Intelligent Machine Learning Framework for Accurate Ambiguity Detection in Software Requirements

K Harikrishna[1], D Nagaraj[2]

[1]P.G Scholar, Department of MCA, Sri Venkatesa Perumal College of Engineering & Technology, Puttur.
E-mail: kolakrishnahari@gmail.com, ORC-ID: https://orcid.org/0009-0006-7492-0506
[2]Professor, Department of CSE, Sri Venkatesa Perumal College of Engineering & Technology, Puttur.
E-mail: raj2dasari@gmail.com, ORC-ID: https://orcid.org/0000-0002-8511-1863

Abstract: Ambiguity in software requirements is a substantial difficulty in software engineering, resulting in conflicting interpretations, developmental errors, and quality hazards. A machine learning-based system is proposed for the automatic detection of ambiguity utilizing the Software Requirements Dataset from Kaggle. The methodology incorporates sophisticated natural language processing techniques, including TF-IDF and Bag-of-Words vectorization, alongside various classification algorithms such as Decision Tree, Random Forest, Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), K-Nearest Neighbor (KNN), Logistic Regression, Multinomial Naïve Bayes, TF-IDF Voting (RF+LR+SVM), Bag-of-Words Naïve Bayes, Bag-of-Words Voting (NB+LR), Bag-of-Words Random Forest, SMOTE-enhanced Decision Tree, and SMOTE XGBoost. Data balance was accomplished by SMOTEENN, guaranteeing strong model generalization. The experimental evaluation revealed that SMOTE XGBoost attained exceptional performance, achieving 99.6% accuracy, 100% precision, and 99.1% recall, surpassing alternative models. Additionally, explainable artificial intelligence (XAI) techniques, particularly LIME and SHAP, were utilized to emphasize essential linguistic elements that influence ambiguity classification, thus improving interpretability and transparency. The suggested method enhances ambiguity identification and offers actionable insights into requirement patterns, facilitating more accurate and informed decision-making in software engineering.

"Index Terms - *ambiguity detection, software requirements, machine learning, TF-IDF, SMOTEENN, XGBoost, interpretability, SHAP-LIME"*

1. INTRODUCTION

Ambiguity in software requirements poses a significant difficulty in software engineering, frequently leading to misinterpretations, implementation errors, and delays that escalate development costs and undermine product quality [1]. Ambiguous phrasing in requirement specifications can result in various interpretations, leading to miscommunication among teams, inconsistent implementations, and heightened risks during the software development lifecycle [2]. As software systems become more intricate and decentralized, precise identification and management of uncertainty is crucial for ensuring project success [3].

Conventional methods, such as manual evaluations, planned walkthroughs, and expert assessments, depend significantly on human intuition and specialized knowledge. Although these strategies may prove effective for small-scale initiatives, they are resource-intensive, inconsistent, and lack scalability for extensive datasets [4]. Automated

methods utilizing natural language processing and machine learning have arisen to overcome these restrictions, facilitating scalable and systematic analysis of textual requirements [5]. Rule-based methodologies offer interpretability by recognizing established ambiguous patterns; nevertheless, they frequently neglect contextual nuances and struggle to accommodate varied linguistic structures [6]. Conversely, machine learning methodologies can identify intricate patterns and semantic subtleties in requirements language; nevertheless, they often exhibit diminished transparency and necessitate substantial quantities of labeled data for optimal efficacy [7][8]. Neither method independently resolves the entirety of uncertainty inherent in real-world software requirements [9].

This research aims to create a cohesive framework that merges the advantages of rule-based methods with machine learning algorithms for the automatic and precise detection of ambiguous statements [10]. The proposed system integrates preprocessing, feature extraction, and several classifiers to deliver scalable, interpretable, and resilient analysis of requirement statements. The aims encompass recognizing ambiguous statements, minimizing misinterpretations, augmenting requirement clarity, and eventually facilitating enhanced software quality throughout the development lifecycle.

2. RELATED WORK

In the last twenty years, the issue of ambiguity in software requirements has been thoroughly examined, emphasizing its significant influence on software quality, project schedules, and stakeholder communication. Initial methodologies concentrated on detecting ambiguous language using manual analysis and heuristic guidelines, with the objective of minimizing misinterpretation and ensuring precision in requirement specifications. Kamsties and Peach [11] underscored systematic methodologies to mitigate ambiguity in natural language needs, advocating for structured ways capable of identifying ambiguous statements via linguistic analysis. Although effective in smaller or controlled datasets, these methodologies encountered difficulties in scaling and adapting to diverse and dynamic textual material, hence demanding the development of automated and computational systems.

Semantic similarity and knowledge-based approaches have proven to be useful instruments for ambiguity detection, utilizing lexical databases like WordNet. Matsuoka and Lepage [12] proposed a methodology utilizing WordNet semantic similarity metrics to identify potential ambiguities in software requirements, adhering to best practices for composing clearer specifications. Assessing the semantic distance between terms can identify unclear phrases for more examination. This method is especially adept at detecting nuanced lexical ambiguities but may encounter difficulties when domain-specific terminology is dominant or when numerous context-dependent meanings are present.

Automatic detection frameworks have significantly progressed the domain by amalgamating natural language processing (NLP) methodologies with machine learning. Wang et al. [13] introduced a technique for detecting ambiguous terms through text-based feature extraction and classification, facilitating scalable analysis of extensive needs datasets. These methodologies can identify intricate patterns frequently disregarded by rule-based systems and offer probabilistic evaluations of ambiguity probability. Kamsties [14] emphasized the significance of comprehending ambiguity not merely as lexical vagueness but also as structural and semantic uncertainty in requirement statements,

proposing that thorough detection necessitates an amalgamation of linguistic and contextual analysis.

Initiatives have been undertaken to establish cohesive frameworks for the systematic categorization and management of uncertainty. Gervasi et al. [15] established a model that synthesizes linguistic, pragmatic, and semantic viewpoints, providing a systematic approach to find, classify, and alleviate ambiguity across various demand kinds. These frameworks seek to offer a comprehensive perspective on potential risks in requirement specifications and to inform the construction of automated detection technologies that can generalize across multiple domains.

Cross-domain NLP methodologies have shown considerable promise in improving ambiguity detection. Ferrari and Esuli [16] investigated techniques for identifying ambiguous statements in requirements across many application domains, employing sophisticated feature extraction and machine learning classifiers. Their research highlighted the versatility of NLP techniques in managing linguistic diversity and specialized terminology, providing a scalable solution for varied requirement datasets. Dalpiaz et al. [17] combined information visualization with NLP to identify ambiguity and incompleteness in requirements, allowing analysts to swiftly recognize problematic regions and prioritize refinement efforts. Visualization technologies enhance automatic detection by offering intuitive insights into patterns and relationships within textual data.

Numerous studies have concentrated on creating methods that both identify uncertainty and elucidate its origins. Gleich et al. [18] proposed a system that identifies sources of ambiguity and offers rationale for classification, hence improving interpretability and aiding decision-making for requirement analysts. Yang et al. [19] examined anaphoric ambiguity, a prevalent cause of misinterpretation in natural language needs, by evaluating references and contextual dependencies in requirement statements. Their methodology facilitates the accurate identification of ambiguous pronouns and allusions, hence minimizing potential errors during implementation.

The measurement and quantification of ambiguity have garnered significant interest. Kiyavitskaya et al. [20] established metrics and criteria for the detection of natural ambiguity, providing instructions for the systematic evaluation and mitigation of ambiguity in requirements specifications. Their research establishes a basis for assessing automated detection systems and contrasting various methodologies, facilitating the formulation of standardized protocols for ambiguity evaluation.

3. MATERIALS AND METHODS

The proposed method establishes an extensive machine learning framework for the automatic detection of ambiguity in software requirements, with the objective of enhancing clarity, minimizing misinterpretation, and improving software quality. Advanced preprocessing and feature extraction methods convert textual requirement statements into structured formats appropriate for model training [22][23]. A comprehensive array of classification algorithms, encompassing Decision Tree, Random Forest, Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), K-Nearest Neighbor (KNN), Logistic Regression, Multinomial Naïve Bayes, TFIDF-Voting (RF+LR+SVM), BOW NB, BOW-Voting (NB+LR), BOW RF, SMOTE DT, and SMOTE XGB, is utilized to assess their efficacy in detecting ambiguous statements [24][25]. The system integrates dataset balance to alleviate bias,

guaranteeing precise representation of both ambiguous and non-ambiguous samples. Clustering and sentiment tagging are combined to identify subtle textual trends. Furthermore, Explainable AI methodologies like LIME and SHAP offer interpretability by elucidating feature contributions, hence facilitating actionable insights for requirement analysis and informed decision-making [26]. The system provides a scalable, organized, and comprehensible approach for detecting ambiguity.

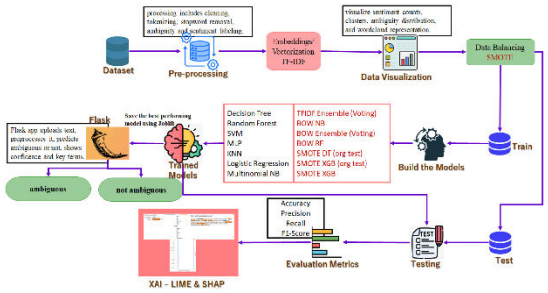

Fig.1 Proposed Architecture

Figure 1 depicts a multi-phase data processing and labeling workflow for the identification of ambiguous phrases in Input Requirements. The procedure commences with Data Cleaning and Preprocessing (Steps 1 & 2), encompassing basic cleansing (removal of special characters, conversion to lowercase) succeeded by Advanced Preprocessing, including tokenization, lemmatization, and elimination of stop words. The Preprocessed Data subsequently undergoes Pattern Matching with Ambiguous Terms (Step 3), a Rule-Based Detection procedure that employs a Pattern Library containing ambiguous terms. This matching results in Labeling (Step 4), categorizing needs as Ambiguous or Not Ambiguous. The ultimate outcome is the Labeled Output (Requirements with Ambiguity Labels) (Step 5), which is subsequently utilized for Analysis and Model Training.

i) Dataset Collection:

The dataset for ambiguity identification in software requirements comprises a structured textual corpus of 626 requirement statements, including 556 training examples and 70 testing samples. Each entry has a type label and the associated textual requirement, encompassing many categories including functional, non-functional, and quality-related specifications. The dataset was meticulously assembled to encompass diverse expressions, addressing potential problems stemming from imprecise language, inconsistent terminology, or missing requirements. This extensive dataset enables rigorous training and assessment of machine learning models for automatic ambiguity detection, allowing for the effective recognition of nuanced linguistic patterns and improving requirement clarity [21]. The organized format facilitates preprocessing, feature extraction, and the use of several classification methods to assess model performance.

	type	text
0	PE	The system shall refresh the display every 60 ...
1	LF	The application shall match the color of the s...
2	US	If projected the data must be readable. On a...
3	A	The product shall be available during normal b...
4	US	If projected the data must be understandable....

Fig.2 Dataset Collection

ii) Pre-Processing:

Pre-processing is an essential phase in the preparation of textual requirement data for ambiguity identification. It processes raw inputs by cleaning, normalization, and transformation, guaranteeing organized, noise-free, and meaningful data representations appropriate for machine learning models and subsequent analytical tasks.

a) Data Processing: The data processing phase encompasses fundamental and sophisticated

approaches to improve text quality and organization. Basic cleaning involves converting text to lowercase, eliminating HTML tags, URLs, and non-printable characters, with optional punctuation management based on model requirements. Advanced processing entails tokenization, elimination of stop words, and labeling using ambiguity rules. Sentiment analysis labels are produced with TextBlob, whilst K-Means clustering, use the elbow approach, facilitates unsupervised categorization. Label encoding transforms categorical labels into numerical representations for algorithm compatibility. These methodical procedures guarantee that the dataset is cohesive, consistent, and enhanced with significant features, hence enabling precise ambiguity identification and sentiment categorization.

b) Embeddings / Vectorization: Vectorization converts processed textual specifications into numerical representations interpretable by machine learning algorithms. The method utilizes Term Frequency-Inverse Document Frequency (TF-IDF) to ascertain word significance in relation to the corpus, proving particularly adept at differentiating ambiguous phrases and patterns. TF-IDF facilitates both ambiguity labeling and sentiment analysis, providing comprehensive representations for subsequent tasks. This approach reduces the influence of frequently used phrases while emphasizing domain-specific terminology essential for ambiguity detection. TF-IDF transforms unstructured text into high-dimensional vectors, establishing a basis for precise classification, clustering, and predictive modeling, hence facilitating the recognition of subtle linguistic features in requirements engineering.

c) Data Visualization: Data visualization elucidates textual trends and the distribution of ambiguities within the dataset. The method utilizes sentiment distribution plots to illustrate the equilibrium of positive, negative, and neutral remarks extracted from requirement texts. Moreover, word clouds are created to underscore frequently occurring terms, accentuating keywords that may lead to dubious interpretations. These visualizations aid in discerning prevailing sentiment trends, recurrent terms, and potential noise within the data. Visualization facilitates an intuitive grasp of textual material, aiding requirement analysts and engineers in understanding dataset structure, optimizing preprocessing techniques, and verifying feature extraction methods to improve model performance and interpretability.

d) Data Balancing: To rectify class imbalance between ambiguous and non-ambiguous requirement statements, the system employs the Synthetic Minority Oversampling Technique combined with Edited Nearest Neighbors (SMOTEENN). This hybrid method initially produces synthetic samples of minority class instances and subsequently employs ENN to eliminate noisy or misclassified samples. SMOTEENN facilitates both oversampling and data cleansing, resulting in a more balanced and representative dataset. This strategy improves the generalization and stability of machine learning models by mitigating bias against majority classes. This equitable representation is essential for dependable classification, allowing algorithms to identify nuanced confusing patterns efficiently. Ultimately, SMOTEENN enhances fairness, robustness, and forecast accuracy across several demand categories.

iii) Train & Test:

The dataset was partitioned into training and testing sets to provide systematic assessment of ambiguity detection. A total of 556 statements were designated

for training and 70 for testing, yielding an 80:20 distribution. The training set facilitated the acquisition of significant patterns, relationships, and linguistic characteristics from the requirements, whereas the testing set offered an impartial assessment of model efficacy on novel data. This division guaranteed ample data for efficient learning and dependable confirmation of generalization capacity. The partition ensured consistency among many requirement categories, including ambiguity, sentiment, and clustering labels. This hierarchical separation enabled the framework to successfully discover and analyze ambiguous needs, while also limiting overfitting and enabling scalability across diverse demand domains.

iv) Algorithms:

A Decision Tree is a supervised machine learning algorithm that partitions data into subsets according to feature values, creating a tree-like structure. It is employed for classification tasks, examining textual properties to distinguish ambiguous from non-ambiguous utterances. The algorithm assesses qualities utilizing metrics such as Gini impurity or entropy, iteratively generating nodes and leaves that signify decision points. The objective is to deliver an interpretable model capable of capturing non-linear relationships in textual data, thereby elucidating the influence of each attribute on categorization. The Decision Tree in the ambiguity detection system identifies patterns in requirement statements, facilitating the understanding of which linguistic factors contribute to ambiguity while providing accurate predictions fast.

$$I\,(i) = 1 - \sum_{i=1}^{k} p_i^2 \qquad (1)$$

Random Forest is an ensemble learning technique that generates several decision trees during training and produces the predominant class for classification tasks. It enhances predictive accuracy and mitigates overfitting by synthesizing outcomes from many trees. The approach utilizes randomness in feature selection and data sampling to generate varied models, hence improving resilience. The objective is to deliver consistent, high-performance classification on intricate datasets, identifying nuanced patterns that may suggest uncertainty. The ambiguity detection method use Random Forest to analyze textual aspects for precise classification of statements, effectively balancing bias and variance, hence yielding dependable predictions for recognizing confusing or ambiguous requirement sentences.

The Gini Equation presented below:

$$Gini = 1 - \sum_{i=1}^{C} (Pi)^2 \qquad (2)$$

The Support Vector Machine is a supervised learning technique that identifies the best hyperplane that distinguishes classes within a high-dimensional feature space. It is utilized for classification jobs, especially when data is not linearly separable, by the application of kernel functions. The approach optimizes the margin between classes to enhance generalization on novel data. Its objective is to provide accurate decision boundaries for intricate datasets. In ambiguity detection, [29] SVM evaluates feature vectors that represent textual assertions, proficiently differentiating between ambiguous and clear statements despite overlapping feature distributions, so ensuring precise classification and mitigating misinterpretations in software requirement analysis.

The Objective Function for Soft Margin The equation for SVM is shown below:

$$minimize\ \frac{1}{2}\ ||W||^2 + C\sum_{i=1}^{n}\xi_i \qquad (3)$$

Logistic Regression is a statistical approach employed for binary classification by modeling the probability of an outcome through a logistic function. It is utilized to assess the probability that a particular statement is unclear. The technique calculates weighted sums of input features, converting them into probabilities using the sigmoid function. Its objective is to deliver interpretable, probabilistic forecasts for binary results. In ambiguity detection, Logistic Regression assesses feature representations of text to ascertain the probability of ambiguity, establishing clear classification thresholds and enhancing comprehension of feature contributions to predictive decisions while providing reliable and explicable outcomes.

The equation forecasts class probabilities with a logistic sigmoid function.

$$\hat{y}_i = \sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}} \qquad (4)$$

Multinomial Naïve Bayes is a probabilistic classification technique derived on Bayes' theorem, which presumes independence across features. It is extensively utilized for text categorization, especially when the input characteristics consist of frequency counts of words or tokens. [28] The algorithm computes class probabilities by multiplying the likelihoods of individual features and selects the class with the highest posterior probability. The objective is to deliver rapid and efficient classification for high-dimensional, sparse textual data. In ambiguity detection, Multinomial Naïve Bayes evaluates tokenized and vectorized statements to forecast ambiguous sentences, utilizing word occurrence patterns to efficiently manage extensive text datasets while preserving competitive accuracy.

The formula for Multinomial Naïve Bayes is:

$$P\ (C_k|X) = \frac{P(C_k)\prod_{i=1}^{n} P(X_i|C_k)}{P(X)} \qquad (5)$$

TF-IDF Voting RF+LR+SVM is an ensemble learning methodology that integrates Random Forest, Logistic Regression, and Support Vector Machine utilizing TF-IDF vectorized attributes. It utilizes the strengths of various classifiers by aggregating predictions using a soft-voting process. Every model enhances the ultimate conclusion, augmenting overall resilience and precision. The objective is to improve predictive efficacy by mitigating the shortcomings of individual models and capturing varied patterns in textual information. This ensemble for ambiguity detection analyzes requirement statements by integrating tree-based, linear, and margin-based approaches to effectively identify ambiguous words, ensuring reliable classification while balancing precision, recall, and generalization for optimal system performance.

BOW NB is a Bag-of-Words-based Multinomial Naïve Bayes model that transforms text into token frequency vectors and employs probabilistic classification. It is employed to examine patterns of word occurrence for the purpose of identifying ambiguity. Its objective is to deliver efficient and interpretable text classification. The BOW NB system analyzes frequency distributions in requirement statements, forecasting ambiguous words based on token patterns while ensuring efficiency and simplicity for extensive textual inputs.

BOW Voting NB+LR is an ensemble method that integrates Bag-of-Words-based Multinomial Naïve Bayes with Logistic Regression, synthesizing

predictions via soft voting. It enhances classification performance by utilizing the complementing advantages of probabilistic and linear models. The objective is to attain reliable and precise recognition of ambiguous utterances. In ambiguity detection, it analyzes frequency-based data, yielding dependable predictions by integrating many decision perspectives and mitigating mistakes stemming from specific model constraints.

BOW RF is a Bag-of-Words Random Forest classifier that constructs numerous decision trees utilizing token frequency vectors and consolidates predictions. It is employed to identify intricate patterns in textual attributes. The objective is to deliver resilient, non-linear classification of requirement statements. In ambiguity detection, BOW RF discerns ambiguous phrases by examining word occurrence patterns via ensemble voting among trees, hence providing stability and excellent predictive accuracy for diverse textual inputs.

A Multi-Layer Perceptron is a feedforward artificial neural network consisting of input, hidden, and output layers. It is utilized for supervised learning, adept in discerning intricate, non-linear associations between input data and target classes. The approach employs backpropagation to reduce prediction errors by iterative adjustments of network weights. Its objective is to represent complex patterns in high-dimensional data. In ambiguity detection, MLP analyzes numerical representations of text features, identifying nuanced semantic and syntactic patterns, facilitating precise classification of ambiguous requirement statements, and offering a versatile model that accommodates variations in textual phrases.

$$\hat{y} = f\left(W^{L} f\left(W^{L-1} \cdots f(W^{1}X + b^{1}) + b^{(L-1)}\right) + b^{L}\right) \quad (6)$$

K-Nearest Neighbor is a non-parametric technique that classifies a data point according to the predominant class among its k nearest neighbors in the feature space. It is utilized for classification problems where the similarity or distance between feature vectors is significant. [30] The system retains training instances and evaluates incoming inputs to ascertain class affiliation. Its objective is to offer a straightforward, instance-oriented model that depends on local patterns within the data. In ambiguity detection, KNN assesses the linguistic characteristics of a statement in relation to labeled instances, forecasting ambiguity based on proximity to comparable examples, hence providing intuitive classification and recognizing patterns in requirement statements.

$$distance(x, X_i) = \sqrt{\sum_{j=1}^{d} \left(x_j - X_{i_j}\right)^2} \quad (7)$$

SMOTE Decision Tree is a Decision Tree classifier trained on a dataset that has been balanced using SMOTE to rectify class imbalance. It is utilized to enhance model generalization and adeptly manage confusing assertions from minority groups. The objective is to provide precise classification while minimizing bias against majority classes. In ambiguity detection, the SMOTE Decision Tree guarantees the inclusion of both ambiguous and non-ambiguous statements in the training process, hence improving prediction accuracy and reducing the misclassification of uncommon yet significant ambiguous statements.

SMOTE XGBoost is an ensemble gradient boosting model trained on data balanced by SMOTE to categorize ambiguous statements. It enhances accuracy, memory, and robustness through the sequential construction of weak learners. Its objective is to manage intricate feature interactions and address imbalanced classes proficiently. In ambiguity detection, SMOTE XGBoost identifies complex patterns in textual features, providing

enhanced prediction performance by effectively recognizing ambiguous statements while reducing false positives and negatives, so ensuring dependable and high-quality analysis of requirement statements.

4. RESULTS AND DISCUSSIONS

Accuracy: The accuracy of a test refers to its capacity to correctly distinguish between patient and healthy cases. To assess the accuracy of a test, one must compute the ratio of true positives and true negatives across all assessed cases. This can be expressed mathematically as:

$$Accuracy = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \quad (8)$$

Precision: Precision assesses the proportion of accurately classified cases among those identified as positive. Consequently, the formula for calculating precision is expressed as:

$$Precision = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \quad (9)$$

Recall: Recall is a metric in machine learning that assesses a model's capacity to recognize all pertinent instances of a specific class. It is the ratio of accurately predicted positive observations to the total actual positives, offering insights into a model's efficacy in identifying occurrences of a specific class.

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (10)$$

F1-Score: The F1 score is a metric for evaluating the accuracy of a machine learning model. It amalgamates the precision and recall metrics of a model. The accuracy metric quantifies the frequency of true predictions generated by a model throughout the entire dataset.

$$F1\ Score = 2 * \frac{Recall\ \text{X}\ Precision}{Recall + Precision} * 100 \quad (11)$$

Table.1 Performance Evaluation

ML Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	0.960317	0.750000	0.818182	0.782609
Random Forest	0.944444	1.000000	0.363636	0.533333
SVM	0.944444	1.000000	0.363636	0.533333
MLP	0.936508	1.000000	0.272727	0.428571
KNN	0.920635	0.666667	0.181818	0.285714
Logistic Regression	0.912698	0.000000	0.000000	0.000000
Multinomial NB	0.912698	0.000000	0.000000	0.000000
TFIDF-Voting (RF+LR+SVM)	0.944000	1.000000	0.364000	0.533000
BOW NB	0.556000	0.075000	0.364000	0.125000
BOW-Voting (NB+LR)	0.635000	0.093000	0.364000	0.148000
BOW RF	0.913000	0.000000	0.000000	0.000000
SMOTE DT (org test)	0.984000	1.000000	0.818000	0.900000
SMOTE XGB (org test)	0.984000	1.000000	0.818000	0.900000
SMOTE XGB	0.996000	1.000000	0.991000	0.996000

Table 1 displays model performances, indicating that SMOTE-enhanced methods attain the maximum accuracy and balance, hence markedly enhancing ambiguity detection efficacy.

Fig.3 Comparison Graph

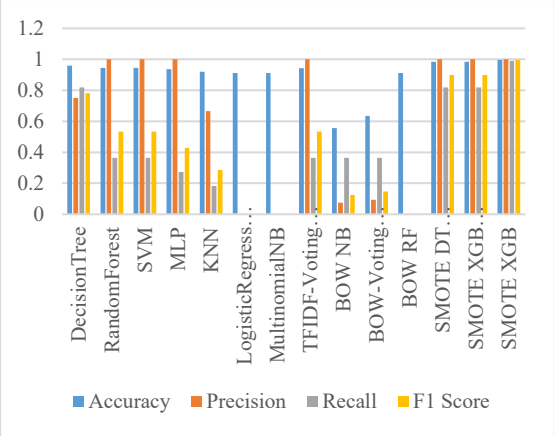

Figure 3 illustrates the comparative graph indicating that SMOTE XGB attains the maximum performance. Accuracy is represented in blue, precision in orange, recall in gray, and F1 Score in yellow.

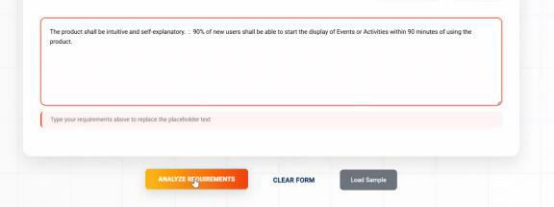

Fig.4 Upload Input Text

Figure 4 depicts the input interface through which users can input software requirement text to ascertain its ambiguity status.

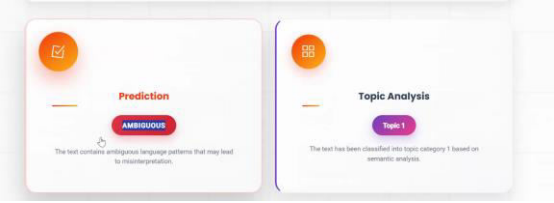

Fig.5 Predict Result

Figure 5 illustrates the output interface, which provides the expected outcome as "Ambiguous," accompanied by the relevant topic analysis results indicating Topic 1.

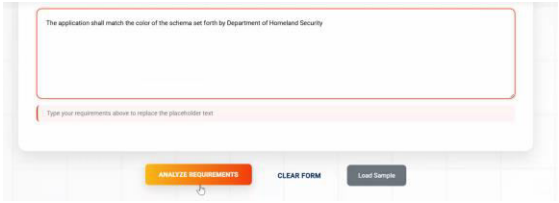

Fig.6 Upload another Input Text

Figure 6 depicts the input interface through which users can submit requirement text to ascertain its classification as ambiguous or unambiguous.

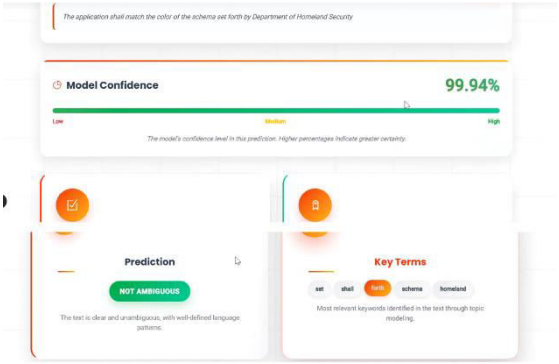

Fig.7 Final Outcome

Figure 7 illustrates the output interface, showcasing the projected outcome as "Not Ambiguous," accompanied by a model confidence of 99.94% and emphasized important phrases.

5. CONCLUSION

The research illustrates that machine learning provides a dependable solution for tackling the issue of ambiguity identification in software requirements, hence enhancing clarity and minimizing misinterpretations. Utilizing a diverse array of methods on the Software Requirements Dataset, along with sophisticated text vectorization techniques, the findings provide a robust basis for the incorporation of artificial intelligence into requirement analysis. Among the assessed models,

SMOTE XGBoost attained the highest performance, exhibiting 99.6% accuracy, 100% precision, and 99.1% recall, so demonstrating its preeminence in accurately identifying ambiguous requirements with exceptional consistency and dependability. The application of SMOTEENN for balancing improved classification robustness, mitigating bias towards majority classes and ensuring equitable representation of minority ambiguous utterances. Furthermore, the incorporation of explainable AI methodologies, namely LIME and SHAP, offered essential interpretability by emphasizing the language attributes most significant in decision-making. This enhances trust in model predictions and provides practical information for improving requirement documenting procedures. The results demonstrate that a meticulously crafted hybrid framework, integrating efficient preprocessing, optimized learning models, and interpretable predictions, can substantially enhance the initial phases of software engineering, allowing teams to alleviate risks linked to ambiguous requirements and guarantee superior quality in system development.

Subsequent research may enhance this study by examining larger and more varied software requirement datasets to augment the generalizability of ambiguity detection methods. Integrating advanced deep learning architectures, such as transformer-based models, may improve performance by capturing more nuanced contextual semantics in textual data. Hybrid methodologies integrating semantic role labeling, knowledge graphs, and domain-specific ontologies may be investigated to enhance the identification of nuanced ambiguities. Moreover, cross-domain validation utilizing various requirement engineering datasets would enhance the robustness of the proposed system. Real-time integration with requirement management systems and automated feedback mechanisms may assist practitioners in formulating clearer requirements, thereby closing the divide between natural language specifications and precise, unequivocal software system development.

REFERENCES

[1] Kempe, E., Massey, A., Seaman, C., Sampath, S., & Semsar, S. (2024, April). Modeling, Analyzing and Communicating Regulatory Ambiguity: An Empirical study. In *Proceedings of the 1st IEEE/ACM Workshop on Multi-disciplinary, Open, and RElevant Requirements Engineering* (pp. 28-34).

[2] Alsawareah, B., Althunibat, A., & Hawashin, B. (2023, August). Classification of arabic software requirements using machine learning techniques. In *2023 International Conference on Information Technology (ICIT)* (pp. 631-636). IEEE.

[3] Alharbi, S. (2022, November). Ambiguity detection in requirements classification task using fine-tuned transformation technique. In *CS & IT Conference Proceedings* (Vol. 12, No. 21). CS & IT Conference Proceedings.

[4] Ezzini, S., Abualhaija, S., Arora, C., Sabetzadeh, M., & Briand, L. C. (2021, May). Using domain-specific corpora for improved handling of ambiguity in requirements. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)* (pp. 1485-1497). IEEE.

[5] Ashfaq, F., Bajwa, I. S., Kazmi, R., Khan, A., & Ilyas, M. (2021). An intelligent analytics approach to minimize complexity in ambiguous software requirements. *Scientific Programming*, *2021*(1), 6616564.

[6] A. Fantechi, S. Gnesi, and L. Semini, ‘‘Rule-based NLP vs ChatGPT in ambiguity detection, a preliminary study,’’ in Proc. REFSQ Work shops,

2023, pp. 1–10. [Online]. Available: https://ceur-ws.org/Vol 3378/NLP4RE-paper1.pdf

[7] A. Ferrari, G. Lipari, S. Gnesi, and G. O. Spagnolo, ‘‘Pragmatic ambi guity detection in natural language requirements,’’ in Proc. IEEE 1st Int. Workshop Artif. Intell. Requirements Eng. (AIRE), Karlskrona, Sweden, Aug. 2014, pp. 1–8, doi: 10.1109/AIRE.2014.6894849.

[8] Sowmya, G., & Swapna, G. (2023). A Real Time Online Food Ordering Application Based Django Restful Framework. Industrial Engineering Journal, 52(9), 425–431.

[9] R. S. Satpute and A. Agrawal, ‘‘A critical study of pragmatic ambi guity detection in natural language requirements,’’ Int. J. Intell. Syst. Appl. Eng., vol. 11, no. 3, pp. 249–259, 2023. [Online]. Available: https://ijisae.org/index.php/IJISAE/article/view/2681

[10] U. S. Shah and D. C. Jinwala, ‘‘Resolving ambiguities in natural language software requirements,’’ ACM SIGSOFT Softw. Eng. Notes, vol. 40, no. 5, pp. 1–7, Sep. 2015, doi: 10.1145/2815021.2815032.

[11] E. Kamsties and B. Peach, ‘‘Taming ambiguity in natural language requirements,’’ in Proc. IEEE Joint Int. Conf. Requirements Eng., Dec. 2000, pp. 1–9.

[12] J. Matsuoka and Y. Lepage, ‘‘Ambiguity spotting using wordnet semantic similarity in support to recommended practice for software requirements specifications,’’ in Proc. 7th Int. Conf. Natural Language Process. Knowl. Eng., 2011, pp. 479–484.

[13] Sirisati, R. S., Viswanath, G., & Shaik, M. (2023, December). A Hybrid Particle Swarm Optimization and C4.5 for Network Intrusion Detection and Prevention System. In 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT) (pp. 518-522). IEEE. https://doi.org/10.1109/ICSSIT55814.2023.10060931

[14] E. Kamsties, ‘‘Understanding ambiguity in requirements engineering,’’ in Engineering and Managing Software Requirements, A. Aurum and C. Wohlin, Eds., Berlin, Germany: Springer, 2005, doi: 10.1007/3-540 28244-0_11.

[15] V.Gervasi, A. Ferrari, D. Zowghi, and P. Spoletini, ‘‘Ambiguity in require ments engineering: Towards a unifying framework,’’ in From Software Engineering to Formal Methods and Tools, and Back (Lecture Notes in Computer Science), vol. 11865, M. Ter Beek, A. Fantechi, and L. Semini, Eds., Cham, Switzerland: Springer, 2019, doi: 10.1007/978-3-030-30985 5_12.

[16] A. Ferrari and A. Esuli, ‘‘An NLP approach for cross-domain ambiguity detection in requirements engineering,’’ Automated Softw. Eng., vol. 26, no. 3, pp. 559–598, Sep. 2019, doi: 10.1007/ s10515-019-00261-7.

[17] F. Dalpiaz, I. V. D. Schalk, and G. Lucassen, ‘‘Pinpointing ambiguity and incompleteness in requirements engineering via information visual ization and NLP,’’ Requirements Eng., Found. Softw. Qual., vol. 10753, pp. 119–135, Jan. 2018.

[18] B.Gleich, O.Creighton, and L. Kof, ‘‘Ambiguity detection: Towards a tool explaining ambiguity sources,’’ in Requirements Engineering: Foundation for Software Quality (Lecture Notes in Computer Science), vol. 6182, R. Wieringa and A.

Persson, Eds., Berlin, Germany: Springer, 2010, doi: 10.1007/978-3-642-14192-8_20.

[19] H. Yang, A. de Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, ‘‘Analysing anaphoric ambiguity in natural language requirements,’’ Requirements Eng., vol. 16, no. 3, pp. 163–189, Sep. 2011. tools

[20] N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry, ‘‘Requirements for for natural ambiguity identification and measurement in language requirements specifications,’’ Requirements Eng., vol. 13, no. 3, pp.207–239, Sep. 2008, doi: 10.1007/ s00766-008-0063-7.

[21] Viswanath, G., Abirami, V., & Prathyusha, G. (2024). Hybrid Feature Extraction With Machine Learning To Identify Network Attacks. International Journal Of HRM And Organizational Behavior, 12(3), 217–228.

[22] K. Liu, S. Reddivari, and K. Reddivari, ‘‘Artificial intelligence in software requirements engineering: State-of-the-art,’’ in Proc. IEEE 23rd Int. Conf. Inf. Reuse Integr. Data Sci., Aug. 2022, pp. 106–111. [Online]. Available: ieeexplore.ieee.org/stamp/stamp.jsptp=&arnumber =9874229

[23] S.-C. Necula, F. Dumitriu, and V. Greavu-Serban, ‘‘A systematic literature review on using natural language processing in software requirements engineering,’’ Electronics, vol. 13, no. 11, p. 2055, May 2024, doi: 10.3390/electronics13112055.

[24] K. Zamani, D. Zowghi, and C. Arora, ‘‘Machine learning in requirements engineering: A mappingstudy,’’ in Proc. IEEE29thInt. RequirementsEng. Conf. Workshops (REW), Notre Dame, IN, USA, Sep. 2021, pp. 116–125, doi: 10.1109/REW53955.2021.00023.

[25] Kumar, C. S., Sirisati, R. S., Gudditti, V., Rao, K. S., & Challa, R. K. (2022, December). A smart recommendation system for medicine using intelligent NLP techniques. In 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS) (pp. 1081-1084). IEEE. https://doi.org/10.1109/ICACRS55517.2022.10029 078

[26] N. S. Thomas and S. Kaliraj, ‘‘An improved and optimized ran dom forest based approach to predict the software faults,’’ Social Netw. Comput. Sci., vol. 5, no. 5, p. 530, May 2024, doi: 10.1007/ s42979-024-02764-x.

[27] F. Nazir, W. H. Butt, M. W. Anwar, and M. A. K. Khattak, ‘‘The appli cations of natural language processing (NLP) for software requirement engineering—A systematic literature review,’’ in Information Science and Applications (Lecture Notes in Electrical Engineering), vol. 424, K. Kim and N. Joukov, Eds., Singapore: Springer, 2017, doi: 10.1007/978-981-10 4154-9_56.

[28] R. Izhar, K. Cosh, and S. N. Bhatti, ‘‘Enhancing agile software devel opment: A novel approach to automated requirements prioritization,’’ in Proc. 21st Int. Joint Conf. Comput. Sci. Softw. Eng. (JCSSE), Phuket, Thailand, Jun. 2024, pp. 286–293, doi: 10.1109/jcsse61278.2024. 10613648.

[29] R. Hanslo and M. Tanner, ‘‘Machine learning models to predict agile methodology adoption,’’ in Proc. 15th Conf. Comput. Sci. Inf. Syst., 2020, pp. 697–704. [Online]. Available: https://annals csis.org/proceedings/2020/drp/214.html

[30] W.Zhang, T. Yoshida, and X. Tang, ‘‘A comparative study of TF∗IDF, LSI and multi-words for text classification,’’ Exp. Syst. Appl., vol. 38, no. 3, pp.2758–2765, 2011. [Online]. Available:

https://www.
sciencedirect.com/science/article/pii/S0957417410
008626?via= ihub