Vol-2 Issue-2 2024
Scientific Research Journal of Science, Engineering and Technology
ISSN: 2584-0584, Peer Reviewed Journal

Classification of Extracted Features from Software Requirements Specification Documents using Support Vector Machine Learning Technique

Sadiq Mohammed Waziri[1], Fatima Umar Zambuk[2], Badamasi Imam Ya'u[3], Mustapha Abdurrahman Lawal[4]

[1,2,3,4] Department of Computer Science, Faculty of Science, Abubakar Tafawa Balewa University Bauchi, Bauchi State, Nigeria

Correspondence: Sadiq Mohammed Waziri
Email id: swaziri.pg@atbu.edu.ng

Abstract
This paper presents the results of an experiment on the classification of extracted features from Software Requirements Specification (SRS) documents using various Machine Learning techniques. The primary focus was on the linear Support Vector Machine (SVM) technique, with comparative analysis involving three additional techniques, namely Decision Tree (DT), Naïve Bayes (NB), and K-Nearest Neighbors (KNN). During the experimentation, features, which are fundamental building blocks of Software Product Lines [1], were classified into optional and mandatory. This differentiation facilitates both variability and similarity within a product family [1]. While previous research has explored similar classifications using diverse techniques, this study specifically identifies the most effective method for binary classification of features for feature modeling. We conducted the experiment on nine selected documents from the PURE dataset. We evaluated the performance of each model rigorously based on accuracy, precision, recall (sensitivity), and F1-score. The findings provide valuable insights into the optimal classification technique, enhancing the development and management of software product lines.

Keywords: Requirements Feature, Feature Extraction, Feature Classification, Feature Modeling, Support Vector Machine

1 Introduction

The demand for mass production of software products has driven research into automating the extraction [2] and classification of features [2] from various sources, such as conversational agents, codebases, programme documents, user manuals and software requirements specifications documents. This automation aims to enhance production speed and precision while maintaining high-quality deliverables. In Software Product Line Engineering (SPLE), feature extraction and classification are essential for feature modeling [1], which identifies variability and commonality within a Software Product Family (SPF) [1]. Effective feature modeling necessitates categorizing features into "mandatory" and "optional" classes. The "mandatory" class highlights commonalities within an SPF, while the "optional" class underscores variations, facilitating a binary classification.

This research paper presents the results of classifying extracted features from Software Requirements Specification (SRS) documents using Machine Learning techniques, with a primary focus on the Linear SVM. Additionally, it explores the effectiveness of DT, NB and KNN. The study aimed to determine the optimal method for binary classification of features for feature modeling, tested on nine selected documents from the PURE dataset. We evaluated each model's performance based on accuracy, precision, recall and F1-score.

We structured the paper into Section 1: Introduction; Section 2: Materials and Methods; Section 3: Results and Discussions, and Section 4: Conclusion. Other sections discussed include Conflict of Interest, Author Contributions, Funding, Acknowledgements, and Data Availability.

2 Materials and Methods

The existing system in focused on automating feature extraction for Software Product Lines (SPLs) from Software Requirements Specification (SRS) documents. It involved extracting SRS documents to create raw data, preprocessing them for noise removal, and preparing them for NLP analysis using SpaCy and a trained model. Techniques like sentencization, tokenization, and noun chunking were employed for NLP tasks. However, the existing system has limitations that include inability to classify the extracted features based on their relationships and the lack of a catalogue for creating Feature Models (FMs).

The proposed system aimed to enhance SPLE domain engineering by extending techniques for feature classification. It utilized Support Vector Machine (SVM) for classification, creating a catalogue for Feature Models. Features were classified as mandatory or optional based on relevance to functional requirements, with attributes like priority attached for additional context. Implementation involved using SVM and other classification techniques, particularly Decision Tree, Naïve Bayes and K-Nearest Neighbor for comparison. Python and SciKit Learn are used for implementation of the classification.

The evaluation metrics for the proposed model included accuracy, precision, recall, and F1-score. We calculated the metrics based on True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN). These metrics provided insight into the model's performance in binary classification tasks. The expected results included a feature catalogue with classified features, a complete system for feature extraction and classification, and improved efficiency and accuracy in feature model engineering.

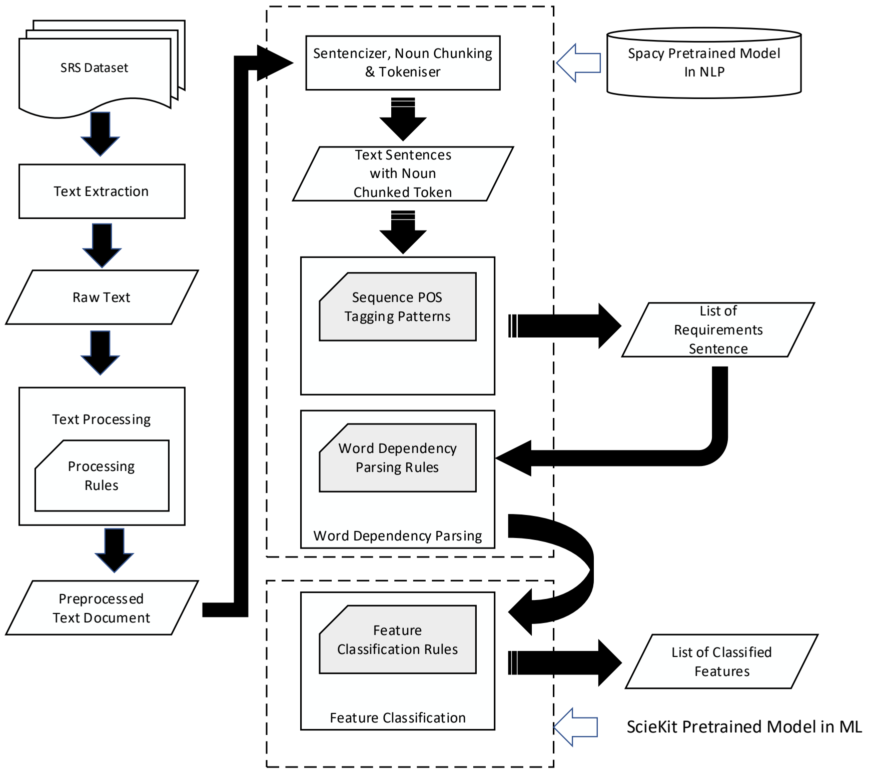
SRS Dataset
Text Extraction
Raw Text
Text Processing
Processing Rules
Preprocessed Text Document
Sentencizer, Noun Chunking & Tokeniser
Spacy Pretrained Model In NLP
Text Sentences with Noun Chunked Token
Sequence POS Tagging Patterns
List of Requirements Sentence
Word Dependency Parsing Rules
Word Dependency Parsing
Feature Classification Rules
Feature Classification
List of Classified Features
ScieKit Pretrained Model in ML

Figure 1: Architectural Design of the Proposed System

2.1 SVM Classification Technique

SVMs are a recent development in supervised machine learning, based on statistical learning theory [3]. SVMs are built around the concept of a margin that maximizes the distance between data points on either side of a separating hyperplane, thereby reducing predicted generalization error [4]. This margin is key to SVM's ability to categorize data optimally by their positions relative to this boundary.

In classification tasks, SVM is described as Support Vector Classifier (SVC) [5]. The formula $f(x) = (w * x) + b$ [6][7] gives the decision function for a linear SVC.

$f(x)$ = output,
w = weight vector,
x = feature vector
b = bias term.

Predictions are made based on the sign of $f(x)$.

Key concepts include:

- **Support Vectors:** Data points nearest to the hyperplane, critical for defining its position and orientation. [8][9]
- **Margin:** This is distance between the hyperplane and the nearest support vector, which SVM aims to maximize. [8][9][10]
- **C Parameter:** This manages trade-of between increasing the margin and reducing the classification errors. A smaller C will allow a larger margin with some misclassifications whereas a larger C reduces misclassifications at the expense of a smaller margin. [11]

SVMs are powerful popular tools for classification and regression [f], leveraging support vectors and margins to optimize performance. They handle complex data relationships effectively, especially when using techniques like the kernel trick for non-linear data.

3 Results and Discussions

Based on the results obtained in Table 1, performance of the proposed model is interpreted as follows: Precision measures the accuracy of positive predictions. In this case, the values of the *precision* are between 0.62 and 0.93. Higher precision indicates fewer false positives.

Accuracy is the ratio of the accurately predicted observations to the total of all observations. It is a measure the overall correctness of a model. Values of *accuracy* of the proposed model range from 0.75 to 0.95, which is generally good.

Recall measures the capability of the model to locate the total relevant cases within the given dataset. It is the ratio of accurately predicted positive observations to the total observations in actual class. The recall values range from 0.70 to 1.00.

Finally, the F1-Score considers both false positives and false negatives as it is the mean (weighted) of *precision* and *recall*. It balances precision and recall and the values range from 0.69 to 0.96.

Summarily, based on the results provided, the SVM classification technique seems to perform well, with high precision, accuracy, recall, and F1-Score values for most of the feature extraction from requirements documents (CCTNS, DH, LIS, Agentmom, Mashbot, Microwave). However, for other systems like e-Store, Puget, and Pontis, the performance is comparatively lower, especially in terms of recall and F1-Score; an indication that SVM may not be as effective for these requirements systems, and further investigation or optimization may be needed.

Table 1: **Requirements Sentence Extraction Result Comparison using NLP in sPacy**

No	SRS	Precision	Accuracy	Recall	F1-Score
1	CCTNS	0.93	0.92	1.00	0.96
2	DH	0.91	0.95	0.93	0.93
3	e-Store	0.91	0.90	0.70	0.73
4	LIS	0.81	0.90	0.90	0.85
5	Puget	0.62	0.79	0.79	0.69
6	Agentmom	0.80	0.80	0.80	0.71
7	Mashbot	0.89	0.82	0.88	0.90
8	Pontis	0.82	0.75	0.75	0.71
9	Microwave	0.89	0.90	0.94	0.91

Table 2. Success Rate for Feature Extraction

No	SRS	Extracted Requirement Sentences	Extracted Feature	Success Rate (%)
1	CCTNS	37	30	81.08
2	DH	33	27	81.82
3	e-Store	85	56	65.12
4	LIS	26	23	88.46
5	Puget	36	27	75.00
6	Agentmom	28	22	78.57
7	Mashbot	35	34	97.14
8	Pontis	135	80	59.26
9	Microcare	123	85	69.11

Table 2 showcases the results of a feature extraction process applied to Software Requirement Specifications (SRS) documents. By analyzing the data, we learnt about how effective feature extraction technique is. We also identified areas that need to be improved.

A key observation is the significant variation in success rates across different SRS documents. The percentage of requirement sentences from which a feature was successfully extracted ranges from as low as 59.26% (Pontis) to as high as 97.14% (Mashbot). This suggests that the effectiveness of the feature extraction process can differ depending on the specific SRS document it's applied to.

Several factors could contribute to these variations. The complexity of the language used in the SRS documents, the way requirements are structured, and the capabilities of the feature extraction algorithm itself all play a role. For instance, documents with highly technical language or poorly organized requirements might pose challenges for the feature extraction process.

Mashbot stands out with a very high success rate, indicating that the feature extraction process was highly effective in identifying features from its requirement sentences. This suggests that the SRS document for Mashbot might be well-structured and written in a way that aligns well with the feature extraction algorithm's capabilities.

Documents like Pontis and Microcare have significantly lower success rates. This implies that the feature extraction process struggled to extract features from a substantial portion of their requirement sentences. Further investigation into these specific cases might reveal areas where the algorithm can be improved or the SRS documents can be restructured for better feature extraction. This table emphasizes the importance of evaluating feature extraction techniques in a practical setting using real-world SRS documents. While the process appears successful for some documents, there's the need for improvement in others. By analyzing these results, developers can identify and address potential shortcomings in the feature extraction process, ensuring it effectively captures the essential features from their SRS documents.

In terms of classification of the features, the four models being compared (Figure 3) on the x-axis are SVM, Decision Tree, Naive Bayes, and KNN. Based on the gradings on the y-axis and heights of the bars, the SVM model appears to have the highest bars across all metrics, suggesting it is the best performing model for this particular text classification task.

The Decision Tree and KNN models have lower bars compared to SVM, indicating they might have performed a little worse on this task based on the metrics chosen, but it shows that they performed relatively better than the Naïve Bayes. The actual values of the comparison are shown in Table 4.

Table 3. Feature Classification Comparison

No	SRS	Accuracy	Precision	Recall	F1-Score
1	CCTNS	0.92	0.94	1.00	0.96
2	DH	0.95	0.93	0.93	0.90
3	e-Store	0.90	0.95	0.70	0.73
4	LIS	0.90	0.88	0.90	0.83
5	Puget	0.79	0.74	0.73	0.67
6	Agentmom	0.80	0.87	0.80	0.71
7	Mashbot	0.82	0.89	0.84	0.90
8	Pontis	0.75	0.88	0.69	0.71
9	Microwave	0.90	0.89	0.92	0.91

Table 4. Model Comparison

Model	Ave. Accuracy	Ave. Precision	Ave. Recall	Ave. F1-Score
SVM	0.86	0.89	0.83	0.86
DT	0.82	0.83	0.80	0.82

NB	0.80	0.82	0.78	0.79
KNN	0.81	0.82	0.79	0.81

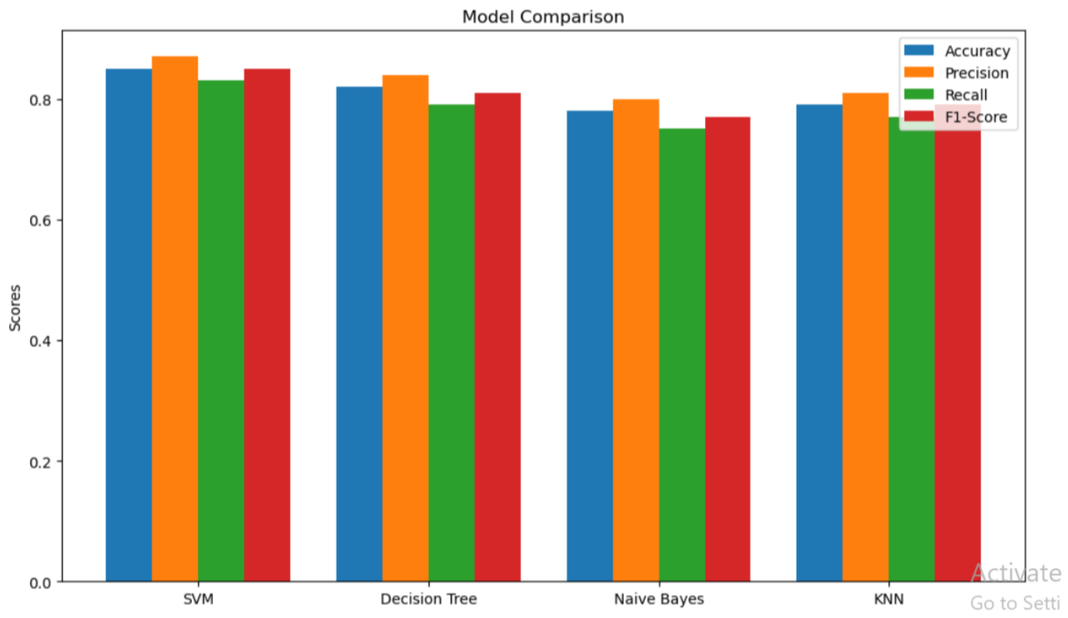

Figure 3: Graphical Comparison of the Models

Automating SPLE processes by extracting and classifying features, essential for modeling software product lines, is highly feasible, as demonstrated by the outcome of the study. Traditionally, this involves several steps: feature extraction, classification, feature modeling, and product configuration. SPLE is valuable for developing similar software products efficiently, but managing its processes, especially feature management, can be time-consuming and error-prone. Manual feature extraction and classification are prone to inconsistency and errors, particularly in complex product lines.

Automating these tasks offers a compelling solution. NLP tools were used to extract features from requirements documents using relevant keywords. SVM (and other machine learning algorithms) classified the features as mandatory or optional based on predefined rules, as shown by the system's performance in F1-scores. Classified features are used to build a feature catalogue, guiding the creation of feature models that capture relationships between features.

Once a feature model is in place, automation can extend to product configuration, generating valid product variants based on desired features and predefined constraints. This automation streamlines SPLE processes, as demonstrated by the success of this simple classification model. However, it's important to recognize limitations and adapt to the evolving nature of product lines. Combining automation with human expertise and continuous improvement can significantly enhance SPLE efficiency and effectiveness.

4 Conclusion

This study highlighted the potential of automating Software Product Line Engineering (SPLE) processes through feature extraction and classification. Automating these tasks can streamline SPLE, saving time and reducing errors. A key aspect of SPLE involves classifying features as mandatory (present in all products) or optional (included only in specific variants). The study demonstrated that SVM effectively performed binary classification. The advantage of a simple classification model, like the one used in the study, lies in its efficiency and ease of interpretation. However, for very complex product lines with numerous features and intricate relationships, more advanced models are necessary.

5 Conflict of Interest

There was no conflict of interest in financial, commercial, legal, or professional relationships with organizations or individuals.

6 Author Contributions

S. M. Waziri carried out the experiment, implementation, and manuscript writing. F. U. Zambuk and B. I. Ya'u supervised the project and contributed from the beginning to the end of the manuscript writing. Dr. M. A. Lawal is the departmental project coordinator who searches for and recommends the best organisations for publishing papers in journals. He also guides in formatting.

7 Funding

The Department of Computer Science, Abubakar Tafawa Balewa University, Nigeria approved this study. It also supported research; however, the funding was by the main author.

8 Acknowledgements

I acknowledge the Department of Computer Science, Abubakar Tafawa Balewa University, Bauchi, Nigeria, for support and guidance.

9 Data Availability

The dataset generated in this study is available in the PURE (Public Requirements) data pool at http://nlreqdataset.isti.cnr.it/.

10 References

[1] Pohl, K., Bockle, G., & van der Linden, F. (2005). Software Product Line Engineering: Foundations, Principles, and Techniques. Springer, Heidelberg, Berlin, Germany.

[2] Quba, G. Y., Al Qaisi, H., Althunibat, A.and AlZu'bi, S. (2021). Software Requirements Classification using Machine Learning Algorithms. *2021 International Conference on Information Technology (ICIT), Amman, Jordan, 2021, pp. 685-690, doi: 10.1109/ICIT52682.2021.9491688.*

[3] Jakkula, V. ().Tutorial on Support Vector Machine (SVM). School of EECS, Washington State University, Pullman 99164. [Unpublished]

[4] Müller, A. C. and Guido, S. (2002). Introduction to Machine Learning with Python: A Guide for Data Scientists. Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

[5] Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.
[6] Shawe-Taylor, J., & Cristianini, N. (2000). An Introduction to Support Vector Machines : and other Kernel-Based Learning Methods. Cambridge: Cambridge University Press.

[7] Mavroforakis, M & Theodoridis, S (2020). A Geometric Approach to Support Vector Machine(SVM) Classification. EEE Transactions on Neural Networks, Vol. 17, No. 3, May 2006. DOI: 10.1109/TNN.2006.873281

[8] Awad, M., Khanna, R. (2015). Support Vector Machines for Classification. In: Efficient Learning Machines. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4302-5990-9_3

[9] Alpaydin, E. (2014). Introduction to Machine Learning. The MIT Press Cambridge, Massachusetts, London, England.

[10] Canedo, E. D. & Mendes, B. C. (2020). Software Requirements Classification using Machine Learning Algorithms. Entropy. DOI: 10.3390/e22091057

[11] Cunha, V.C., Magoni, D., Inácio, P.R.M., Freire, M.M. (2022). Impact of Self C Parameter on SVM-based Classification of Encrypted Multimedia Peer-to-Peer Traffic. In: Barolli, L., Hussain, F., Enokido, T. (eds) Advanced Information Networking and Applications. AINA 2022. Lecture Notes in Networks and Systems, vol 449. Springer, Cham. https://doi.org/10.1007/978-3-030-99584-3_16.