Received 4 August 2024, accepted 24 August 2024, date of publication 29 August 2024, date of current version 19 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3452011

RESEARCH ARTICLE

EnsCL-CatBoost: A Strategic Framework for Software Requirements Classification

JALIL ABBAS, CHENG ZHANG, AND BIN LUO, (Senior Member, IEEE)
School of Computer Science and Technology, Anhui University, Hefei 230039, China

Corresponding author: Cheng Zhang (cheng.zhang@ahu.edu.cn)

This work was supported by the Anhui Provincial Natural Science Foundation under Grant 2008085MF189.

ABSTRACT Accurate classification of software requirements, distinguishing between functional and non-functional aspects, is crucial for developing reliable and efficient software systems. However, existing methods often struggle with insufficient semantic understanding and managing diverse software requirements. In this study, we introduce an innovative framework named EnsCL-CatBoost (Ensembled Contrastive Learning with CatBoost) to address these challenges by enhancing classification accuracy and robustness. Our method uses a weighted ensemble of Doc2Vec, Word2Vec, and FastText, leveraging their strengths for richer semantic representation. Unlike conventional strategies, we incorporate contrastive learning with the InfoNCE loss function, boosting discriminative power by clustering similar samples and distancing dissimilar ones, thus enhancing robustness and model generalization. To tackle class imbalance, we integrate SMOTE-Tomek links into the embedding process, achieving balanced class distribution before classification. Additionally, our evaluation extends beyond test datasets to new, unlabeled datasets, demonstrating practical applicability in real-world scenarios. We compare our framework's performance with five machine learning classifiers which we trained using traditional embedding techniques. The results show that our method significantly outperforms others, achieving 94% accuracy. Our research transparently presents tool-based results, highlighting the transformative potential of automation in software requirement classification and setting a new benchmark for practical deployment in diverse environments, paving the way for future research in the field.

INDEX TERMS Software requirements classification, SMOTE-Tomek, EnsCL-CatBoost, weighted ensemble, machine learning classifiers, contrastive learning.

I. INTRODUCTION

Requirement analysis is a critical phase within the software development life cycle, serving as a pivotal step in generating crucial artifacts that guide the subsequent stages of development. In managing and analyzing requirements, text classification emerges as a vital tool. Text classification systematically arranges text documents into predefined categories based on their inherent properties and attributes. In software development, functional requirements (FR) and non-functional requirements (NFR) are two significant types [1]. FRs encompass all features and capabilities that a software system must offer to its users, including user authentication, data archiving, search capabilities, and payment processing. NFRs, on the other hand, represent the constraints and quality characteristics of the software, such as scalability, usability, security, performance, and accessibility. High-quality software systems are built on the foundation of FRs and NFRs, which adapt to user needs and ensure high performance and effectiveness [2]. These requirements are meticulously documented to serve as a blueprint for the development team, ensuring that the final product aligns with the stakeholders' expectations.

A critical challenge in the initial stage of software development is accurately determining user needs from free-form utterances. This process requires substantial time and labour investment, as it sets the direction for the entire project. Manually analyzing text documents to detect requirements

The associate editor coordinating the review of this manuscript and approving it for publication was Mohammad J. Abdel-Rahman.

is infeasible in terms of time, cost, and accuracy. While manual analysis might be manageable for small datasets, it becomes exponentially more challenging as the volume of documents increases to hundreds, thousands, or even millions. For instance, in large-scale software projects, the extensive volume of requirements documentation can overwhelm human analysts, leading to potential oversights and inconsistencies. Thus, automatic requirement classification is essential for solving the requirement classification problem for large volumes of textual documents [3]. Automated systems can process vast amounts of data swiftly and consistently, ensuring that all relevant requirements are identified and categorized accurately.

Despite the availability of clear definitions and documentation, the automatic categorization of naturally written software requirements into functional and various non-functional subcategories remain a significant challenge. This difficulty arises primarily due to the diverse terminologies and sentence structures used by stakeholders and requirements engineers to express similar requirements, as highlighted by Abad et al. [4]. This variability leads to a high level of inconsistency in the articulation of requirements, complicating the automated classification process and increasing the risk of errors. For example, different stakeholders might use varied terminologies to describe performance requirements, making it difficult for automated systems to recognize them as equivalent. This inconsistency necessitates sophisticated techniques capable of understanding and interpreting a wide range of linguistic expressions to ensure accurate classification.

To enhance the accuracy of requirement classification, Machine Learning (ML) techniques have become common for the classification process [5], [6]. ML algorithms utilize data to automate the understanding of user requirements and transform language data into meaningful insights. By leveraging ML, developers can achieve more efficient and user-centric software solutions. The use of ML techniques for extracting NFRs from software requirement documents or other textual sources has gained popularity due to their ability to process and analyze large datasets efficiently. For instance, ML models can be trained on data to recognize patterns and predict requirement categories, significantly reducing the manual effort required and improving accuracy.

However, several limitations and challenges are associated with ML approaches [7]. One primary limitation is the class imbalance issue, where the dataset has a disproportionate ratio of FR to NFRs or vice versa [8]. ML algorithms, when trained on such imbalanced datasets, tend to be biased towards the majority class, resulting in poor classification performance for the minority class. Additionally, the effectiveness of ML classifiers in categorizing software requirements is often hindered by inadequate data preprocessing and suboptimal feature extraction. Insufficient preprocessing leads to a distorted representation of data, impacting the classifiers' ability to accurately interpret and classify software requirements [9]. This issue can result in significant misclassification errors and reduce the overall accuracy of the system. Similarly, rudimentary feature extraction methods fail to capture the complex semantic relationships inherent in the data, limiting the classifiers' performance. These limitations highlight the urgent need for more sophisticated preprocessing, feature extraction, and data augmentation techniques.

To address this, our research introduces a systematic framework for requirement classification, utilizing hybrid embedding, contrastive learning, and class balancing with automatic labeling. By employing these advanced techniques, our framework aims to enhance the quality of data representation and improve the model's ability to discern subtle differences between requirement categories. Our method makes two key contributions:

- **Hybrid Embedding & Contrastive Learning:** We introduce a novel method that combines weighted ensemble techniques and the InfoNCE loss function within a contrastive learning framework. This approach enhances vector representations by dynamically assigning weights based on model performance and optimizing semantic feature capture. Furthermore, it refines embeddings, enhancing discriminative power and generalization capability through contrastive learning.
- **Class Balancing and Automatic Labeling:** Employed the SMOTE-Tomek technique to strategically balance class distribution, thereby optimizing the learning conditions for the classifier. By extending the application of our refined model to automatically classify new, unseen datasets, we demonstrate substantial improvements in predictive accuracy and generalization. We compared the results of our method with existing methods, affirming the robustness and real-world applicability of our approach.

The rest of the paper is structured as follows: Section II delivers an extensive review of Software Requirement Classification Methodologies, offering insights into the existing paradigms and their impact on the field. This sets the groundwork for Section III, where we presented a newly developed framework, EnsCL-CatBoost, including the brief description of the methodologies used in it. Section IV details the experiments conducted and is divided into three subsections: Discussion of Proposed Model Results, Comparative Analysis, and Practical Applicability. In Section V, we address the potential limitations and challenges faced by our study, discussing the threats to its validity. The conclusion is provided in Section VI, which provides a summary of our findings and suggests directions for future research.

II. LITERATURE REVIEW

In recent years, several researchers have made significant contributions to the field of software requirement classification using ML and Deep Learning (DL) techniques. A notable contribution in this arena is the comprehensive review by Binkhonain and Zhao [10] in 2019. Their work focused on the classification of NFRs, highlighting the pivotal role of Natural Language Processing (NLP) and data mining in this process. They provided valuable insights into the evolution of ML algorithms for NFR identification and classification. Working on the classification of requirements, Rahimi et al. [11] introduced a framework specifically for classifying FRs using five distinct methodologies, including Naïve Bayes (NB), Support Vector Machine (SVM), Decision Tree (DT), Logistic Regression(LR). Their findings underscored the superior accuracy of SVM, showcasing its potential in the precise classification of FRs. Advancing the study of ML methodologies, research identified by the study [5] undertook a thorough assessment of text vectorization techniques, specifically employing the Bag-of-Words (BoW) model in conjunction with SVM and K-Nearest Neighbors (KNN) algorithms. This investigation aimed at effectively distinguishing between non-functional requirements NFRs and FRs. The research utilized the PROMISE_exp database as a foundational dataset for this exploration. Through this analysis, the study meticulously evaluated various performance indicators including recall, precision, and the F1 score. They combined BoW with SVM and demonstrated exceptional proficiency, especially notable in binary classification but yield a result of 66% to 72% for subcategories classification of NFR types.

The study [12] delved into the comparative analysis of classic and contextualized word embedding for sentiment analysis, employing advanced DL architectures like Bi-LSTM and Convolutional Neural Network (CNN). This research highlighted the nuanced performance of different embedding, including GloVe, Word2Vec, FastText, and ARBERT, across various benchmark datasets, underscoring the nuanced advantages of generated embedding and the distinctive capabilities of DL models based on dataset size. Tiun et al. [13] directed their attention towards automating the identification and categorization of both Functional requirements (FR) and Non-functional requirements (NFR). They undertook this task by leveraging an array of feature extraction techniques, including Doc2Vec, Bag-of-Words (BoW), FastText, and TF-IDF, in combination with classifiers such as LR, SVM, CNN, and NB. Notably, their findings highlighted FastText as a particularly promising model for classification, as it achieved the highest F1 score among the evaluated methods. This research illuminates the potential of combining diverse feature extraction methods with a range of classifiers to effectively tackle the complexity of requirement classification tasks. In a separate endeavor, Shreda and Hanani [14] conducted a comparative analysis of various classification algorithms, focusing on different feature extraction techniques applied to non-functional requirements sourced from the PURE dataset. Their investigation uncovered that the CNN method surpassed other approaches, achieving an impressive precision value of 92%. These research endeavors collectively underscore the richness of approaches and methodologies employed within the realm of software requirement classification. By showcasing the efficacy of diverse techniques and algorithms, they contribute towards advancing the accuracy and efficiency of software engineering practices. (F. Yucalar,) [15] This study utilizes numerous ML algorithms, including NB, Rotation Forests, CNNs, and BERT, to predict FR and NFR. Utilizing a unique Turkish dataset with 4600 samples, BERTurk emerged as the most successful algorithm, achieving a 95% F-score in discriminating between FR and NFR.

Research labeled as [1] explores the collective efficacy of filter-based feature selection algorithms and diverse machine learning classifiers for classifying requirements across two standard benchmark datasets. Empirical results illustrate that traditional feature selection methods bolster the predictive accuracy of both machine learning and DL models. Furthermore, a newly devised predictor, which integrates traditional feature selection with a self-attention-based linear classifier, surpasses existing methodologies by 4% and 1% in F1-Score on the PROMISE and PROMISE_exp datasets, respectively. The study advocates for further investigation into the utilization of alternative word embedding techniques and the fine-tuning of hyper-parameters in traditional machine learning classifiers to optimize predictive performance. The study [16] delves into the influence of diverse vectorization methods on the classification efficacy of Non-Functional Requirements (NFRs) and their respective categories within expansive contexts. Through comparative experiments spanning five open datasets, nine distinct vectorization methods, and four supervised classification techniques, the study assesses performance metrics utilizing AUC and the Scott-Knott ESD test. The results indicate that select advanced methodologies, particularly when paired with specific classifiers, outshine conventional approaches. Additionally, leveraging pre-trained data offers strategic advantages for certain NFR categories. Ultimately, the study advocates for a nuanced consideration of tailored combinations of vectorization methods and classifiers to optimize the classification accuracy of NFR categories.

Canedo and Mendes [17] conducted a study focusing on text feature extraction and classification of software requirements. Despite utilizing, LR, SVM, NB, and KNN algorithms, their research encountered challenges due to unbalanced data collection, resulting in poor performance. Haque et al. [18] presented an automated approach for Non-Functional Requirements (NFRs) classification, aiming to enhance software development quality. Their methodology incorporates machine learning feature extraction and classification techniques. However, the findings from their analysis, based on the PROMISE dataset, did not yield satisfactory outcomes. Notably, the Stochastic Gradient Descent (SGD) SVM classifier was identified as the most promising, with reported precision, recall, F1 score, and accuracy values of 0.66, 0.61, 0.61, and 0.76, respectively. Researchers highlighted [19] that one of the reasons for such issues is due to imbalanced dataset. The study [8] presents a pioneering hybrid methodology devised to tackle the pervasive challenge of class imbalance in real-world datasets, particularly vital in domains such as disease diagnosis. Leveraging a blend of simulated annealing for under-sampling and a suite

TABLE 1. A review of software requirement classification methodologies.

Study	Techniques/Models	Key Findings	Research Gaps	Our Contribution
Binkhonain and Zhao [10]	Systematic review of 24 ML-based approaches, including 16 ML algorithms	Identified processes for classifying NF and highlighted challenges affecting performance and application	The challenges in applying ML-based approaches and underscores the necessity for collaboration between RE and ML researchers.	Proposed enhancements for better NFR classification accuracy and generalization capabilities through hybrid embedding and contrastive learning
Rahimi et al. [11]	NB, SVM, DT, LR, SVC, Ensemble ML technique	Developed an ensemble ML technique for FR classification by combining 5- ML models.	Dataset was manually balanced, which is not reflective of real-world, imbalanced datasets	Effective handling of imbalanced datasets using data balancing technique
Quba et al. [5]	BoW, SVM, KNN	Well define separation NFR and FRs	Focus on binary classification	Enhanced multi-class classification achieving 94% accuracy on the same PROMISE_exp dataset
Sabbeh and Fasihuddin [12]	Bi-LSTM, CNN, Word2Vec, GloVe, FastText, ARBERT	Comparative analysis of classical and contextualized word embeddings found BERT and BiLSTM performed best for sentiment analysis.	Limited applicability to varied datasets due to domain-specific pre-training and differences in Arabic dialects.	Handling diverse datasets and dialects with robust model training and generalization techniques
Tiun et al. [13]	Doc2Vec, BoW, FastText, TF-IDF, LR, SVM, CNN, NB	Traditional classifiers can outperform complex DL models for short texts.	Limited exploration of automated labeling	Presented automatic labeling of new un-labelled datasets using proposed model
Shreda and Hanani [14]	CNN, Word2Vec, BERT, TF, TF-IDF, LR, SVM, NB	Proposed an automatic ML-based approach to identify and classify NFRs using semantic and syntactic analysis.	Reliance on manually labeled data, which is time-consuming and error-prone, limits generalization across different datasets.	Application of our model to diverse, unseen datasets with substantial improvements in predictive accuracy and generalization
Yucalar [15]	NB, Rotation Forests, CNNs, BERT	BERTurk was used to differentiate FR and NFR in a unique Turkish dataset of 4600 samples, comparing 20 ML algorithms with statistical validation.	Limited dataset scope to Turkish language and specific projects; relies on manually labeled data and majority voting, introducing bias and inconsistency.	Utilized automated labeling to reduce reliance on manual data, enhancing model generalization, applicability, and reducing bias and inconsistency.
Saleem et al. [1]	Filter-based Feature Selection, ML Classifiers, Deep Learning Predictors	FNReq-Net, integrating feature selection with a linear classifier, outperformed state-of-the-art classifiers by 4% on PROMISE and 1% on PROMISE_exp datasets.	Multiclass classification is hindered by subtle inter-class differences, default hyperparameters, and limited benchmark datasets.	Addresses these gaps by employing hybrid embedding techniques within a contrastive learning framework to enhance semantic feature capture, which improves multi-class classification performance.
Leelaprute and Amasaki [16]	Various Vectorization Methods, Supervised Classification Techniques	Advanced methods combined with respective classifiers performed better than traditional approaches	Model validation is limited by out-of-sample bootstrap method, difficulty in handling class imbalances, and reliance on manually preprocessed data.	We enhance NFR classification robustness through SMOTE-Tomek based class balancing evaluation and reduced manual preprocessing.
Canedo and Mendes [17]	LR, SVM, NB, KNN, BoW, TF-IDF, Chi Squared (CHI2)	TF-IDF with LR achieved strong F-measures on PROMISE_exp: 0.91 (binary), 0.74, and 0.78 (general)	Performance is not good due to class imbalance, restricted feature diversity, and manual preprocessing.	Achieved 94% accuracy on the same dataset, addressing class imbalance, feature restriction, and manual preprocessing with hybrid embedding and contrastive learning.
Haque et al. [18]	Feature Extraction (BoW, TF-IDF, N-gram), ML Techniques (NB, KNN, SVM, D-tree, SGD SVM)	SGD SVM was the most promising, but it did not go well due to imbalanced dataset.	Performance is hindered by class imbalance and limited feature extraction techniques.	Address class imbalance and enhance feature representation using a novel hybrid embedding approach, combining with weighted ensemble techniques
Hasannataj Joloudari et al. [19]	SMOTE, CNN, DNN, TL, OSS, NearMiss, ROS, RUS	Combining oversampling, under-sampling, and a CNN achieved 99.08% accuracy on 24 imbalanced datasets, outperforming other models.	further validation on diverse datasets with overlapping, outliers, noise, and multiclass scenarios is needed for generalizability and bias reduction.	Our method enhances generalizability and reduces bias by validating on diverse datasets with multiclass scenarios.
Desuky and Hussain [8]	Simulated Annealing, ML Classifiers (SVM, Decision Tree, KNN, Discriminant Analysis)	Simulated annealing for under-sampling improved classification of imbalanced datasets, outperforming recent techniques on 51 real-world datasets.	It lacks computational efficiency on large-scale datasets and does not compare its method with a wider range of advanced imbalance handling techniques.	Our proposed model demonstrates good computational efficiency, is compared with existing methods, and provides real-world applicability.
Jude et al. [20]	Oversampling, Under-sampling, ML Techniques, Stratified Splitting.	Used stratified splitting, explainable AI, and a hybrid ML algorithm outperformed SMOTE in addressing class imbalance in software defect detection.	Limited dataset diversity; requires validation on real-world applications; potential scalability limitations with hybrid/explainable AI aspects.	Our method validates on diverse real-world applications, enhancing predictive accuracy.

of machine learning classifiers including support vector machine, DT, k-nearest neighbor, and discriminant analysis, the proposed technique exhibits enhanced effectiveness in classifying imbalanced datasets across 51 real-world scenarios. Comparative evaluation against contemporary methodologies validates the superior performance of the proposed approach, indicating its promising capability to alleviate misclassification issues inherent in imbalanced datasets. Jude and Uddin [20] and others have spotlighted the critical issue of dataset imbalance in classification tasks, advocating for strategies like oversampling and under-sampling to ensure a more balanced representation and improve algorithmic fairness and accuracy. These collective endeavors underscore the continuous evolution

Algorithm 1 EnsCL-CatBoost Algorithm

```
1. Input: DataFrame df containing software
   requirements with columns [‘Requirements’,
   ‘Classes’]
2. Result: Trained Model, test metrics, train
   metrics, cv scores
3. procedure EnsCL-CatBoost(df)
4. Step 1: Pre-processing
5. df [‘Requirements’] ← normalize text df
   [‘Requirements’]
6. df [‘Requirements’] ← remove duplicates and
   unnecessary columns from df [‘Requirements’]
7. Step-2: Feature Extraction
8. features ← w2v encode (df [‘Requirements‘])
9. features ← d2v encode (df [‘Requirements‘])
10. features ← ft encode (df [‘Requirements‘])
11. Step 3: Weighted Ensemble Classifier
12. weights ← Extract Weights (w2v, d2v, ft model)
13. Data Splitting
14. Xtrain, Xtest, ytrain, ytest ← train test
    split (featuresres, labelsres, test size =
    0.2, random state = 42)
15. Step 4: Contrastive Learning for Feature
    Refinement
16. For epoch = 1 to N do
17.     Xtrain refined ← CL (Xtrain, learning
    rate = 0.001, batch size = 32)
18.     Xtest refined ← CL (Xtest, learning
    rate = 0.001, batch size = 32)
19. End for
20. Step 5: SMOTE-Tomek Resampling
21. featuresres, labelsres ← smote tomek.fit
    resample (features, df [‘Classes’])
22. Step-6: Training ML Classifier
23. catboost ← CatBoostClassifier(iterations
    = 1000, depth = 4, learning rate = 0.1,
24. for epoch = 1 to N do
25. Train catboost on Xtrain refined, ytrain
26. Step-7: Model Evaluation
27. ypred ← ensemble.predict(Xtest refined)
28. test metrics ← CalculateMetrics(ytest, ypred)
29. ypred train ← ensemble.predict(Xtrain refined)
30. train metrics ← CalculateMetrics(ytrain,ypred
    train)
31. Step-8: Cross-Validation
32. cv scores ← cross val score (ensemble,
    featuresres, labelsres, cv = 10, scoring =
    ‘accuracy‘)
33. Step-9: Further Validation
34. further results ← ensemble.predict(dfunlabeled)
35. Step-10: Final Outputs
36. Performance metrics test metrics
37. Cross-validation scores cv scores
38. Predicted labels for unlabeled dataset
39. end procedure
```

and sophistication of methodologies in the software requirement classification domain, highlighting the fruitful intersection of ML/DL techniques with software engineering practices to address complex challenges. Table-1 provides a concise review on software requirement classification, highlighting the existing methodologies and their outcomes and also identifies areas that require further investigation, thereby contextualizing our research within the broader landscape of software requirement classification studies.

III. PROPOSED FRAMEWORK: EnsCL-CatBoost

Our study presents EnsCL-CatBoost, an innovative framework crafted to automate the balancing and labeling of software requirements, enhancing classification accuracy. The methodology is composed of the following key steps:

- A dataset is selected and processed to clean and prepare the data for analysis.
- Advanced feature extraction methods transform raw text data into meaningful representations, capturing the semantic nuances of software requirements.
- A weighted ensemble of classifiers improves overall performance by combining the strengths of multiple classifiers, enhancing robustness and accuracy.
- Incorporation of contrastive learning refines the feature embeddings, increasing their discriminative power and making it easier to distinguish between different classes of software requirements.
- A hybrid SMOTE-Tomek resampling technique addresses class imbalance by oversampling the minority class and under-sampling the majority class, preventing bias in the classification model.
- The ML model is trained on the features obtained from the previous steps, crucial for learning the patterns and relationships within the data.
- The effectiveness of the EnsCL-CatBoost framework is evaluated using key performance metrics.
- Further validation is performed through the classification using another unlabeled dataset, demonstrating robust capability in handling real-world software requirement classification tasks.

The structure of proposed approach is presented in figure-1 and further elaborated in the algorithm.

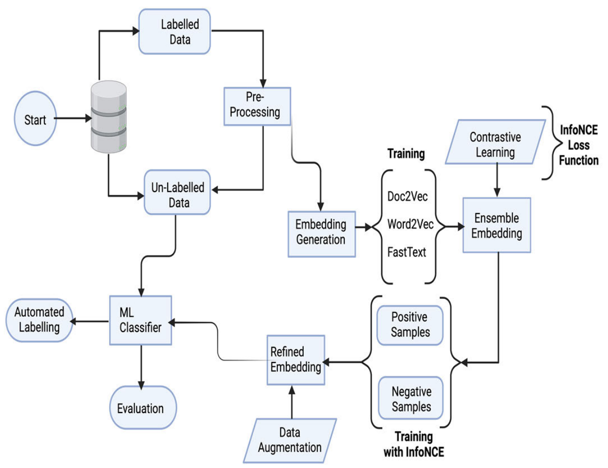


FIGURE 1. Proposed EnsCL-CatBoost framework.

A. DATA PREPROCESSING PHASE

In our research, we utilized two datasets. Dataset-1 is the labeled PROMISE_exp [21], which consists of raw free-text documents containing 969 requirements, including 12 requirement types of NFRs and FR. Supervised learning in any ML approach requires a pre-labeled dataset to train the models, making Dataset-1 essential for our methodology. This dataset allows the supervised learning algorithms to accurately classify requirements based on the pre-labeled data. Dataset-2 is an unlabeled CCHIT

(Certification Commission of Healthcare Information Technology) [22] dataset, containing 1065 artifacts sourced from the requirements documents of a health information system. By using Dataset-1 for training and Dataset-2 for testing, we ensure the robustness and generalizability of our approach. The distribution of requirements and the types used in Dataset-1 are presented in table-2 and its graphical interpretation is shown in figure-2.

TABLE 2. Description of software requirement types in the dataset.

Requirement Type	Lebel	Description	Freq.
Availability	A	Availability is about how often the system is ready to use when you want it.	31
Fault Tolerance	FT	The measure of a system, product, or component's operational fidelity under conditions of hardware or software faults.	18
Legal and Licensing	L	Certifications or authorizations required by the system.	15
Look and feels	LF	Describe the visual presentation and styling of the product.	49
Maintainability	MN	The extent to which a product or system can be altered by the designated maintainer with efficiency and effectiveness.	24
Operability	O	How user-friendly a product or system is, basically if it's easy to use and manage.	77
Performance	PE	Resource utilization effectiveness in producing desired outcomes within predefined parameters.	67
Portability	PO	The level of effectiveness and efficiency in porting a system, product, or component across various hardware, software, or operational contexts.	12
Scalability	SC	The capacity of a product or system to be effectively tailored to meet the demands of evolving hardware, software, or other usage scenarios.	22
Security	SE	The effectiveness of a system in securing information, ensuring access is limited to authorized users.	125
Usability	US	The degree to which a user can efficiently use a product or system to achieve specific goals.	85
Functional	F	Requirements define what a system is supposed to do.	444
Total Requirements			*969*

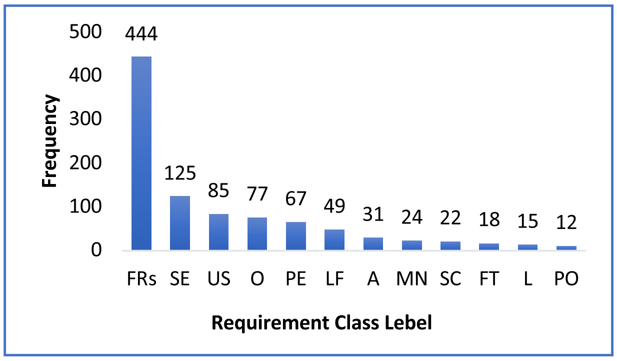


FIGURE 2. Distribution of requirement types in dataset.

Table-3 and table-4 represent a corpus of the first five requirements of the two datasets. The key difference in these datasets, as shown by these tables, is that the first dataset has requirements with class labels, while the second dataset lists specific requirements without classification labels.

TABLE 3. A corpus of Dataset-1.

	Proj. #	Requirements	Class Label
0	1	The system shall refresh display every 60 Seconds	PE
1	1	The application shall match the color of the schema set	LF
2	1	If projected the data must be readable. On	US
3	1	The product shall be available during normal	A
4	1	If projected the data must be understandable	US

TABLE 4. A corpus of Dataset-2.

	art_id	art_title
0	45	In areas where results from multiple patients…
1	46	The system shall (will) provide the ability to…
2	47	The system shall allow authorized users to upd...
3	48	The system shall associate (store and link) ke…
4	49	The system shall be able to support the standa…

The preprocessing steps are outlined as follows:

Import Libraries and Modules: Begin by importing essential libraries in the Jupyter Notebook environment, which is widely favored by data scientists for analyzing and visualizing data, employing programming languages such as Python.

Clean Data: Remove duplicates and handle missing values to ensure the datasets are clean and free from noise.

Remove Non-Essential Columns: Eliminate columns that are not necessary for analysis to streamline the datasets.

Tokenization: Convert the text to lowercase and tokenize it, splitting the text into individual tokens such as words and punctuation symbols.

Filter Tokens: Remove stop-words (e.g., ‘a’, ‘the’, etc.) to minimize noise in the text.

Part-of-Speech Tagging: Tag each token with its corresponding part-of-speech to provide more context for each word.

Lemmatization: Lemmatize the tokens to their base or root forms, utilizing the part-of-speech tags to refine the text and make it more consistent and simpler to analyze.

Prepare Data for Analysis: Combine the processed tokens back into coherent text, ensuring the data is ready for advanced analysis or machine learning tasks.

The corpus of first five entries of both datasets after preprocessing is shown in table-5 and table-6.

B. ADVANCED FEATURE EXTRACTION AND ENSEMBLE MODELING

In this step we transformed preprocessed text data into numerical formats that ML algorithms can work with. In our approach, we transition from traditional to advanced feature

TABLE 5. A corpus of processed Dataset-1.

	Requirements
0	area result multiple patient display system
1	system shall provide ability record author
2	system shall allow authorize user update
3	system shall associate store link key
4	system shall able support standard identify

TABLE 6. A corpus of processed Dataset-2.

	Requirement Text	_Class_
0	system shall refresh display every 60 Seconds	PE
1	application shall match color schema set	LF
2	project data must readable projection screen	US
3	product shall available normal business hour	A
4	project data must understandable projection	US

extraction methods (Doc2Vec, Word2Vec and FastText) to capture various semantic features of the text data. These models are individually trained to transform raw text into meaningful vector representations. Subsequently, a weighted ensemble of these embeddings is computed, aiming to combine the strengths of each embedding technique. Now we give the explanation of these embedding techniques.

1) Doc2Vec

Doc2Vec, an extension of Word2Vec, is designed to create embeddings for sentence, paragraph, or document-length texts, which goes beyond word-level embeddings. Like Word2Vec, it can use a similar architecture (like Skip-gram and CBOW) but adds another layer to handle larger text structures. Each document in Doc2Vec [23], [24] is mapped to a unique vector, and simultaneously, words are also mapped to unique vectors. The model is trained to predict words in a document, similar to how Word2Vec predicts words in a context window.

2) Word2Vec

Word2Vec is a popular word embedding technique crafted by Tomas Mikolov [25]. It uses a neural network model to learn word associations from a large corpus of text. Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence. Word2Vec has two architectures CBOW (Continuous Bag of Words) which predicts the current word based on the context whereas, Skip-gram predicts surrounding context words from the current word. Word2Vec embeddings capture semantic relationships between words, but they do not take word order into account and treat each word as a unique entity.

3) FAST-TEXT

FastText, extends Word2Vec to consider sub-word information. Instead of learning vectors for words directly, FastText [26] represents each word as a bag of character n-grams. This approach allows the embeddings to understand prefixes, suffixes, and the internal structure of words, which helps in handling morphologically rich languages or misspelled words. It can be particularly powerful for languages where the meaning of words changes significantly with slight modifications.

4) WEIGHTED ENSEMBLE

A weighted ensemble [27] of these embedding techniques involves combining the embeddings from Doc2Vec, Word2Vec and FastText in a way that leverages the strengths of each. Here are the steps to create such an ensemble:

- *Generate Embeddings:* Train each model (Doc2Vec, Word2Vec, FastText) on dataset to create embeddings.
- *Normalize Embeddings:* Adjust these embeddings to a standard level to ensure they all contribute equally.
- *Assign Weights:* Allocate weights to each type of embedding based on their performance in tests or cross-validation. Give higher weights to the more effective models.
- *Combine Embeddings:* Calculate a weighted average of these embeddings for each word or document.
- *Fine-tune and Evaluate:* Continuously refine the weights and evaluate the ensemble against a validation set, making necessary adjustments.

By combining different types of embeddings, the ensemble model is able to understand a wide variety of linguistic features, from simple meanings to complex text structures and detailed aspects of words.

C. CONTRASTIVE LEARNING ENHANCEMENT

Following the creation of ensemble embeddings, we introduce a contrastive learning [28] component to refine these embeddings further. The approach of combining multiple embeddings with contrastive learning represents a robust strategy in ML to enhance model performance. Contrastive learning is a powerful strategy in machine learning that focuses on learning to differentiate between similar and dissimilar data points. It does so by pulling embeddings of similar (positive) samples closer together while pushing embeddings of dissimilar (negative) samples apart in the embedding space.

1) InfoNCE LOSS FUNCTION

The core of our contrastive learning implementation involves the InfoNCE loss function, a sophisticated contrastive loss designed to amplify the discriminative power of embeddings. Derived from noise-contrastive estimation, the InfoNCE loss function works by maximizing the mutual information between closely related samples while minimizing it between unrelated ones. During training, this function calculates the similarity between a “query” sample and a “positive” sample (which is similar to the query) in contrast to several "negative" samples (which are dissimilar). By optimizing this loss, the model learns to arrange the embedding space such that it clusters together embeddings of similar items and disperses those of dissimilar items.

This leads to enhanced model robustness against noise and better generalization performance across diverse datasets, ultimately elevating the overall efficacy of our model.

D. DATA AUGMENTATION PHASE

To address the class imbalance, present in the dataset, we apply the SMOTE-Tomek, a hybrid technique combining Synthetic Minority Over-sampling Technique (SMOTE) and Tomek links to balance the dataset. This resampling not only enhances the classifier's ability to generalize across less represented classes but also removes noisy instances, leading to cleaner and more interpretable data. SMOTE [19] is a widely used approach to over-sample the minority class in a dataset. Tomek links are pairs of very close instances, but of opposite classes. Removing the instances of the majority class of each pair increases the space between the two classes, making the decision boundary clearer for classification algorithms. Combining SMOTE and Tomek links, serves two purposes: It over-samples the minority class using SMOTE and then cleans the dataset by removing Tomek links. This combination aims to enrich the dataset with valuable examples of the minority class while also making the class boundaries more distinct.

E. CLASSIFICATION PHASE

In our method, the refined features are then fed into CatBoost, a high-performance ML classifier known for its ability to handle categorical features and reduce overfitting, to evaluate its performance against metrics like accuracy, precision, recall, and F1 score. In conducting comparative analysis, we contrast our approach against five ML classifiers: XGBoost, CatBoost, DT, LR and RF which are trained by traditional embedding techniques.

1) XGBoost

XGBoost [27], an acronym for Extreme Gradient Boosting, represents a sophisticated ML framework that accelerates and enhances the process of constructing gradient-boosted DT. It is renowned for its efficiency in executing parallel tree boosting, which significantly contributes to its dominance in tackling regression, classification, and ranking challenges. The XGBoost algorithm refines its models through successive training of multiple shallow trees, where each new tree corrects the errors made by its predecessors, cumulatively building a robust ensemble. The resultant model is an aggregate of these individual trees, each contributing a weighted influence on the final outcome. XGBoost's prowess lies in its capacity to perform parallel tree construction, diverging from the traditional sequential approach of classic gradient-boosted decision trees. This approach ensures a high degree of accuracy and a more efficient use of computational power, solidifying XGBoost as a powerhouse in the realm of advanced ML algorithms.

2) CatBoost

CatBoost [29], [30], standing for Categorical Boosting, is a state-of-the-art open-source library that excels in classification tasks for datasets with a rich mix of categorical and numerical features, like those found in software requirement datasets. Developed by Yandex, it is particularly effective because it bypasses the need for preliminary data transformation steps such as One-Hot Encoding, thus streamlining the data preprocessing workflow. Its innovative approach includes an algorithm named symmetric weighted quantile sketch (SWQS) that adeptly handles missing data, enhancing the model's robustness and reducing the likelihood of overfitting. With its capability to maintain data integrity and its fast, efficient training process, CatBoost is an advantageous tool for the classification of software requirements, ensuring high predictive accuracy and model interpretability in a domain where these qualities are highly valued.

3) DECISION TREE

A Decision Tree (DT) [31] is a widely used algorithm in ML for both classification and regression tasks, employing a tree-like structure to model decisions and their corresponding outcomes. Each internal node in the tree represents a test conducted on a specific attribute, with each branch indicating the possible outcomes of that test. At the leaf nodes, the tree assigns class labels or numerical values based on the collective decisions made along the branches. DTs excel in scenarios where the relationship between input features and output is nonlinear or intricate, owing to their inherent adaptability. Their non-parametric nature, coupled with ease of interpretation and feature selection capabilities, renders them highly favored in various ML applications. performance. Understanding the underlying concepts and computations involved in DT is vital for effectively employing them in classification tasks.

4) LOGESTIC REGRESSION

Logistic regression (LR) [32] is an ML algorithm for software requirements classification, providing probabilities and classifying new data using datasets. Its versatility extends to addressing both binary and multi-class classification problems. By fitting data to the LR, which follows the sigmoidal equation

$$F(x) = \frac{1}{1 + e^{-x}}$$

LR estimates the likelihood of an event occurring, with outputs constrained between 0 and 1. The threshold of 0.5 serves as the boundary between classes, where predictions exceeding this threshold are assigned to class 1, and those falling below are assigned to class 0. Unlike linear regression, LR outputs probabilities that can be interpreted as the likelihood of belonging to different classes, making it suitable for discrete label assignments. In software engineering contexts, LR aids in determining the most influential variables for classification tasks, making it a valuable tool for software requirement analysis and decision-making.

5) RANDOM FOREST

Random Forest (RF) classifiers [33], [34] which are an amalgamation of DT and ensemble learning, prove to be highly effective for the classification of software requirement datasets, a domain characterized by high-dimensional and complex data structures. Their inherent resistance to over-fitting, combined with the capacity to handle numerous features, makes RF classifiers particularly suited for discerning the nuanced categories of requirements. During their training phase, RF classifiers use bagging to expose each tree to varied data slices and feature sets, enhancing accuracy and generalizability. The out-of-bag error metric, derived from training data not used in the bootstrap sample, provides a convenient and built-in validation mechanism. This means that as the RF model is being trained on a requirement dataset, it is simultaneously being validated, streamlining the model development process. This ability to internally evaluate performance ensures that RFs are tuned to deliver reliable predictions, a vital attribute when sorting and predicting the classification of software requirements.

F. EVALUATION PHASE

We demonstrate the practical applicability of our approach, by preprocessing a new, unlabeled dataset to predict class labels based on learned patterns to assess its impact on model performance. To evaluate the effectiveness of our proposed framework and other machine learning classifiers in categorizing software requirements into functional requirements (FR) and non-functional requirements (NFR) during the evaluation phase, we conduct a detailed analysis using several key performance metrics. We calculate precision, recall, accuracy, and the F1-Score to provide a comprehensive view of their performance. Precision highlights the classifiers' ability to avoid false positives.

$$P = \frac{TP}{TP + FP}$$

Recall emphasizes the sensitivity to detect true positives.

$$R = \frac{TP}{TP + FN}$$

F1-Score combines precision and recall into a single metric, providing a balanced view of both precision and recall.

$$\mathrm{F1-Score} = 2\mathrm{x}\frac{PxR}{P + R}$$

Accuracy provides a baseline for overall performance, indicating the proportion of correctly classified instances.

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$

IV. EXPERIMENTAL RESULTS

In order to provide a structured and detailed presentation, this section is divided into three subsections: Discussion of Proposed Model Results, Comparative Analysis, and Practical Applicability.

A. DISCUSSION OF PROPOSED MODEL RESULTS

In this section, we present the results of our novel method, which employs an ensemble of embedding techniques. After preprocessing a dataset embedding methods such as Doc2Vec, Word2Vec and FastText are employed to convert the cleaned text into numerical representations. Each model generates embeddings by considering different aspects of the data. A weighted ensemble method is then applied to these embeddings by averaging them. This ensemble approach aims to leverage the strengths of each individual embedding technique, potentially leading to a more robust and comprehensive representation of the text data. Once the embeddings are prepared, the dataset, now transformed into a high-dimensional vector space, is typically split into training and testing sets. The training set is used to fine-tune the parameters of the models, while the testing set serves to evaluate their generalizability and effectiveness. The effectiveness of this approach is demonstrated through Principal Component Analysis (PCA), a statistical technique used to simplify the complexity in high-dimensional data. The utility of PCA in our analysis lies in its ability to reduce the dimensionality of embedding space, thus providing a visualizable overview of the distribution and clustering of word embeddings. Here, we examine PCA visualizations of Doc2Vec, Word2Vec and FastText, and Weighted Ensemble embeddings to explore these relationships.

The PCA plot for Doc2Vec figure-3 shows a largely horizontal dispersion with limited vertical spread. This indicates that while the Doc2Vec model captures a specific variance, it might be predominantly capturing similarities over a narrow range of the data's features, potentially reflecting its focus on document-level context which might be less diverse in semantic variation. The Word2Vec embeddings in figure-4 visualization reveals a more dynamically dispersed set of points, suggesting a richer and more nuanced capture of word contexts within the embedding space.

FastText's plot in figure-5 features a dense core with extensions, suggesting its effectiveness at capturing both common and unique linguistic patterns due to its sub-word information processing. Our ensemble approach figure-6 results in a more unified and cohesive clustering compared to individual methods. This indicates that the combined embeddings are not only harmonized but also optimize the representation of words, achieving less variability and enhanced semantic accuracy. Through these PCA plots, we can discern that our ensemble method enhances the embeddings quality by integrating the strengths of individual models, leading to improved performance.

The comprehensive view shown in the combined PCA plot, figure-7 highlights the distinct clustering behaviors of

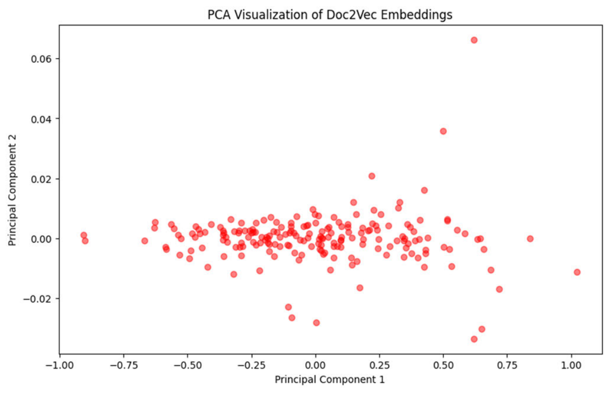


FIGURE 3. Visual insights into Doc2Vec embeddings via PCA.

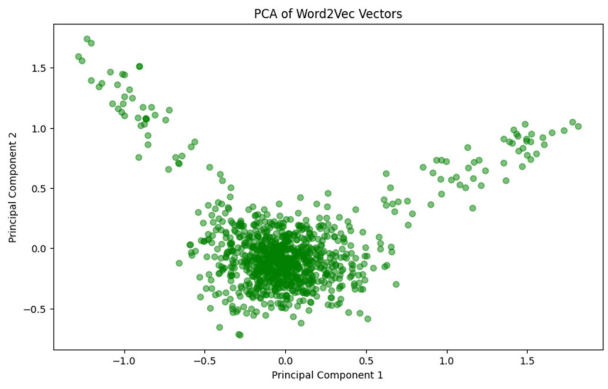


FIGURE 4. Visual insights into Word2Vec embeddings via PCA.

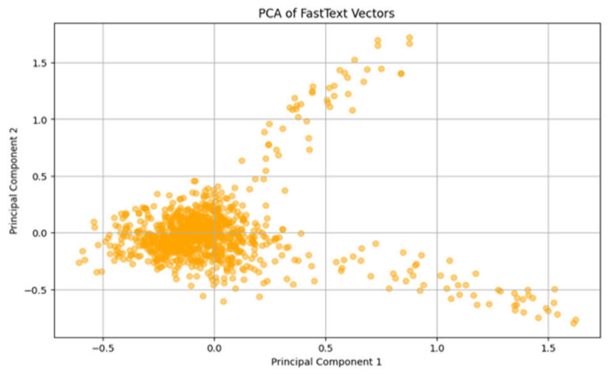


FIGURE 5. Visual insights into FastText embeddings via PCA.

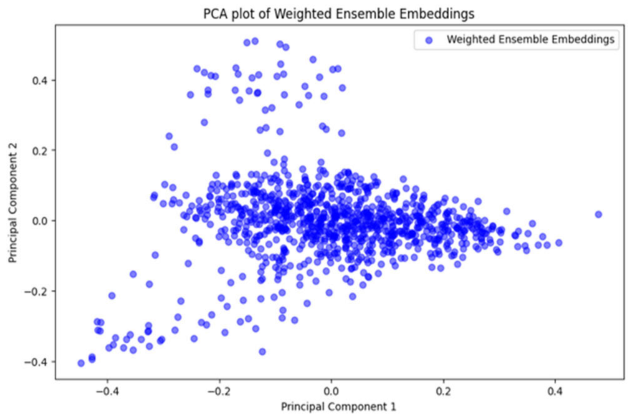


FIGURE 6. Visual insights into weighted ensemble embeddings via PCA.

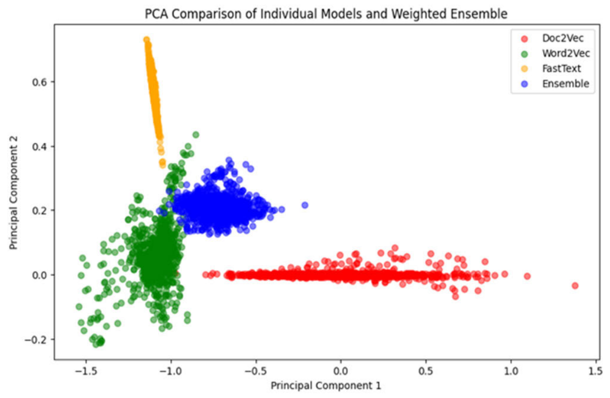


FIGURE 7. Visual insights of PCA comparisons.

each model, with the ensemble method demonstrating an optimal balance of diversity and density in the embeddings. Such uniformity and robustness indicate that the ensemble approach can effectively reduce noise and variability, potentially leading to superior performance in tasks requiring high precision and generalizability across different textual inputs.

After ensemble embedding, we implement the InfoNCE loss function, a contrastive loss mechanism. This function is integral to our methodology, specifically designed to amplify the discriminative power of the embeddings. By leveraging the InfoNCE loss, we ensure that similar (positive) samples are positioned closer together in the embedding space, while dissimilar (negative) samples are distinctly separated. The effectiveness of the InfoNCE loss function in our model is quantitatively depicted in the figure-8, which charts the training loss over successive epochs. Starting with a relatively high loss at the initial epoch, there is a noticeable rapid decrease as the model begins to learn the discriminative features essential for reducing the loss. This sharp decline highlights the model's quick adaptation to the training data, significantly influenced by the contrastive learning dynamics introduced by the InfoNCE loss. As training progresses, the loss trajectory continues to descend, approaching the target loss level (indicated by the dashed red line). This stabilization near the target loss is indicative of the model's effective learning and its ability to generalize well from the training data, representing the enhanced quality and discriminative capabilities of the embeddings facilitated by the InfoNCE loss function.

After the initial training phase with our ensemble model enhanced by the InfoNCE loss function, we applied SMOTE-Tomek to balance the classes within our training data. This step was crucial in addressing class imbalance, which can significantly affect model performance.

Following the application of SMOTE-Tomek, we proceeded to evaluate the model's robustness and generalizability

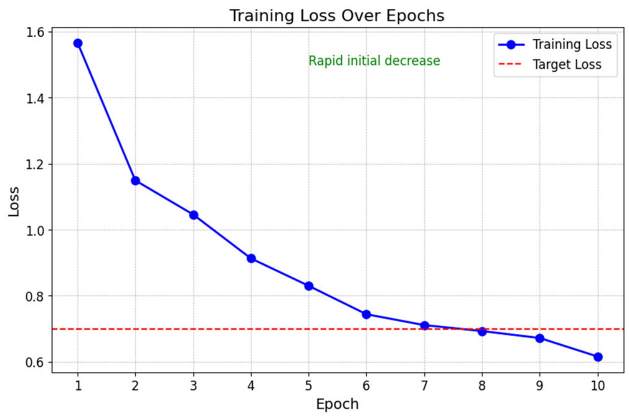


FIGURE 8. Progression of training loss across epochs with target benchmark.

through cross-validation. This method, essential for assessing performance stability across different data subsets, involved partitioning the balanced training data into ten folds. Each fold was then used iteratively as a validation set, with the remaining folds used for training. The results depicted in figure-9 showcase the cross-validation accuracy scores for the EnsCL-CatBoost model trained after applying the SMOTE-Tomek balancing and the InfoNCE loss enhancement on ensembled embeddings show consistently high performance, with an average accuracy of approximately 93.69%. The scores range from a low of about 91.76% to a high of 95.53% across the ten folds, indicating relatively stable performance despite the inherent variability of data in each fold.

The slight variations in accuracy across different folds highlight the model's ability to generalize across diverse subsets of the data while maintaining robustness, owing to effective class balancing and the model's capacity to handle complex patterns in the data. The mean accuracy, marked by the red dashed line in the plot, suggests that the model, coupled with SMOTE-Tomek resampling, effectively addresses class imbalance, enhancing predictive reliability in multi-class classification scenarios.

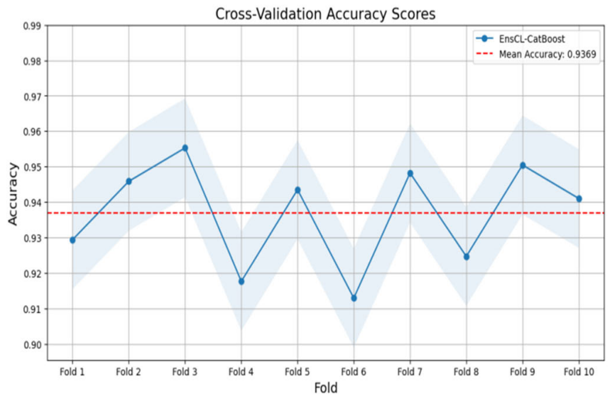


FIGURE 9. Accuracy scores from cross-validation using EnsCL-CatBoost.

The performance of our method is shown by the confusion matrix in figure-10, in which the diagonal cells (which are shaded blue) represent the number of correct predictions for each class, while the off-diagonal cells indicate the instances where the model has made an error. The accuracies, depicted in the titles, are high. This suggests that a EnsCL-CatBoost is performing well across the board. By analyzing confusion matrix, we gained a detailed insights into the classification performance beyond overall accuracy. The classification report of each class of Dataset-1 using EnsCL-CatBoost is provided in table-7.

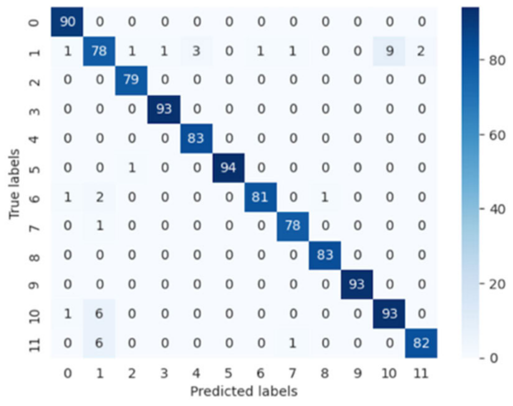


FIGURE 10. EnsCL-CatBoost confusion matrix visualization.

TABLE 7. Classification report for EnsCL-CatBoost model.

Class	Precision	Recall	F1-Score
0	0.99	0.99	0.99
1	0.79	0.60	0.68
2	0.99	1.00	1.00
3	0.99	1.00	1.00
4	0.95	0.95	0.95
5	1.00	1.00	1.00
6	0.86	0.92	0.89
7	0.90	0.96	0.93
8	0.99	1.00	1.00
9	0.99	1.00	1.00
10	0.85	0.89	0.87
11	0.91	0.94	0.92
Accuracy =93.69%			

B. COMPARATIVE ANALYSIS

To provide a thorough and structured evaluation, this sub-section compares the performance of five ML classifiers: XGBoost, CatBoost, Decision Tree (DT), Logistic Regression (LR), and Random Forest (RF) with our proposed EnsCL-CatBoost model. For this comparison analysis of our proposed model with other existing ML classifiers, we performed experiments to train these five ML classifiers with the traditional feature extraction technique TF-IDF and calculated the cross-validation accuracy scores for the classifiers across a tenfold cross-validation process. Results are provided in Figure 11, which illustrates a multiline graph that displays the cross-validation accuracy scores for these five classifiers. The fluctuating lines in the graph highlight the variable performance of each classifier across different folds, emphasizing their inconsistent efficacy in handling the complexities and diverse characteristics of the dataset.

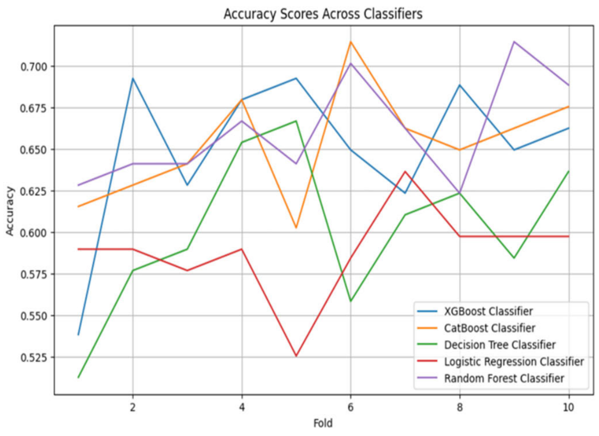


FIGURE 11. Cross validation accuracy scores of 5-ML classifiers trained by traditional embedding.

Next, we will analyze the classification results using four key performance metrics: Accuracy, Precision, Recall, and F1-Score. Table-8 presents a comparative analysis between the EnsCL-CatBoost model and the other ML classifiers. This statistical analysis underscores the superior effectiveness of our proposed model in handling the classification of software requirements and demonstrates its advantage over other ML methods trained by traditional methodology in both accuracy and consistency. The EnsCL-CatBoost framework, which uses a combination of advanced feature extraction and resampling techniques, significantly outperforms the other ML classifiers which are trained using traditional techniques. With an accuracy and recall of 0.936 whereas, precision and F1 score of 0.934. In contrast, the XGBoost classifier showed a notable drop in performance with an accuracy of 0.675258. While the CatBoost and RF classifiers performed slightly close to XGBoost with accuracy of 0.659794 and 0.654639 respectively. The LR classifier also exhibited inferior performance with an accuracy of 0.623711. Lastly, the DT classifier displayed the poorest performance among these ML classifiers, with an accuracy of 0.551546, indicating its inadequacy in handling the dataset's complexity.

TABLE 8. Classification report of EnsCL-CatBoost and 5-ML classifiers.

Classifier	Accuracy	Precision	Recall	F1 Score
EnsCL-CatBoost	0.936911	0.934969	0.936911	0.934601
XGBoost Classifier	0.675258	0.681559	0.675258	0.644779
CatBoost Classifier	0.659794	0.646435	0.659794	0.601600
DT Classifier	0.551546	0.532404	0.551546	0.528407
LR Classifier	0.623711	0.664752	0.623711	0.559380
RF Classifier	0.654639	0.635558	0.654639	0.591886

The graphical representation of the accuracy and precision of the proposed framework in comparison with five ML classifiers is presented in figure-12 which indicates that EnsCL-CatBoost had been thoroughly trained and have a solid foundation to classify software requirements.

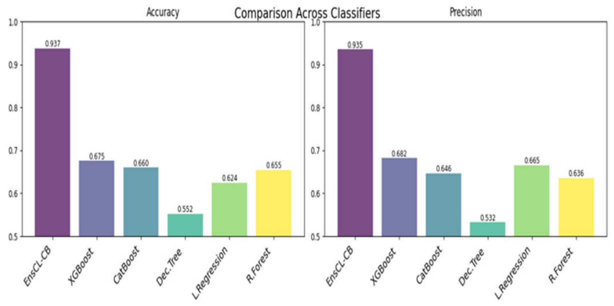


FIGURE 12. Performance evaluation of machine learning classifiers on accuracy and precision metrics.

C. PRACTICAL APPLICABILITY OF EnsCL-CatBoost MODEL

In this subsection, we demonstrate the practical applicability of the EnsCL-CatBoost model by applying it to Dataset-2, an unlabeled dataset of software requirements. This real-world scenario allows us to showcase the model's predictive capabilities in automating the classification process.

EnsCL-CatBoost model is challenged with Dataset-2, an unlabeled dataset of software requirements, a corpus of which is already presented in table-4. This dataset had not been previously classified, thus presenting a real-world scenario for our trained classifier to demonstrate its predictive capabilities. The trained EnsCL-CatBoost predict class labels for the unlabeled Dataset-2 based on learned patterns, thereby enabling the automated classification of requirements and a corpus of it is provided in table-9. This reveals that our model's ability to categorize requirements without the need of manual intervention is particularly impressive. Figure 13 illustrates the distribution of the automatically labeled class names within the dataset-2, with a clear depiction of the frequency of each class. This distribution emphasizes the framework's adeptness at not only categorization but also at handling a wide variety of class labels.

TABLE 9. Automated requirement classification of Dataset-2 using EnsCL-CatBoost.

	Requirements	_Class_
0	area result multiple patient display system	F
1	system shall provide ability record author	SC
2	system shall allow authorize user update	F
3	system shall associate store link key	F
4	system shall able support standard identify	US

In our research, we have not only utilized our model for automated labeling of an unlabeled Dataset-2, but

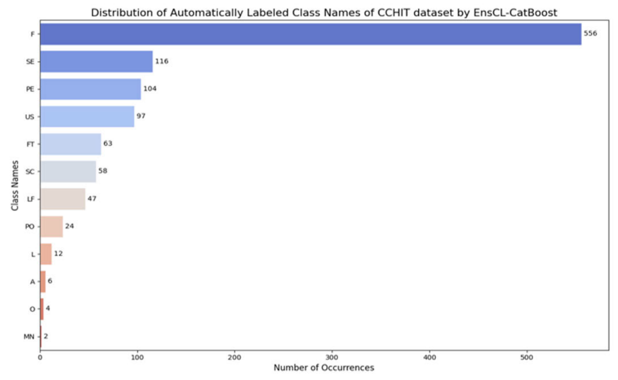


FIGURE 13. **Automated class distribution in Dataset-2 using EnsCL-CatBoost labels showing the frequency of each class.**

also assessed the accuracy of these predicted labels to demonstrate the effectiveness and reliability of our model. To ensure robust evaluation and accuracy of the framework on Dataset-2, we adopted a systematic methodology. Initially, we utilized our trained EnsCL-CatBoost model to predict labels for a small subset of the unlabeled dataset-2. This subset was then balanced using the SMOTE-Tomek technique. We trained the model on this balanced subset and used it to predict labels for the remaining unlabeled data subsets. The accuracy of this predicted dataset-2 was evaluated using 10-fold cross-validation, yielding a mean accuracy of 92.39%, as illustrated in figure 14.

This process helps ensure that the model generalizes well to unseen data. This figure shows the accuracy for each fold during cross-validation, with the initial folds exhibiting higher accuracy possibly due to initial variability and data characteristics. The later folds show a slight decrease, indicating that the model's performance stabilizes. This comprehensive methodology ensured that the predicted labels were reliable and the model generalizes well to unseen data.

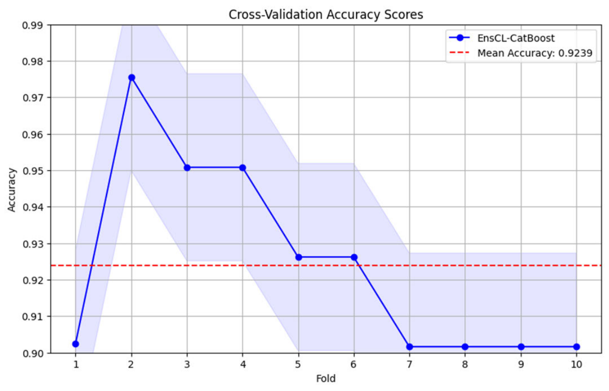


FIGURE 14. **10-Fold cross-validation accuracy scores of daaset-2 using EnsCL-CatBoost model.**

Thus, The EnsCL-CatBoost framework's automation and precision in classifying complex datasets, as demonstrated by its high accuracy and consistency, make it a valuable tool in the realm of ML, particularly for software requirements classification.

V. THREATS TO VALIDITY

In this section, we outlined potential threats that may impact the validity of our research. While we aim to comprehensively address these threats, we also provided information on the steps taken to mitigate their impact.

One significant threat pertains to the accuracy of our measurement and analysis methods in capturing the true relationship between FR and NFR classification within the EnsCL-CatBoost framework. Challenges such as data quality issues, including mislabeled or incomplete requirements, could significantly affect the performance of our classifiers. To address this, we implement rigorous data preprocessing techniques that include automated scripts to identify and rectify inconsistencies, and manual review stages to ensure the integrity of the data. These steps help improve the overall quality of the dataset, thus minimizing the impact of such threats on our research outcomes.

Another threat to the validity relates to the selection of ML models and techniques. While the study employed advanced feature extraction methods and class balancing techniques, the choice and configuration of ML classifier might have introduced biases or limitations. The performance and effectiveness of this can vary significantly with different hyper-parameter settings. To address this, we conduct comprehensive optimization procedures, carefully tuning hyperparameters and evaluating classifier performance across various metrics. By systematically exploring the performance, we ensure that the selected classifier is robust and generalize well across dataset, thereby enhancing the validity and reliability of our research findings.

There is always a concern that a model might perform exceptionally well on training or selected test data but fail to generalize to new, unseen datasets, especially in highly imbalanced scenarios typical of many real-world applications. To address this, we have implemented rigorous cross-validation techniques throughout the model training process to monitor for overfitting. We also used a combination of performance metrics to provide a more holistic view of model performance across various aspects of the data.

VI. CONCLUSION

In this research, we have successfully developed and validated the EnsCL-CatBoost framework, a novel approach that integrates the strengths of weighted ensemble models and contrastive learning to address the challenges of requirement classification in software development. By combining Doc2Vec, Word2Vec, and FastText models optimized with the InfoNCE loss function, the framework enhances semantic representation, discriminative power, noise resilience, and generalization capabilities. The inclusion of SMOTE-Tomek links addresses class imbalance, leading to improved predictive accuracy when these strong and balanced features are loaded into the CatBoost ML model. Our strategic framework achieved a remarkable classification accuracy of 94%, outperforming other ML classifiers trained by the traditional TF-IDF method, including XGBoost (67.5%), CatBoost (65.9%), Decision Tree (55.1%), Logistic Regression (62.4%), and Random Forest (65.5%). This presents a marked distinction from other methods evaluated, emphasizing the strength and flexibility of our proposed method. Additionally, the successful performance of EnsCL-CatBoost on another unseen, unlabeled dataset demonstrates its practical applicability and potential for broader use in automating and refining requirement classification processes. Decision-makers and policymakers can benefit significantly from the EnsCL-CatBoost model's high accuracy and robust handling of class imbalance. This model can enhance project management and resource allocation by providing precise identification and prioritization of software requirements, leading to better decision-making and more efficient use of resources. Thus, our proposed model's robust performance in real-world scenarios provides actionable insights that drive innovation and improve project outcomes.

FUTURE WORK

Our proposed EnsCL-CatBoost framework sets a robust benchmark and opens numerous avenues for future research. Future studies could expand advanced ML and DL techniques to further improve classification performance, explore various neural networks and ensemble methods for additional enhancements, and apply these methodologies to a wider range of datasets to validate their adaptability. Additionally, integrating sophisticated linguistic features and experimenting with diverse contrastive loss functions could enrich text classification capabilities. These directions have the potential to refine requirement classification systems and advance software development by incorporating more strategic, automated processes, fostering innovation in the field.

DATA AVAILABILITY STATEMENT

https://github.com/AleksandarMitrevski/se-requirements-classification

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] S. Saleem, M. N. Asim, L. Van Elst, and A. Dengel, "FNReq-Net: A hybrid computational framework for functional and non-functional requirements classification," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 35, no. 8, 2023, Art. no. 101665.

[2] L. Kumar, S. Baldwa, S. M. Jambavalikar, L. B. Murthy, and A. Krishna, "Software functional and non-function requirement classification using word-embedding," in *Proc. Int. Conf. Adv. Inf. Netw. Appl.* Cham, Switzerland: Springer, Mar. 2022, pp. 167–179.

[3] D. A. López-Hernández, J. Octavio Ocharán-Hernández, E. Mezura-Montes, and Á. J. Sánchez-García, "Automatic classification of software requirements using artificial neural networks: A systematic literature review," in *Proc. 9th Int. Conf. Softw. Eng. Res. Innov. (CONISOFT)*, Oct. 2021, pp. 152–160.

[4] Z. S. H. Abad, O. Karras, P. Ghazi, M. Glinz, G. Ruhe, and K. Schneider, "What works better? A study of classifying requirements," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Sep. 2017, pp. 496–501.

[5] G. Y. Quba, H. Al Qaisi, A. Althunibat, and S. AlZu'bi, "Software requirements classification using machine learning algorithm's," in *Proc. Int. Conf. Inf. Technol. (ICIT)*, Jul. 2021, pp. 685–690.

[6] K. Kaur and P. Kaur, "The application of AI techniques in requirements classification: A systematic mapping," *Artif. Intell. Rev.*, vol. 57, no. 3, p. 57, Feb. 2024.

[7] K. M. Habibullah, G. Gay, and J. Horkoff, "Non-functional requirements for machine learning: Understanding current use and challenges among practitioners," *Requirements Eng.*, vol. 28, no. 2, pp. 283–316, Jun. 2023.

[8] A. S. Desuky and S. Hussain, "An improved hybrid approach for handling class imbalance problem," *Arabian J. Sci. Eng.*, vol. 46, pp. 3853–3864, Jan. 2021.

[9] J. Couto, O. Borges, R. Prikladnicki, and D. D. A. Ruiz, "Challenges in using machine learning to support software engineering," in *Proc. 23rd Int. Conf. Enterprise Inf. Syst.*, 2021, pp. 1–8.

[10] M. Binkhonain and L. Zhao, "A review of machine learning algorithms for identification and classification of non-functional requirements," *Expert Syst. Appl., X*, vol. 1, Apr. 2019, Art. no. 100001.

[11] N. Rahimi, F. Eassa, and L. Elrefaei, "An ensemble machine learning technique for functional requirement classification," *Symmetry*, vol. 12, no. 10, p. 1601, Sep. 2020.

[12] S. F. Sabbeh and H. A. Fasihuddin, "A comparative analysis of word embedding and deep learning for Arabic sentiment classification," *Electronics*, vol. 12, no. 6, p. 1425, Mar. 2023.

[13] S. Tiun, U. Mokhtar, S. Bakar, and S. Saad, "Classification of functional and non-functional requirement in software requirement using Word2vec and fast Text," in *Proc. J. Phys., Conf.*, 2020, vol. 1529, no. 4, Art. no. 042077.

[14] Q. A. Shreda and A. A. Hanani, "Identifying non-functional requirements from unconstrained documents using natural language processing and machine learning approaches," *IEEE Access*, vol. 9, pp. 22991–23003, 2021, doi: 10.1109/ACCESS.2021.3052921.

[15] F. Yucalar, "Developing an advanced software requirements classification model using BERT: An empirical evaluation study on newly generated Turkish data," *Appl. Sci.*, vol. 13, no. 20, p. 11127, Oct. 2023.

[16] P. Leelaprute and S. Amasaki, "A comparative study on vectorization methods for non-functional requirements classification," *Inf. Softw. Technol.*, vol. 150, Oct. 2022, Art. no. 106991.

[17] E. Dias Canedo and B. Cordeiro Mendes, "Software requirements classification using machine learning algorithms," *Entropy*, vol. 22, no. 9, p. 1057, Sep. 2020.

[18] A. Haque, M. Abdur Rahman, and M. S. Siddik, "Non-functional requirements classification with feature extraction and machine learning: An empirical study," in *Proc. 1st Int. Conf. Adv. Sci., Eng. Robot. Technol. (ICASERT)*, May 2019, pp. 1–5.

[19] J. H. Joloudari, A. Marefat, M. A. Nematollahi, S. S. Oyelere, and S. Hussain, "Effective class-imbalance learning based on SMOTE and convolutional neural networks," *Appl. Sci.*, vol. 13, no. 6, p. 4006, Mar. 2023.

[20] A. Jude and J. Uddin, "Explainable software defects classification using SMOTE and machine learning," *Ann. Emerg. Technol. Comput. (AETiC)*, vol. 8, no. 1, 2024.

[21] M. Lima, V. Valle, E. Costa, F. Lira, and B. Gadelha, "Software engineering repositories: Expanding the promise database," in *Proc. XXXIII Brazilian Symp. Softw. Eng.*, 2019, pp. 427–436.

[22] Y. Shin and J. Cleland-Huang, "A comparative evaluation of two user feedback techniques for requirements trace retrieval," in *Proc. 27th Annu. ACM Symp. Appl. Comput.*, 2012, pp. 1069–1074.

[23] A. M. Dakhel, M. C. Desmarais, and F. Khomh, "Dev2vec: Representing domain expertise of developers in an embedding space," *Inf. Softw. Technol.*, vol. 159, Jul. 2023, Art. no. 107218.

[24] M. Bilgin and I. F. Şentürk, "Sentiment analysis on Twitter data with semi-supervised Doc2 Vec," in *Proc. Int. Conf. Comput. Sci. Eng. (UBMK)*, Oct. 2017, pp. 661–666.

[25] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 1–9.

[26] M. Umer, Z. Imtiaz, M. Ahmad, M. Nappi, C. Medaglia, G. S. Choi, and A. Mehmood, "Impact of convolutional neural network and FastText embedding on text classification," *Multimedia Tools Appl.*, vol. 82, no. 4, pp. 5569–5585, Feb. 2023.

[27] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 785–794.

[28] Z. Xian, R. Huang, D. Towey, C. Fang, and Z. Chen, "TransformCode: A contrastive learning framework for code embedding via subtree transformation," *IEEE Trans. Softw. Eng.*, vol. 50, no. 6, pp. 1600–1619, Jun. 2024.

[29] A. A. Ibrahim, R. L. Ridwan, M. M. Muhammed, R. O. Abdulaziz, and G. A. Saheed, "Comparison of the CatBoost classifier with other machine learning methods," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 11, pp. 1–7, 2020.

[30] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: Unbiased boosting with categorical features," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–11.

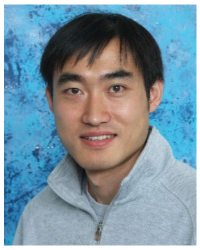
[31] S. Ma and J. Zhai, "Big data decision tree for continuous-valued attributes based on unbalanced cut points," *J. Big Data*, vol. 10, no. 1, p. 135, Aug. 2023.

[32] T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, and M. A. Khan, "Performance analysis of machine learning algorithms in intrusion detection system: A review," *Proc. Comput. Sci.*, vol. 171, pp. 1251–1260, Jan. 2020.

[33] *Random Forest Algorithm*. JavaTpoint. Accessed: Mar. 25, 2024. [Online]. Available: https://www.javatpoint.com/machine-learning

[34] A. Liaw and M. Wiener, "Classification and regression by randomForest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

JALIL ABBAS received the M.S. degree in computer science degree from the University of Lahore, Pakistan. He is pursuing the Ph.D. degree with the School of Computer Science and Technology, Anhui University, China, on a scholarship awarded by the Chinese Government. He served as a Lecturer in colleges and universities, motivated by his dedication to learning and teaching. His various papers have been published in international journals. He has diversified and excellent research interests in the areas of requirement engineering, machine learning, and deep learning.



CHENG ZHANG received the Ph.D. degree from Durham University, Durham, U.K., in 2011. He is currently an Associate Professor with the School of Computer Science and Technology, Anhui University, China. He has received research funding from the National Natural Science Foundation of China (NSFC), Ministry of Education of China, and Anhui Provincial Natural Science Foundation. His research has been published in the journal of IEEE TRANSACTIONS ON SOFTWARE ENGINEERING. His main research interests include the study of empirical software engineering, software architecture, microservices, and cloud workflow. He is a member of the Technical Committee on Software Engineering, China Computer Federation.

BIN LUO (Senior Member, IEEE) received the B.Eng. degree in electronics and the M.Eng. degree in computer science from Anhui University, Hefei, China, in 1984 and 1991, respectively, and the Ph.D. degree in computer science from the University of York, York, U.K., in 2002. He is currently a Full Professor with Anhui University. His research interests include pattern recognition, digital image processing, and cognitive computation. He chairs the IEEE Hefei Subsection. He serves as the Associate Editor-in-Chief for *Visual Intelligence* journal, the Editor-in-Chief for the *Journal of Anhui University* (Natural Science Edition), and an Associate Editor for several international journals, including *Pattern Recognition*, *Pattern Recognition Letters*, and *Cognitive Computation*.