of each security aspect of the software and to develop the corresponding solution required. For that reason, the classification of software security requirements in sub-categories will improve the activities of requirements engineers.

This work presents a multiclass classification of software security requirements using classical machine learning techniques and deep learning techniques. The DOSSPRE dataset generated by Kadebu et al. [9] will be used for this work. The first part of this work will be a literature review of existing works on this topic presented in session 2. The second part of this work will be the description of the methodology of this approach in session 3. In this part, the different steps to be implemented and the expected results are described. The results of this approach are presented in the session 4. Finaly, this works is summarized in the last session 5.

II. RELATED WORK

Due to the fact that the neglection of security requirements in the early stage of the software development life cycle can lead to poor end product quality, there are a lot of approaches presenting the analysis and classification of software requirements, but just few of those publications are focusing on the multiclass classification of software security requirements. As example, [7] analyze and describes security requirements found in a system requirements specification document in other to develop classification models. Text mining techniques were used to analyze the security-based descriptions, which are then categorized into four types of security requirements using the J48 decision tree method: data integrity, cryptography-encryption, access control, and authentication-authorization. A prediction model has been created for each kind of security requirement. The result analysis indicated that all the four models have performed very well in predicting their respective type of security requirements. But this works does not consider machine learning as classification method and do not classify all the sub-category of software security requirements.

An approach regarding assessment of software security reequipments from the perspective of completeness, contradiction and consistency has been proposed by [12]. Security standards introduced by the ISO have used to construct a model for classifying security-based requirements using natural language processing based machine learning techniques. This approach gives software organizations guidelines for creating thorough and clear requirements specification documents pertaining to security-oriented features. The classification of software security requirements in different under categories and the performance evaluation of different machine learning techniques using software requirements dataset was not in scope of this approach.

[13] Provide a method for automatically classify software requirements using machine learning, to represent text data from software requirements specifications and categorize requirements into functional and non-functional groups. The PROMISE_exp dataset, which has labeled requirements, was the experimented dataset used in this study. All of the database's software documents were cleaned using a normalization, extractions, and selection of applicable techniques. SVM and KNN algorithms were used in this study to classify the requirements. With an average F-measure of 0.74 across all cases, it has been observed that using BoW with SVM is better than using KNN algorithms. But this work does not consider the classification of software security requirements.

With the introduction of a comprehensive dataset that included 9529 functional requirements from 315 different projects, [17] used five categories to group software requirements: ubiquitous, unwanted behavior, state-driven, event-driven, and optional features. Automated classification investigated by the use of machine learning (ML), deep learning (DL), and natural language processing (NLP) techniques. Every software requirement underwent a number of processes, including feature extraction methods like TF-IDF and normalization. To categorize functional requirement subcategories, a number of machine learning (ML) and deep learning (DL) experiments were carried out. The multiclass classification of software security requirements was not a target of this work.

The classification of software requirements into functional and non-functional requirement has been presented by [18] using the RE'17 dataset challenge. They used the dataset to determine how word embedding affects NFR and FR classification in comparison to conventional features (like bag-of-words). Additionally, they wanted to determine whether using a sophisticated neural classifier is required to achieve the best NFR and FR classification performance. The application of FastText appears to be the most promising classification model based on the results obtained. Therefore, this work does not focus on the classification of software security requirements.

A deep learning framework-based automated non-exclusive method for classifying functional requirements is proposed by [8]. In particular, a convolutional neural network

ISSN (online): 2581-3048
Volume 10, Issue 5, pp 316-327, May-2026
https://doi.org/10.47001/IRJIET/2026.105042

(CNN) is trained using Word2Vec and FastText word embeddings for document representation. In order to conduct this study, pertinent enterprise data that had been manually categorized was compiled and used for model training. The effects of data trained using Word2Vec and FastText word embeddings from SRS documentation were compared to pre-trained word embeddings models that are accessible online. But the consideration of multiclass classification of software security requirements was not part of this method.

In [4] software security requirements have been divided into "Identification, Authentication, Authorization, Immunity, Integrity, Intrusion Detection, Non-Repudiation, Privacy, Security Auditing, Survivability, Physical Protection, and System Maintenance Security Requirements". This classification is quite detailed and presents System Maintenance as a Security Requirement. Nevertheless, this classification approach did not take into account the use of dataset to support the classification process or the application of machine learning techniques. As a result, numerous studies have categorized software requirements using various methods, such as supervised learning and natural language processing.

An efficient method for classifying software requirements into multiclass using machine learning techniques has been developed by [2]. In order to categorize non-functional requirements (NFRs) into the following five categories: maintainability, operability, performance, security, and usability, they specifically look into the design and use of two types of neural network models: an Artificial Neural Network (ANN) and a Convolutional Neural Network (CNN). Two popular datasets with almost 1,000 NFRs to demonstrate and experimentally assess this work were taken as reference. Therefore, the subcategories of security requirements were not classified with this method, since the focus on software security requirement was not a target of this study.

There are diverse machine leaning model applied for software requirements classification task. [6] notice that NB, KNN, and SVM are mostly utilized in the requirements analysis and classification phase, whereas in the requirements validation phase SVM, DT, and NB are mostly utilized. in the other hand, DT and SVM algorithms are commonly applied during the requirement specification stage. They draw the conclusion that the requirement analysis and classification phase offers a variety of supervised learning algorithms, in contrast to the requirements specification and validation phases.

An approach applying machine learning techniques to classify NFR according to product, process, and external factors was presented by [16]. The authors created four machine learning based algorithms: Logistic Regression (LR), Support Vector Machine (SVM), Decision Tree (DT) and K Nearest Neighbor Classifier (KNN), as well as one deep learning technique called Long Short-Term Memory (LSTM) model in order to classify the NFR. After analyzing the models, the authors come to the conclusion that the LSTM model, which achieved an accuracy of 99.6%, is suitable for this kind of classification.

[14] Presents supervised learning models learning from labeled data, where input features are already associated with the correct outputs. This group can further be separated between classification algorithms used for binary classification and regression algorithms used to predict continuous values, such as prices or temperatures. In this group, there are algorithms such as Linear Regression, Logistic Regression, Decision Tree (DT), Random Forest (RF), Support Vector Machines (SVM), Multi-Layer Perceptron (MLP), Artificial Neural Networks (ANN), K-Nearest Neighbors (KNN) and Gradient Boosting (XGBoost, LightGBM, CatBoost).

Deep learning methods include algorithms that learn complex data representations using multi-layered artificial neural networks. Particular applications for these algorithms include speech and audio processing, natural language processing, and image recognition. Algorithms like ANN, Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) belong to this category. This category also includes pre-trained Language Models (PLM) algorithms like Generative Pre-trained Transformer (GPT) and Bidirectional Encoder Representations from Transformers (BERT).

The relevant research for this study is summarized in the Table 1. The most of study are comparing the application of different machine learning algorithms and deep learning algorithms for requirements engineering activities. Other research works develop methods for effective software requirements classification and finally, systematics literature review was conducted in other to determine what are the most applied machine learning techniques for each requirement engineering activity.

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 10, Issue 5, pp 316-327, May-2026
https://doi.org/10.47001/IRJIET/2026.105042

Table 1: Summary of the relevant literature

Title	Task	Method	Dataset
Jane Cleland-Huang et al.[3]	Development of a classification model	• analyzed using text mining techniques • classified into four types of security requirements • J48 decision tree method	Not available online.
Ruchika Malhorta et al.[12]	Methodology for semi-automatic classification model	• Construction of a model • Use NLP • Use ML techniques	Not available online.
Gaith Y Quba et al. [13]	Comparison of text feature extraction techniques and machine learning techniques	• Comparison of BoW vs TF-IDF VS CHI^2 • Use Logist Regression (LR) • Use Support Vector Machine (SVM) • Use Multinomial Naïve Bayes (MNB)	PROMISE_exp
Touseef Tahir et al. [17]	Automated software requirements classification	• Generate datasets • Use ML techniques • Use DL techniques	EARS
Donal Firesmith et al. [4]	defines the different types of security requirements enabling to specify security requirements	• Providing associated examples • provides guidelines	No data sets
Sanjanasri JP et al. [8]	Automated non-exclusive approach for classification of functional requirements	• Using deep learning framework, CNN • Using Word embedding (Word2Vec and FastText)	AUTOSAR
Cody baker et al. [2]	leverage ML techniques to develop an effective approach to classify software requirements	• Using deep learning model (ANN, CNN)	PROMISE
Shoaib Hassan et al. [6]	investigate the research trends, mainRE activities, ML algorithms, and data sources	• Systematic mapping • investigate studies focused on ML in RE activities • Period of 2002 -2023	-
MAF Saroth et al. [16]	Approach for categorizing NFR based on product, process, and external factors	• Develop four ML algorithms(LR, SVM, DT, KNN) • Develop a DL algorithm (LSTM)	No defined
Rosado da Cruz et al. [14]	conduct a systematic review on AI, ML and NLP for requirements engineering	• Literature review • Identify algorithms ML and DL algorithms for RE	-
Sabrina Tiun et al. [18]	find out the effect of word embedding against traditional features (such as bag-of-words) in NFR and FR classification	• Investigate effect of embedding • Investigate effect of bag-of-words	RE`17

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 10, Issue 5, pp 316-327, May-2026
https://doi.org/10.47001/IRJIET/2026.105042

The literature review shows that there are existing approaches describing the classification of software requirements into functional and non-functional requirements, but not focusing on software security requirements. Other studying considering the classification of software security requirement are just considering security requirement as a subclass of non-functional requirements. So only very few of the research publications are focusing on the multiclass classification of software security requirements. On the other hand, classical machine learning techniques are used for software requirements classification in the most of relevant works found in the literature review, but their performance on the multiclass classification of software security requirements is missing.

In other to contribute to fill this gap, this work proposes an approach to investigate the application of machine learning techniques for multiclass classification of software security requirements. The methodology of this approach will contribute to enhance the prioritization of software security requirements in the early stage of the software development process.

III. METHODOLOGY

In this section the different steps to develop and apply machine learning algorithms to classify software security requirements in multiclass are presented. Since the target of this work is to fill the gap on the classification of software security requirements in subcategory using machine learning and deep learning models, the performance of different algorithms will be compared on DOSSPRE dataset. The Figure 1 presents the steps of this approach.

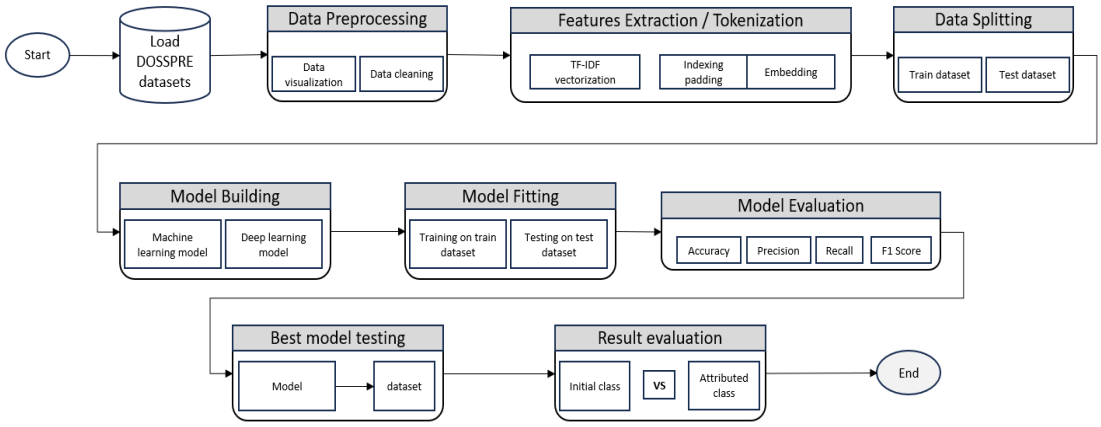

Figure 1: Description of the methodology for this approach

The DOSSPRE dataset will be preprocess then the feature extraction will prepare the data for the next steps. The data will be split and used by the built model in other to generate the necessary measurements metrics to evaluate the classification models.

3.1 Data preprocessing

The first step of this approach is the data preprocessing. Using an adequate environment for software development using python programming language, the software security requirements dataset will be loaded and visualized. This will give an idea about the structure of the data, the number of columns and raw. It will be also necessary to check, if there are different datatype and duplicated data. The data visualization also gives the possibility to plot the class distribution inside the dataset. After the visualization it is important to clean the data by removing unnecessary symbols inside the sentences. This will facilitate the extraction of the features inside the data.

3.2 Feature extraction

The second step of this approach is the feature extraction from the data. In this step, the sentences are tokenized since machine learning algorithms are only able to read digital input. The tokenization consists of breaking up sentences input into tokens, which are smaller units. These tokens may take the shape of sentences, words, letters, or sub-words. This sentences transformation will help in making text easier for machine leaning models to be interpreted. There are different possibilities to perform the tokenization, in this case we are going to use TF-IDF vectorizer on the one hand and indexing, padding and embedding on the other hand.

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 10, Issue 5, pp 316-327, May-2026
https://doi.org/10.47001/IRJIET/2026.105042

3.3 Data splitting

In the third step of this work the data will be split in two groups, the train and the test dataset. The train dataset will be used in the first phase of the model fitting to make the model be able to recognize the pattern of the different classes and to memorize the pattern to each corresponding class. The test dataset will be used to test the trained model and evaluate his performance, for that reason it will be necessary to prepare enough training data, more than the test data in other to give to model enough information to easily characterize each class.

3.4 Machine learning classifier

The model building and fitting will be the fourth step of this approach, a classical machine learning model and deep learning models will be built. For this study Multinomial Naive Bayes (MNB) will be investigate and his performance will be compared to those of deep learning techniques. Artificial Neural Network (ANN), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) will be implemented and tested in the first stage. In the second stage the best of all the models will be test on four requirements chosen randomly in the dataset.

3.5 Performance metrics

The last step of this work will be the evaluation of the model performances. In this case different measurement metrics will be applied. [10] Highlighted the positive effect of multiple evaluation metrics in assessing machine learning models. This metrics is based on true positive (*tp*), true negative (*tn*), false positive (*fp*) and false negative (*fn*) values.

Accuracy: The model accuracy is one of the model performance evaluation units that will be consider in this approach. The target in this case will be to reach model accuracy closer 80%. The model accuracy is the most basic criteria to assess classification algorithms. It is a gauge of how well the model predicts the class. The following presents the calculation of the accuracy:

$$Accuracy = \frac{(t_p + t_n)}{(t_p + t_n + f_p + f_n)} \quad (1)$$

Precision: It indicates how well the model avoids false positive. By dividing the number of accurate positive predictions (true positives) by the total number of occurrences the model predicted as positive (true and false positives), you may determine accuracy. The precision can be expressed as a percentage or on a scale from 0 to 1. The greater the precision, the better. When the model consistently predicts the target class correctly and never flags anything as incorrect, you can have a perfect precision of 1.0. The calculation of the precision is given by:

$$Precision = \frac{t_p}{(t_p + f_p)} \quad (2)$$

Recall: The frequency with which a machine learning model properly detects positive sample (true positives) from all of the actual positive samples in the dataset is measured by a metric called recall. The recall can be computed by dividing the number of true positive cases by the number of positive cases. Recall can also be called sensitivity or true positive rate. The following equation shows how to calculate the recall value:

$$Recall = \frac{t_p}{(t_p + f_n)} \quad (3)$$

F1-Score: It combines two other important metrics: precision and recall, into a single value. As the harmonic mean of precision and recall, the F1-Score provides a more balanced measure of a model's performance, especially on imbalanced datasets where one class occurs much more frequently than the other. In such scenarios, a high accuracy value can be misleading, but the F1 score gives a better sense of the model's effectiveness in correctly identifying the minority class. The formula to calculate the F1-score is given by:

$$F1 - Score = \frac{t_p}{t_p + \frac{(f_p + f_n)}{2}} \quad (4)$$

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 5, pp 316-327, May-2026

https://doi.org/10.47001/IRJIET/2026.105042

IV. RESULTS

By looking closer inside the DOSSPRE dataset we can observe that the data are quite imbalanced. The Figure 2 shows the classes repartition inside the datasets. The non security requirements are the most represented class with around 60% of the requirements, the rest of 40% is the security requirements subclasses. The security requirements are quite well balanced across the subclasses.

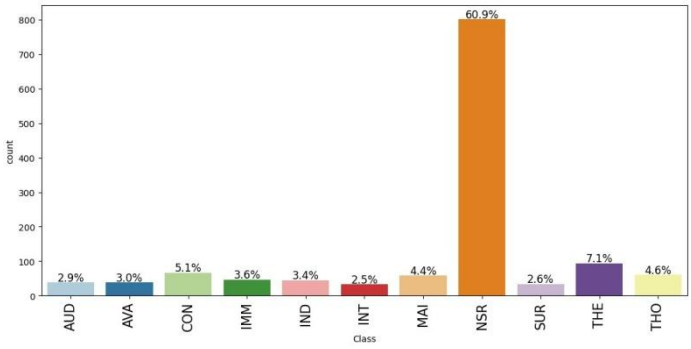

Figure 2: Repartition of the requirements toward the subclasses

The data presents eleven classes with a total of 1316 requirements. After the data cleaning the requirements need to be prepared in the corresponding type to be processed by the machine learning models. The Figure 3 shows two ways of data preparation and the outcomes of this step.

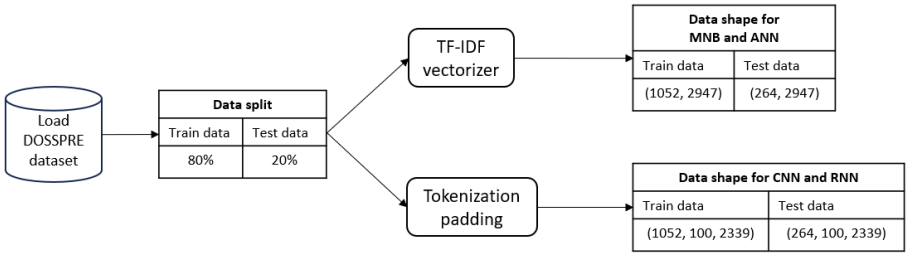

Figure 3: Shape of the split data after preprocessing

4.1 Results of the MNB model

The split data are used to train and test the built models. The Multinomial Naive Bayes model (MNB) shows a result of 71.6%. This performance can be highlighted by the confusion matrix in Figure 4, the other measurements metric precision, recall and F1-score are for the MNB model is summarized in the table 2.

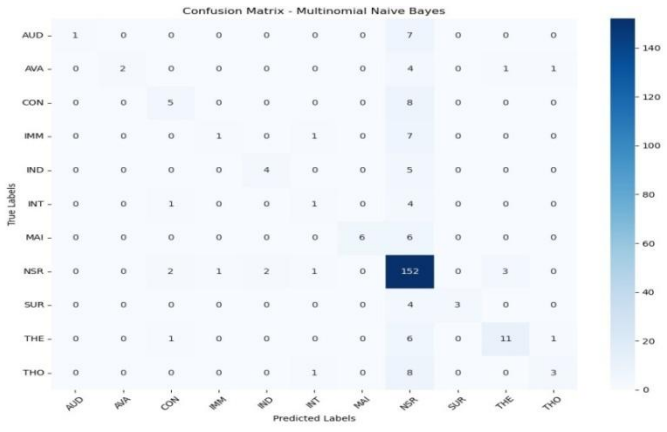

Figure 4: Confusion matrix for the MNB model

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 5, pp 316-327, May-2026

https://doi.org/10.47001/IRJIET/2026.105042

Table 2: Evaluation results for the Multinomial Naives Bayes

Metrics	Value
Accuracy	0.72 (71.59%)
Macro Precision	0.73
Macro Recall	0.38
Macro F1-Score	0.46
Weighted Precision	0.72
Weighted Recall	0.72
Weighted F1-Score	0.68

4.2 Results of the ANN model

The Artificial Neural Network model was also applied on the split dataset. The training of the model was performed with hundred epoch and twenty seven batch size, the early stopping function was also implemented in other to optimize to model training time. The results of the ANN model are shown on the Figure 5 and the table 3.

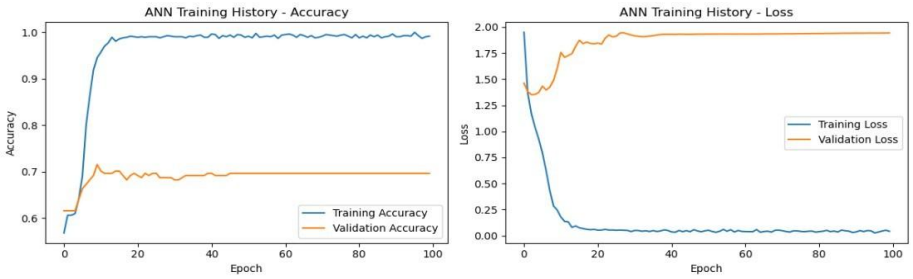

Figure 5: Training and validation results of the ANN model

Table 3: Evaluation results for the Artificial Neural Network model

Metrics	Value
Accuracy	0.71 (70.83%)
Macro Precision	0.58
Macro Recall	0.45
Macro F1-Score	0.49
Weighted Precision	0.69
Weighted Recall	0.71
Weighted F1-Score	0.68

4.3 Results of the CNN model

The Convolutional Neural Network model was built and applied on the test and train dataset. The performance of the trained CNN model was evaluated after hundred epoch, with a batch size of twenty seven. In other to control the training evolution, the early stopping function was also implemented. A summary of the results for the CNN model is presented by the Figure 6 and the table 4.

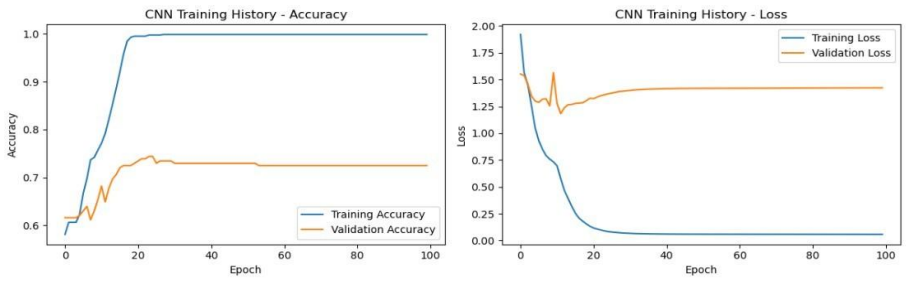

Figure 6: training and validation results of the CNN model

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 5, pp 316-327, May-2026

https://doi.org/10.47001/IRJIET/2026.105042

Table 4: Evaluation results for the Convolutional Neural Network model

Metrics	Value
Accuracy	0.69 (68.56%)
Macro Precision	0.50
Macro Recall	0.34
Macro F1-Score	0.38
Weighted Precision	0.66
Weighted Recall	0.69
Weighted F1-Score	0.65

4.4 Results of the RNN model

A Recurrent Neural Network model was also built trained and tested using the split dataset. In other to evaluate the model performances, a batch size of twenty seven was defined and an epoch of hundred. The early stopping function was also implemented to optimize the training evolution. The results for the RNN model are summarized by the Figure 7 and the table 5.

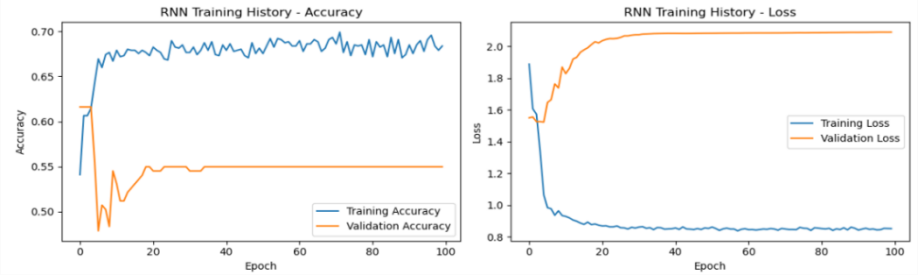

Figure 7: training and validation results of the RNN model

Table 5: Evaluation results for the Recurrent Neural Network model

Metrics	Value
Accuracy	0.62 (62.12%)
Macro Precision	0.20
Macro Recall	0.24
Macro F1-Score	0.21
Weighted Precision	0.55
Weighted Recall	0.62
Weighted F1-Score	0.58

4.5 Results summary

After the evaluation of the trained and tested models on the prepared dataset, we can observe that the Multinomial Naive Bayes model achieved the best performance in comparison to the deep learning model. This can be explained by the fact that, deep leaning model are using neural network, so they need a large and consistent amount of data in other to be well trained and get a good performance on test dataset. In this case the dataset is also quite imbalanced with the NSR class representing 60% of all the data distributed on 11 classes. The comparison of the results is summarized by the table 6 and the Figure 8.

Table 6: summary of the models performances

Model	Accuracy	Precision(Macro)	Recall(Macro)	F1-Score(Macro)	A-Score(Weighted)
MNB	0.72	0.73	0.38	0.46	0.67
ANN	0.71	0.58	0.45	0.49	0.68
CNN	0.69	0.50	0.34	0.38	0.65
RNN	0.62	0.20	0.24	0.21	0.58

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 5, pp 316-327, May-2026

https://doi.org/10.47001/IRJIET/2026.105042

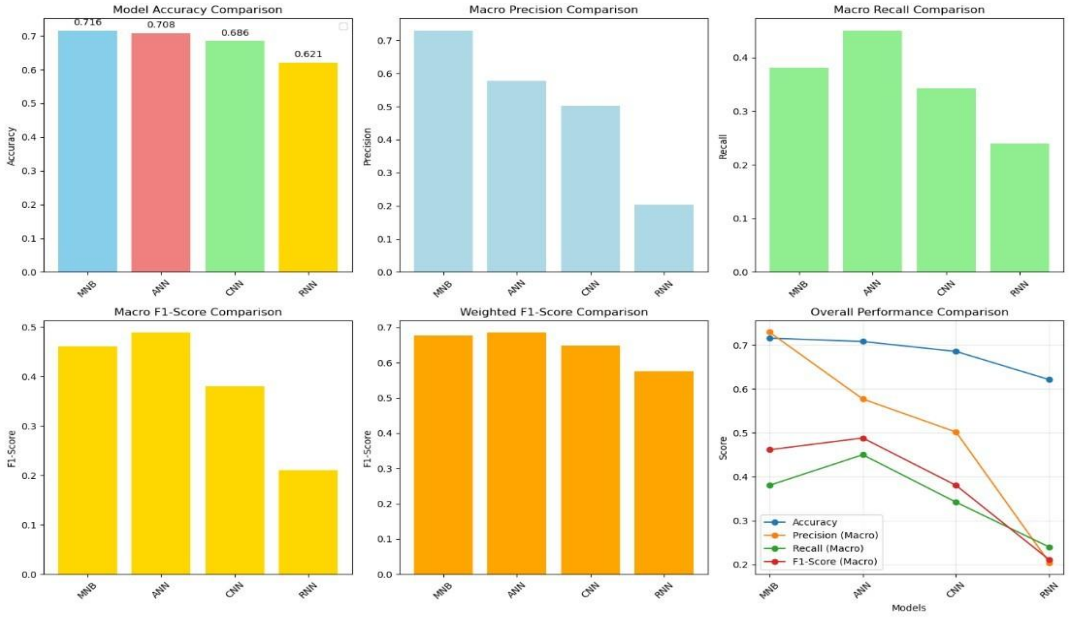

Figure 8: comparison of the models performances

From the deep learning model, the Artificial Neural Network and the Convolutional Neural Network achieved the best performance with respectively 71% and 69% accuracy. The Recurrent Neural Network achieved the worst performance with 62% accuracy. From all the models none could reach a performance higher than 72% accuracy. This shows the impact of the data quality and quantity on the machine learning models and the need to investigate other classification techniques, corresponding to this data.

The best model was tested on a couple of requirements to verify in which class the model will classify them. This will testify the effectiveness of the trained and tested model to properly realize the classification task. The table7 shows the classification results of three requirements chosen randomly in the dataset.

Table 7: Result of the classification test with the best model

Test sentence	Initial class	Classification results
The system shall enable fast transmission of information between bank server and...	NSR	Predicted: NSR (Confidence: 0.88) Top 3 predictions: • NSR: 0.88 • CON: 0.04 • THE: 0.03
The system should be maintained properly so that it should be able to resolve an...	MAI	Predicted: MAI (Confidence: 0.52) Top 3 predictions: • MAI: 0.52 • NSR: 0.21 • THE: 0.08
The system shall enable a visible watermark to be used together with the email c...	NSR	Predicted: NSR (Confidence: 0.84) Top 3 predictions: • NSR: 0.84 • THE: 0.10 • CON: 0.01

We can observe that the classification task has been performed successfully by the best machine learning model (MNB).The predicted class by the model are corresponding to the labeled class in the original dataset.

ISSN (online): 2581-3048
Volume 10, Issue 5, pp 316-327, May-2026
https://doi.org/10.47001/IRJIET/2026.105042

V. CONCLUSION

This work presented us in a detailed and structured approach the importance of software security requirements classification and different techniques enabling to implement this classification task in the early stage of the software development life cycle. The literature review on this topic shows that, there is a gap to be filled since the focus on software security requirements is pretty seldom on the one hand, and the multiclass classification of software security requirements highlighting the subclasses is rarely in the scope of the research works in the other hand. The most of work are focusing on the classification between functional and non-functional requirements.

To fill this gap, we proposed an approach using classical machine learning and deep learning techniques to classification security requirements in multiclass using the DOSSPRE dataset. In the first step of this approach, we prepared the data to be properly processed by the machine leaning models. After that, we split the data into train and test dataset. In the fourth step the machine learning model were built and applied on the prepared dataset in the training and the testing phase. In the fifth step the performances of the models were evaluated.

The results of the performance evaluation have shown that the Multinomial Naive Bayes as classical machine learning model achieved the best performance with an accuracy of 72%. Regarding the deep learning models, the Artificial Neural Network model achieved a better performance than the Convolutional Neural Network and the Recurrent Neural Network with 71% accuracy. The worst performance was realized by the Recurrent Neural Network Model with 68% accuracy. So, in summary the classical machine learning model performs better than all the deep learning models investigated in this study.

The comparison of the models results shows that different factors can impact the deep learning models performance. The first observation was, that the number of requirements (1316) was probably not enough to train the deep learning model, in so far that deep learning models are known to need a lot of data to achieve their best performance. So, a future work could be to combine existing software requirements datasets in other to generate consistent dataset and investigate the impact of data quantity on machine learning models performances. Another possibility could also be to investigate the application of data augmentation techniques on the DOSSPRE dataset.

Another observation was the imbalance repartition of the data through the different classes inside the DOSSPRE dataset. In fact, the NSR class as one of the eleven classes inside the overall requirements has around 60% of all the requirements. This was a considerable disadvantage for an efficient training of the machine learning models. For that reason, a future work could also be to explore the impact of techniques for imbalanced dataset like SMOTE on machine learning model performances.

REFERENCES

[1] Abdulbasit ALazzawi, Bahbibi Rahmatullah, *et al.* “A comprehensive review of software development life cycle methodologies: Pros, cons, and future directions”. *In: Iraqi Journal for Computer Science and Mathematics* 4.4 (2023), pp. 173–190.

[2] Cody Baker *et al*. “Automatic multi-class non-functional software requirements classification using neural networks”. *In: 2019 IEEE 43rd annual computer software and applications conference (COMPSAC).* Vol. 2. *IEEE.* 2019, pp. 610–615.

[3] Jane Cleland-Huang *et al.* “Automated classification of non-functional requirements”. *In: Requirements engineering* 12.2 (2007), pp. 103–120.

[4] Donald Firesmith *et al.* “Engineering security requirements.” *In: J. Object Technol.* 2.1 (2003), pp. 53–68.

[5] Wandji T. Landry G. “A Binary Classification of Software Security Requirements: A Deep Learning Approach”. *In: IRJIET* 09 (2025).

[6] Shoaib Hassan *et al.* “A systematic mapping to investigate the application of machine learning techniques in requirement engineering activities”. *In: CAAI Transactions on Intelligence Technology* 9.6 (2024), pp. 1412–1434.

[7] Rajni Jindal, Ruchika Malhotra, and Abha Jain. “Automated classification of security requirements”. *In: 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE.* 2016, pp. 2027–2033.

[8] Sanjanasri Jp *et al.* “A non-exclusive multi-class convolutional neural network for the classification of functional requirements in AUTOSAR software requirement specification text”. *In: IEEE Access* 10 (2022), pp. 117707– 117714.

[9] Prudence Kadebu *et al*. “A classification approach for software requirements towards maintainable security”. *In: Scientific African* 19 (2023), e01496.

[10] Shah Khalid, Shengli Wu, and Fang Zhang. “A multi-objective approach to determining the usefulness of papers in academic search”. *In: Data Technologies and Applications* 55.5 (2021), pp. 734–748.

[11] Mengmeng Lu and Peng Liang. “Automatic classification of non-functional requirements from augmented app user reviews”. *In: Proceedings of the*

ISSN (online): 2581-3048
Volume 10, Issue 5, pp 316-327, May-2026
https://doi.org/10.47001/IRJIET/2026.105042

21st international conference on evaluation and assessment in software engineering. 2017, pp. 344–353.

[12] Ruchika Malhotra *et al*. "Analyzing and evaluating security features in software requirements". *In: 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH). IEEE*. 2016, pp. 26–30.

[13] Gaith Y Quba *et al.* "Software requirements classification using machine learning algorithm's". *In: 2021 international conference on information technology (ICIT). IEEE*. 2021, pp. 685–690.

[14] Antonio Miguel Rosado da Cruz and Estrela Ferreira Cruz. "Machine Learning Techniques for Requirements Engineering: A Comprehensive Literature Review". *In: Software* 4.3 (2025), p. 14.

[15] P Salini and S Kanmani. "Survey and analysis on security requirements engineering". *In: Computers & Electrical Engineering* 38.6 (2012), pp. 1785– 1797.

[16] MAF Saroth, PMAK Wijerathne, and BTGS Kumara. "Automatic multiclass non-functional software requirements classification using machine learning algorithms". *In: 2024 International Research Conference on Smart Computing and Systems Engineering (SCSE)*. Vol. 7. *IEEE*. 2024, pp. 1–6.

[17] Touseef Tahir *et al.* "Cross-Project Multiclass Classification of EARS Based Functional Requirements Utilizing Natural Language Processing, Machine Learning, and Deep Learning". *In: Systems* 13.7 (2025), p. 567.

[18] Sabrina Tiun *et al.* "Classification of functional and non-functional requirement in software requirement using Word2vec and fast Text". *In: journal of Physics: conference series.* Vol. 1529. 4. *IOP Publishing*. 2020, p. 042077.

Citation of this Article:

Landry Giraud Wandji T. (2026). A Multiclass Classification of Software Security Requirements: A Deep Learning Approach. *International Research Journal of Innovations in Engineering and Technology - IRJIET*, 10(5), 316-327. Article DOI https://doi.org/10.47001/IRJIET/2026.105042

****

