e-Informatica Software Engineering Journal, Volume 15, Issue 1, 2021, pages: 85–114, DOI 10.37190/e-Inf210105

Mining Non-Functional Requirements using Machine Learning Techniques

Rajni Jindal*, Ruchika Malhotra**, Abha Jain***, Ankita Bansal****
*Dept. of Computer Science Engineering, Delhi Technological University
**Dept. of Software Engineering, Delhi Technological University
***Dept. of Computer Science, Delhi University
****Division of Information Technology, Netaji Subhas University of Technology
rajnijindal@dce.ac.in, ruchikamalhotra2004@yahoo.com, abhajain.src@gmail.com, ankita.bansal06@gmail.com

Abstract

Background: Non-Functional Requirements (NFR) have a direct impact on the architecture of the system, thus it is essential to identify NFRs in the initial phases of software development. **Aim**: The work is based on extraction of relevant keywords from NFR descriptions by employing text mining steps and thereafter classifying these descriptions into one of the nine types of NFRs. **Method**: For each NFR type, keywords are extracted from a set of pre-categorized specifications using Information-Gain measure. Then models using 8 Machine Learning (ML) techniques are developed for classification of NFR descriptions. A set of 15 projects (containing 326 NFR descriptions) developed by MS students at DePaul University are used to evaluate the models.
Results: The study analyzes the performance of ML models in terms of classification and misclassification rate to determine the best model for predicting each type NFR descriptions. The Naïve Bayes model has performed best in predicting "maintainability" and "availability" type of NFRs.
Conclusion: The NFR descriptions should be analyzed and mapped into their corresponding NFR types during the initial phases. The authors conducted cost benefit analysis to appreciate the advantage of using the proposed models.

Keywords: requirement engineering, text mining, non-functional requirements, machine learning, receiver operating characteristics

1. Introduction

Non-Functional Requirements (NFRs) are the basic quality constraints which specify the operation of a system [1, 2]. NFRs go hand-in-hand with the Functional Requirements (FRs) and are highly essential to ensure the development of an efficient and a reliable software system that meets the customers needs and fulfills their expectations [3]. Set of NFRs need to be correctly identified in the initial phases of Software Development Lifecycle (SDLC) process as they play a crucial role in the architecture and design of the system which in turn affects the quality of the system [4]. NFRs form the basis for architects to create the technical architecture of the system. This architecture of the system then acts a platform in which the functionality of the system is delivered [5]. Unfortunately, NFRs are discovered in the later phases of software development. This may be due the reason that some requirement engineers tend to overlook NFRs failing to realize their importance and thereby assume them to be implicitly understood [6]. They do not elicit NFRs in the Software Requirement Specification (SRS) document as clearly as they state FRs, but rather state NFRs in a very adhoc and random fashion due to which the final SRS document is organized by functionality with non-functional requirements scattered throughout the document

Submitted: 27 February 2020; Revised: 08 January 2021; Accepted: 19 February 2021; Available online: 28 June 2021

[7, 8]. Failure to identify and analyze NFRs in the early phases can result in unclassified, incomplete or conflicting NFRs, requiring costly rework in later stages of the software development [5]. The work in this paper focuses on mining the descriptions of NFRs which are stated throughout the requirement specification document in an adhoc and random fashion and thereafter classify them into one of the types of NFRs using a suitable Machine Learning (ML) technique. In this work, nine types of NFRs have been considered viz. Availability (A), Look-and-Feel (LF), Legal (L), Maintainability (MN), Operational (O), Performance (PE), Scalability (SC), Security (SE) and Usability (US). The work is based on extraction of relevant keywords from NFR descriptions by employing a series of text mining steps. Firstly, the NFR descriptions are pre-processed to remove irrelevant words from the descriptions. These are the stop words like prepositions, articles, conjunctions, verbs, pronouns, nouns, adjectives and adverbs whose presence will not have any impact on the performance of the prediction models but will rather degrade their performance. Once pre-processing is done, we need to find the words which are relevant in describing the NFR descriptions. One of the methods is to extract the relevant words using fixed set of pre-defined keywords available in the catalogues. These catalogues contain a standardized set of keywords specific to different types of NFRs. However, the main problem of using this approach (standard keyword method) is the difficulty of finding accepted and standardized catalogues for all the types of NFR specified in the datasets [9]. Even for the NFR types whose catalogues were available, it was observed that the keywords specified in the catalogue for that particular NFR type could not classify the NFR descriptions pertaining to that NFR type accurately. This was observed by the authors Cleland-Huang et al. [9] who concluded that classification of NFRs based the keywords extracted from the catalogues result in unclassified, incomplete or conflicting NFRs.

In order to address this problem, the authors in this paper are using a training set to discover a set of keywords specific to each type of NFR. Since the keywords in our study are extracted from pre-categorized requirement specifications, therefore problem of finding accepted and standardized catalogues for all the NFR types is removed.

These keywords are the weighted indicator terms (specific to each NFR type) which are extracted using Information-Gain (IG) measure. IG measure extracts those keywords (also known as indicator terms) from the specifications that most accurately identify the target concept (NFR type). For example, terms such as "authenticate" and "access" represent strong indicator terms for security NFRs as they most accurately define security requirements as compared to other types of requirements. IG measure works by associating a weight to each of the word obtained after pre-processing and then the top-N scoring words are selected as the indicator terms. In our work, the value of N is considered as 10 as the paper by Cleland-Huang et al. [9] showed that good results were achieved when top-10 words were considered as compared to the results achieved when top-5 words or all the words were considered together. Once the indicator terms for each NFR type are identified, prediction models are developed for the classification of future NFR descriptions whose NFR type is not known. The authors in this paper have used in total eight ML algorithms viz. J48 decision tree (J48), Random Forest (RF), Logitboost (LB), Adaboost (AB), Multi-Layer Perceptron (MLP), Radial Basis Function network (RBF), Bagging, Naïve Bayes (NB) for predicting each type of NFR. The usage of large number of ML algorithms allows the authors to perform exhaustive comparison and evaluation. The authors concluded the best ML model for each NFR type which can be used by software practitioners and researchers in classifying an unknown NFR description into its respective type. In addition to this, the authors have also conducted cost benefit analysis to understand and appreciate the advantage of using the proposed models in contrast to not using the models in terms of cost incurred in both the cases.

Thus, there are three main goals of this study which we thrive to achieve:

1. To apply text mining steps to identify indicator terms As discussed, top-10 indicator terms for each NFR type are identified by following

a series of text mining steps. This begins by applying pre-processing steps to remove irrelevant words from the specifications and thereafter applying IG measure to retrieve significant words. IG measure works by associating a weight to each of the word obtained after pre-processing and then the top-N scoring words are selected as the indicator terms specific to each NFR type. These indicator terms serve as independent variables used for model prediction.

2. To develop machine learning models for each type of NFR Corresponding to each NFR type, a separate dataset was considered with top-10 indicator terms of that particular NFR type as independent variables and a binary dependent variable with the value of 1 (if NFR description belongs to that particular NFR type) or 0 (if NFR description does belong to that particular NFR type). Eight ML algorithms are applied to each dataset and therefore eight different prediction models were considered for each NFR type.
3. To conduct cost-benefit analysis Once the prediction models are developed, it is very important to analyze their benefits in terms of the cost incurred. Thus, a cost-benefit analysis is conducted to understand and appreciate the advantage of using the proposed models in contrast to not using the models in terms of cost incurred in both the cases.

Empirical analysis is conducted using an open source PROMISE dataset freely available at http://promisedata.org/repository. A set of 15 projects made by MS students at DePaul University that consisted of a total of 326 NFR descriptions categorized into nine types of NFR ("A", "LF", "L", "MN", "O", "PE", "SC", "SE", "US") were used to evaluate the results. The performance of the ML classifiers was compared and evaluated to find out the ML classifier that best predicted the NFR type using the measures derived from Receiver Operating Characteristic (ROC) analysis viz. Area Under the ROC Curve (AUC) and recall. The results indicated the performance of NB classifier has been best in predicting all the NFRs except SE. Also from cost-benefit analysis, it was concluded that the cost incurred without using our proposed models is more than the cost incurred when using the proposed models. Thus, we suggest that any professional from the industry who would use our models for classifying the NFRs into their types would be in profit.

The paper includes in total eight sections. The current literature has been provided in Section 2. Section 3 explains the background of the research. The methodology behind the research has been highlighted in Section 4. The result analysis is presented in Section 5 and Section 6 provides the discussion of the results in terms of cost/benefit analysis and how the work can be useful for the industry practitioners. Section 7 provides threats to validity. Finally, the paper is concluded in Section 8 highlighting the future work.

2. Related work

The area of NFR classification is an emerging area wherein a lot of research is still being carried out. Different authors have employed different techniques and methodologies in order to classify the descriptions of NFRs into their respective types. Table 1 summarizes the work done in the area of NFR classification with respect to types of NFR into which NFR descriptions have been categorized, ML technique used to perform NFR classification, Natural Language Processing (NLP) techniques used to pre-process the data and the dataset used for conducting empirical validation. (The abbreviations and their full forms used in Table 1 are – NB: Naïve Bayes; DT: Decision Tree; SVM: Support Vector Machine; MNB: Multinomial Naïve Bayes; EM: Expectation Maximisation; DMNB: Discriminative MNB, LDA: Latent Dirichlet Allocation; KNN: K-Nearest Neighbor; RB: Rule-Based; HAC: Hierarchical Agglomerative Clustering; ET: Extra Tree; LR: Logistic Regression; NFR: Non-Functional Requirement; FR: Functional Requirement; A: Availability; LF: Look-and-Feel; L: Legal; MN: Maintainability; O: Operational; PE: Performance; SC: Scalability; SE: Security; US: Usability).

Till now, only a few authors have explored this area and an efficient utilization of resources

and manpower is required to devise new methodologies and techniques for classifying the NFRs. The primary work in this area has been doneby the authors Cleland-Huang et al. [9]. The authors have used NFR-classifier which is based on information retrieval approach. The method is based on characterizing the different types of NFR using the concept of keywords. These keywords are used to detect requirements pertaining to a particular type of NFR. The work by the authors Hussain et al. [10] extended the idea of Cleland-Huang et al. and showed that the usage of linguistic knowledge is very helpful in the classification. The work incorporates the usage of a part-of-speech (POS) tagger. Apart from this, the authors Gokyer et al. [11] have used SVM algorithm in order to relate NFRs in a textual data to the quality determining attributes. This was accomplished with the help of a knowledge base provided by an expert. Rashwan et al. [12] too used SVM algorithm for automatic categorization of sentences into different classes based on the field of ontology. Similar work was done in the paper [13] that extracted NFRs from textual documents and used the extracted requirements to improve the descriptions of NFRs supporting other Requirement Engineering (RE) tasks. The papers by the authors Casamayor et al. [14] performed classification using a semi-supervised learning approach which is based on a lesser number of requirements in contrast to the supervised approach. The authors proposed a recommender system based on Multinominal Naïve Bayes classifier in combination with Expectation-Maximization (EM) technique. Zhang et al. [15] incorporated a SVM classifier and repeated the experiments of Cleland-Huang et al. [9] again in 2011. They have reported comparatively higher results of precision, although lower results of recall than Cleland-Huang et al. [9]. They have shown that a model based on the individual words outperformed the models based on multi-words. The paper by Slankas et al. [16] utilized a K-nearest neighbor (KNN) supervised learning algorithm for performing NFR classification and compared its performance with SVM and Naïve Bayes techniques. It was observed that SVM algorithm performed better than Multinomial Naïve Bayes classifier and KNN classifier outperformed optimal Naïve Bayes classifier with a unique distance metric. The authors Singh et al. [17] have incorporated rule-based classification technique in order to identify and classify the requirement sentences into NFR sub-classes using the concept of thematic roles. PROMISE corpus and Concordia corpus have been used to validate the results. The authors Kurtanovic et al. [18] studied how accurately the classification of requirements as FRs and NFRs can be done with supervised ML approach incorporating meta-data, lexical, and syntactical features. Similar work was done by the authors in their paper [19–24] which aimed at identifying NFRs in the informal documents like user comments, commit messages and installation manuals. Apart from this, few authors [25–28] have primarily worked on the extraction and classification of only the security requirements as these were considered the most significant type of NFR essential for the development of secure and reliable software.

The work in this paper is based on extraction of relevant keywords from NFR descriptions by employing a series of text mining steps and thereafter classify them into one of the nine types of NFRs (“A”, “LF”, “L”, “MN”, “O”, “PE”, “SC”, “SE”, “US”) using a suitable ML based prediction model. These keywords (also known as indicator terms) are extracted for each NFR type using IG measure. Once the indicator terms for each NFR type are identified, prediction models are developed for the classification of future NFR descriptions whose NFR type is not known. In this study, eight different prediction models (corresponding to eight different ML techniques) were developed for each type of NFR. Since there were nine types of NFR considered in this work, therefore a total of 72 prediction models were developed. An extensive evaluation and comparison of these 72 models was performed with the aim to identify best prediction model for each NFR type that could accurately classify future NFR descriptions whose NFR type is not known. The literature shows (Table 1) that majority of the studies have worked on very few classifiers (maybe two or three). Most of the studies have used SVM and

NB classifiers for developing prediction models. The usage of large number of ML techniques allows us to provide a fair evaluation and conclude the best prediction model that could most accurately classify each type of NFR description. Once we have found the best prediction model corresponding to each type of NFR, we have also performed cost-benefit analysis. This analysis was done to understand and appreciate the cost of using the proposed models vis-à-vis the cost of not using the models. This analysis is very important from industry point of view when the models are required to be used in real sense. This analysis is also missing in majority of the studies.

Table 1: Summary of the studies pertaining to NFR classification

S. No.	Paper	Types of NFR used	ML techniques used	NLP technique used	Dataset used
1	[9]	A, LF, L, MN, O, PE, SC, SE, US	NFR-Classifier	Stemming, Stop-words removal, tokenization	Promise NFR dataset (http://promisedata.org/repository) created by students of DePaul University. It contains 15 projects consisting of a total of 684 requirements specifications (326 NFR + 358 FR).
2	[10]	Not provided	DT	Stemming, POS, tokenization	Promise NFR dataset created by students of DePaul University. It contains 15 projects consisting of a total of 765 requirements specifications (495 NFR + 270 FR).
3	[11]	PE, MN, US, Integrity, Portability, Deployability, Dependability	SVM	Stemming, POS	Web-based transactional applications implemented at Cybersoft.
4	[14]	A, LF, L, MN, O, PE, SC, SE, US, Portability, Fitness, Functionality	MNB coupled with EM	Stemming, Stop-words removal, Normalization	Promise NFR dataset (http://promisedata.org/repository) created by students of DePaul University. It contains 15 projects consisting of a total of 625 requirements specifications (370 NFR + 255 FR).
5	[15]	SE, PE, A, SC, MN, L, US, O, LF, Palm Operational, Fitness	SVM	Stemming, POS, *N*-gram, Regular Expression tokenization	Promise NFR dataset (http://promisedata.org/repository) created by students of DePaul University. It contains 15 projects consisting of a total of 625 requirements specifications (370 NFR + 255 FR).
6	[25]	Security requirements, security-relevant sentences and security-related sentences	NB		Three industrial requirements documents viz. Common Electronic Purse Specification (ePurse), Customer Premises Network specification (CPN) and Global Platform Specification (GP). Total number of requirements is 124, 210, 176 for ePurse, CPN, GP with number of security requirements being 83, 42, 63 for ePurse, CPN, GP.
7	[19]	SE, MN, PE, US, Integrity, Portability, Efficiency, Reliability,	SVM, NB, MNB, DMNB, LDA	Stop-words removal	Three different open-source, partially-commercial database systems: (1) MySQL 3.23: contains 320 KLOC of C and C++ source code. Its source control history was used from 31 July 2000–9 August 2004. (2) MaxDB 7.500:

Table 1 continued

S. No.	Paper	Types of NFR used	ML techniques used	NLP technique used	Dataset used
		Interoperability, Testability, Traceability, Accuracy, Modifiability, Modularity, Correctness, Verifiability, Functionality, Understandability, Flexibility			contains 940 KLOC. Its source control history was used from 29 June 2004–19 June 2006. (3) PostgreSQL 7.3: contains 306 KLOC of C code. Its source control history was used from 9 May 2002–26 August 2004.
8	[12]	SE, US, Efficiency, Functionality, Reliability	SVM	Stemming, tokenization	(1) Promise NFR dataset (http://promisedata.org/repository) created by students of DePaul University, contains 15 projects consisting of a total of 684 requirements specifications (326 NFR + 358 FR). (2) A manually annotatedcorpus containing 6 types of requirement documents, 4 are SRSs of different products (online shopping center, student management system, institute of space physics, hospital patient system), 1 supplementary specification document, and 1 use case document. These documents contain 3064 sentences, manually annotated in four main classes (FR, External and Internal Quality, Constraints and other NFRs).
9	[16]	A, SE, L, LF, MN, O, US, Access Control, Audit, Privacy, Capacity, Performance, Recoverability, Reliability	SVM, KNN	Stop-words removal, POS, Lemmatization, Dependency parsing	A series of 11 documents related to Electronic Health Records (EHRs) (https://github.com/RealsearchGroup/NFRLocator). For requirement specifications, CCHIT Ambulatory Requirements, iTrust, and the PROMISE NFR Data Set (http://promisedata.googlecode.com) were used.
10	[20]	Not provided	SVM, MNB	N-gram	2 specifications from Mercedes-Benz (automotive industry).
11	[26]	Confidentiality, Integrity, Identification, Authentication, Accountability, Privacy Availability	SVM, NB, KNN	tokenization	10,963 sentences in six different documents from healthcare domain.
12	[29]	Not provided	SVM	N-gram	Specifications from Mercedes-Benz (automotive industry).
13	[13]	A, LF, L, MN, O, PE, SC, SE, US, Portability, Fitness, Functionality	RB	Stop-words removal, Lemmatization, Dependency parsing	Promise NFR dataset created by students of DePaul University, contains 15 projects consisting of a total of 625 requirements specifications (370 NFR + 255 FR).

Table 1 continued

S. No.	Paper	Types of NFR used	ML techniques used	NLP technique used	Dataset used
14	[30]	SE, PE, SC, US, Reliability	KNN	Not provided	2 case studies: 1st case study utilized the Predictor Models in Software Engineering (PROMISE) dataset, 2nd case study utilized the European Union eProcurement System"s 26 FRs.
15	[31]	SE, PE, L, A, Safety, Privacy, Accuracy, Portability, Reliability, Interoperability, Accessibility	K-means clustering, HAC	Stemming, Stop-words removal, Lemmatization	Three experimental software Java systems from different application domains viz. SmartTrip (an Android mobile application), SafeDrink with a mobile application interface, BlueWallet (subscription-based Web service)
16	[27]	Security requirements of type Authentication, Authorization, Access control, Cryptography-Encryption, Data integrity	DT	Stemming, Stop-words removal, tokenization	Promise NFR dataset (http://promisedata.org/repository) created by students of DePaul University. It contains 15 projects consisting of a total of 684 requirements specifications (326 NFR + 358 FR). Out of 326 NFR specifications, the total number of security requirement specifications is 58.
17	[28]	Security requirements of type Authentication, Authorization, Access control, Cryptography-Encryption, Data integrity	DT	Stemming, Stop-words removal, tokenization	Promise NFR dataset (http://promisedata.org/repository) created by students of DePaul University. It contains 15 projects consisting of a total of 684 requirements specifications (326 NFR + 358 FR). Out of 326 NFR specifications, the total number of security requirement specifications is 58.
18	[17]	Efficiency (Time behavior, Resource Utilization), Functionality (Suitability, Accuracy, SE), US (Operability, Understandability, Attractiveness)	RB	Stemming, POS, tokenization	(1) Promise NFR dataset (http://promisedata.org/repository) created by students of DePaul University. It contains 15 projects consisting of a total of 635 requirements specifications (370 NFR + 265 FR). (2) A manually annotated corpus containing 6 types of requirement documents, 4 are SRSs of different products (online shopping center, student management system, institute of space physics, hospital These documents contain 3064 sentences, manually annotated in four main classes (FR, External and Internal Quality, Constraints and other NFRs).
19	[21]	Reliability, Portability, PE, US	NB, DT, Bagging	Stemming, Stop-words removal, Lemmatization	Two popular Apps viz. Apple App (iBooks in the books category) and Google Play (WhatsApp in the communication category). Total 21969 raw user reviews (6696 FRom iBooks and 4400 FRom WhatsApp) were obtained.

Table 1 continued

S. No.	Paper	Types of NFR used	ML techniques used	NLP technique used	Dataset used
20	[22]	PE, US, Reliability, Supportability, Functionality	SVM, NB, DT, KNN	Stemming	40 Apps from the App Store falling into 10 categories (books, education, games, health, lifestyle, navigation, news, productivity, travel and utilities). A total of 932,388 reviews were obtained.
21	[23]	Reliability, PE, Lifecycle, Capability Usability, System Interface	SVM, KNN	Lemmatization	User requests of open source projects from sourceforge.net, whose user base consists of both software developers and ordinary software consumers.
22	[18]	PE, US, O, SE	SVM	Stop-words removal, POS, Lemmatization, N-gram	NFR dataset consisting of a total of 625 requirements specifications (370 NFR + 255 FR).
23	[32]	A, MN, US, LF, PE, SC, Operability, Fault Tolerance, Portability, Legal and Licensing	MNB, LDA, K-means, HAC	POS, Regular Expression, Entity Tagging, Temporal Tagging	TERA Promise NFR dataset created by students of DePaul University and was updated in 2010. It contains 15 projects consisting of a total of 625 requirements specifications (370 NFR + 255 FR).
24	[24]	MN, US, Reliability, Efficiency, Portability, Functionality	LDA	Stop-words removal, Normalization	Extracted posts (21.7 million) and comments (32.5 million) of the Stack Overflow from July 31, 2008 to September 14, 2014 provided by the MSR (Mining Software Repositories) challenge.
25	[33]	A, L, LF, MN, O, PE, SC, SE, US, Fault tolerance, Portability	Multinomial NB, Bernoulli NB, Gaussian NB, DT, ET, ETs, KNN, Linear LR, MLP, SVM, Label Propagation, Label Spread	Stemming, Stop-words removal, tokenization	TERA Promise NFR dataset created by students of DePaul University and was updated in 2010. It contains 15 projects consisting of a total of 625 requirements specifications (370 NFR + 255 FR).

3. Background of the research

This section includes a brief overview of the ML techniques used for classification along with the description of performance evaluation measures used for evaluating the performance of the prediction models.

3.1. Overview of machine learning techniques

From the literature survey (Table 1) it was observed that majority of the authors have incorporated ML techniques for NFR classification which fall under supervised learning approaches as compared to the techniques which fall under unsupervised learning approaches or semi-supervised learning approaches. Supervised learning ML techniques have accurately performed NFR classification producing promising results. Keeping this in mind, we intended to work on supervised ML techniques. The techniques under supervised learning can be broadly categorized under the following domains: Ensemble Learners, Neural Networks, Bayesian Networks, and Decision Tree. To have a fair evaluation of techniques under all the domains, we selected 1 to 3 ML techniques under each of the domains. In total, we compared and contrasted eight different ML techniques, viz. RF, LB, AB, Bagging (Ensemble learner), MLP, RBF (Neural network), NB (Bayesian network) and J48 (Decision tree). These techniques are popularly used for binary classification in other fields such medical diagnosis [34, 35] network intrusion detection [36], credit card fraud detection [37], defect and change prediction [38, 39], etc. and have shown promising results. Thus, the authors want to explore them for identifying the type of NFR description. A brief overview of these ML techniques is presented in Table 2. These ML classifiers are implemented using the default control settings of an open source tool, WEKA http://www.cs.waikato.ac.nz/ml/weka/. The default control parameters for each of the ML classifier are also provided in Table 2. We have used the default settings and have not tuned the parameters as over-fitting of the parameters may become a threat to external validity.

3.2. Measures for evaluating models

Once the models are trained, testing is performed to evaluate the performance of the models. The performance of the models would be highly optimistic if the testing is performed on the same dataset as the one on which training is performed. Hence, we have used intra-validation technique where the same dataset is partitioned into two subsets, one of which is used for training, while the other is used for testing. The intra-validation technique used in this study is 10-cross validation technique wherein the entire dataset (326 NFR descriptions) is partitioned into 10 equally sized parts (P1, P2,. . . , P10). As can be seen from the Figure 1, for the first time, one part is used for testing (P1), while remaining 9 parts (P2–P10) are used for training the model [48]. The procedure is continued 10 times such that each instance gets validated once and finally a single estimate is produced by combining all the 10 results [49].

Figure 1. Procedure of 10-fold cross validation technique

For evaluating the performance of the models, we need appropriate performance measures which are suitable to be applied in the given study. In this study, amongst a number of measures available, the authors choose to use two performance measures, recall (also known as recall) and the area under the ROC curve (AUC). The rationale behind using these performance measures has been explained below. Recall is one of the traditional measures which tells that out of actual positive data, how many times the model predicted correctly [50]. Another commonly used traditional measure which is not used in this

Table 2. Description of machine learning techniques and control parameters used in the study

ML Technique	Description	Parameter Settings
J48 decision tree [40, 41]	It is an implementation of C4.5 decision tree algorithm used to handle the classification problems. It generates a binary tree which could be either pruned or unpruned. The pruned tree does not have an influence on the performance of the model while discarding the nodes and branches. It also reduces the risk of overfitting to the training data.	Set confidence factor as 0.25. Set minimum number of objects in leaves as 2 and number of folds as 3. Set seed as 1.
Random forest [42]	It is used for building a number of classification trees, thereby leading to a forest. Each object which needs to be classified acts as an input to the tree in the forest. The process of classification is performed by each tree, and it is said that the tree "votes" for that particular class.	Set the classifier as Decision Tree. Set maximum depth as 0, number of features as 0, seed as 1 and number of trees as 100.
Bagging [42]	In this technique, a sample of data is used to generate various sub-samples of the data which are the training sets and used for making the required predictions by developing the desired model.	Set the classifier as Decision Stump or REPTree. Set number of iterations as 10, weight threshold as 100 and seed as 1. No resampling is used.
Logit-boost [43]	In this technique, the regression technique is used as the base learner and this is followed by performing additive logistic regression. It is the most important type of the Boosting technique.	Set the classifier as Decision Stump. Set number of iterations as 10, number of runs and seeds as 1 and weight threshold as 100. No resampling is used.
Ada-boost [43]	This technique is based on combining different weak learning techniques in order to improve the process of classification, leading to improved results. This is done by first assigning equal weights to all the instances present in the training set and thereafter multiple rounds are conducted and in each round the weights of the examples which have not been correctly classified are increased. This is how the performance of a weak learner is improved.	Set the classifier as Decision Stump. Set number of iterations as 10, number of runs and seeds as 1 and weight threshold as 100. No resampling is used.
Multi-Layer Perceptron [44]	In this technique, a set of input values are mapped to a set of output values wherein learning is done using back-propagation. Firstly the inputs are given to the network. Using the weights applied on each layer and the inputs, the desired output of the network is calculated. Then the error is computed which is difference of the actual value of the output and the calculated value. Based on this computed error value, the weights are updated and accordingly the parameters of the network are adjusted. To achieve the desired performance, this process is repeated again and again.	Set number of hidden layers as a wildcard value "a" = (no. of attributes + no. of classes)/2. Set learning rate as 0.3 and momentum as 0.2. Set sigmoid function for transfer.
Radial Basis Function network [45]	An artificial neural network having a single layer is called RBF network. The activation function used here is the radial functions which are applied to the inputs. These inputs are combined with the weights to produce the desired output of the network.	Set number of clusters as 2 and clustering seed as 1.
Naïve Bayes [46, 47]	This is one of the simplest classifier based on probability wherein the approach for classification is based on Bayes" theorem. The most probable class for a new instance is found out using this technique. A parametric model is used to generate the test data. The Bayes" estimates for the model parameters are calculated using the training data.	Set kernel estimator as false. Set supervised discretization as false.

study is precision. Precision tells that when the model predicts something positive, how many times they were actually positive. Mathematically, recall and precision are defined as follows:

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

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

where, *TP* (True Positive): When the document actually belongs to a category "A" (positive) and is predicted by the model to be in category "A" (positive), *FN* (False Negative): When the document actually belongs to a category "A" (positive) and is predicted by the model in category "Not A" (negative), *FP* (False Positive): When the document actually belongs to a category "Not A" (negative) and is predicted by the model to be in category "A". We have focused on recall and did not use precision because of the following two reasons:

1. In this study, *FN* is more significant than FP and as a result *FN* cannot be ignored. If we cannot ignore *FN*, then we have to take into account recall measure. Let us understand why recall is more important than precision in this study. *FN* occurs when a document is predicted by a model to be in category "Not A" when it actually belongs to category "A". *FP* occurs when a document is predicted by the model to be in category "A", when it actually belongs to category "Not A". Now when a *FN* occurs, a document which actually belongs to category "A" is ignored by the stakeholders of the software because it is predicted to be in category "Not A". This may result in delivery of poor quality software and may have serious implications on the industry in terms of its reputation in long term. However, when a *FP* occurs, some extra resources of the industry (in terms of time, money and manpower) may be utilized as the document actually belongs to a category "Not A" (negative) and is predicted by the model to be in category "A". Clearly, *FN* holds a more significant position as releasing poor quality software is more disastrous as compared to utilization of some extra resources. Thus, in this study, we have reported recall and did not consider precision.
2. Moreover, the datasets (9 datasets with binary dependent variable formed from a single dataset) used in this work for model prediction are imbalanced where the number of instances belonging to negative class is more than the number of instances belonging to the positive class. Studies in literature have criticized the use of precision when the models are validated using imbalanced datasets [51, 52].

Instead, to handle the imbalance nature of the datasets, we have used an effective performance measure known as Area Under ROC Curve (AUC). ROC curve is a plot of recall (true-positive) on the y-coordinate versus its 1-specificity (false positive) on the x-coordinate. It is used to measure the accuracy of the model and its values lie in the range of 0 to 1, where an AUC value of 1 indicates the best prediction. When the data is imbalanced, the model is biased towards the majority class while the minority class is predicted with less accuracy. To handle this, studies [51, 53, 54] propose the use of AUC as it is insensitive to class distribution changes. In other words, it is robust to imprecise class distribution and misclassification costs [55].

4. Methodology behind the work

This section discusses the methodology incorporated to accomplish the classification of NFRs. Figure 2 depicts three steps used to develop the predictive models. The steps have been explained in the following subsections.

4.1. Gathering of NFR descriptions

The dataset used in this work for empirical analysis was the same dataset which was used by the authors Cleland-Huang et al. [9]. This dataset consist of a set of 15 projects which were made by MS students at DePaul University available at the open source PROMISE software engineering repository http://promisedata.org/repository. These projects contained a total of 326 NFR descriptions and 358 FR descriptions. These

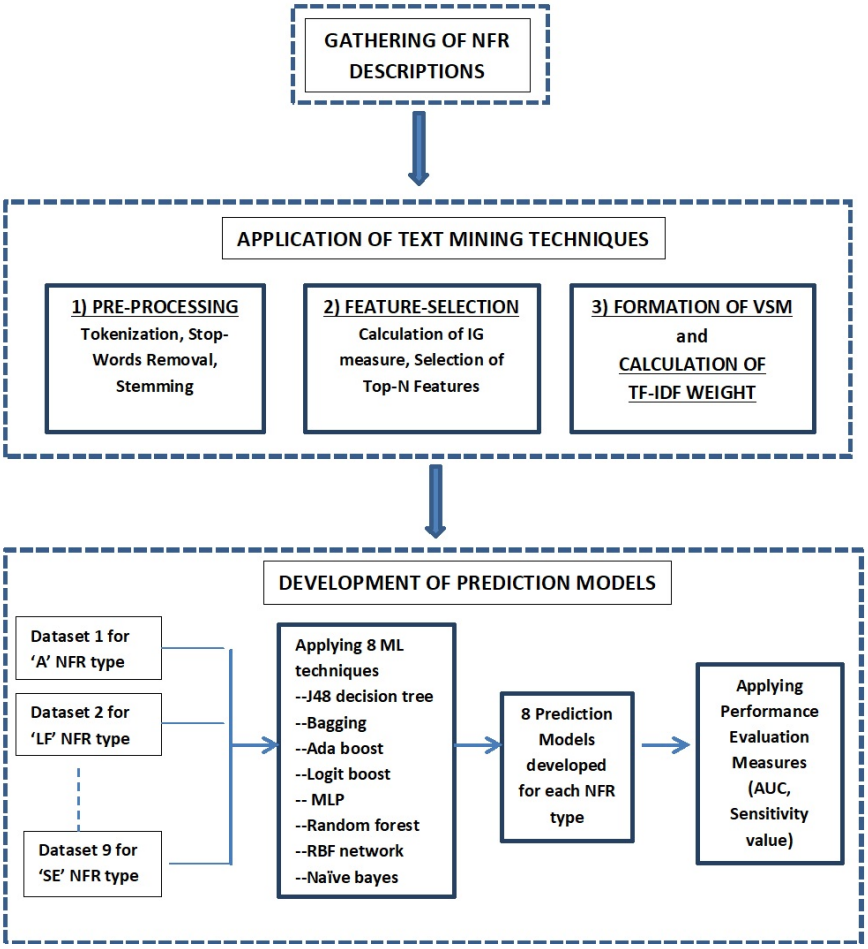

Figure 2. Framework used for classifying NFR descriptions

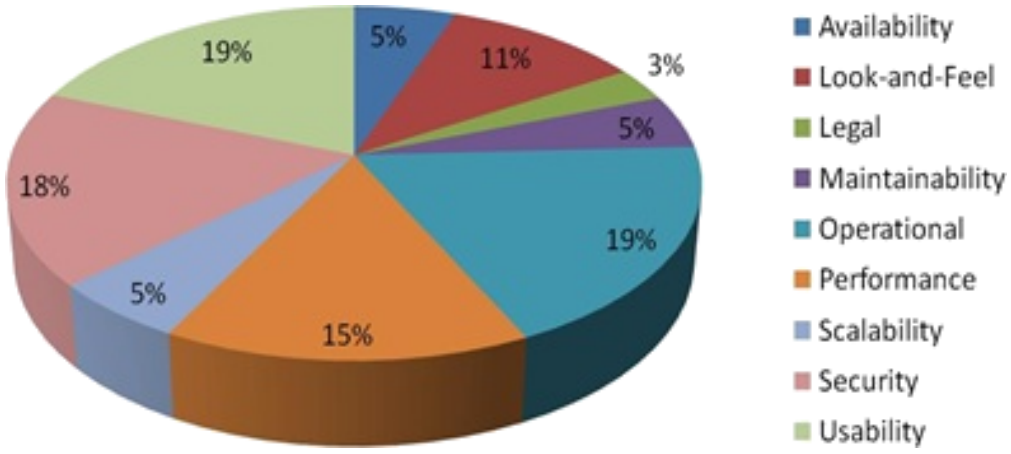

Figure 3. Percentage of requirements belonging to each type of NFR

326 NFR descriptions have been categorized into nine types of NFR viz. availability, look-and-feel, legal, maintainability, operational, performance, scalability, security, and usability. The percentage of NFRs belonging to each NFR type is shown in Figure 3.

4.2. Application of text mining techniques

This step concerns with the analysis of NFR descriptions to identify significant keywords (indicator terms) pertaining to each type of NFR. 326 NFR descriptions were extracted from the requirement specification documents and applied to a series of text mining steps. Figure 2 demonstrates the steps of text mining which has been explained below:

1. Pre-processing: Each document is represented using Bag of Words (BOW) representation method in which a document is considered to be a collection of thousands of words which occur in it at least once. Many of these words are not relevant for the learning task and their usage can degrade the performance of a classifier. Thus, a series of preprocessing tasks like tokenization, stop-words removal and stemming are required in order to remove the irrelevant words from the document. Text mining process begins by first converting the entire document in the form of tokens, i.e., a set of words. This is known as tokenization. This is followed by removing the words from the document that do not add any meaning to the data (stop words like prepositions, articles, conjunctions, etc.). Finally, stemming is performed [56]. Instead of stemming, another popular technique which could be used is lemmatization. However, we preferred stemming over lemmatization as the computation time involved in stemming is lesser as compared to lemmatization which is useful incase of large datasets and long texts.
2. Feature-Selection: Once pre-processing is performed, set of relevant words called indicator terms need to be identified specific to each type of NFR. In this work, Information-Gain (IG) measure has been used as the feature selection method. The working of IG measure is based on finding a collection of words from the document that best identify the target concept (NFR type) [57]. It is based on the concept of entropy deduction which occurs when the dataset is split on an attribute [57]. In other words, IG is the amount of information that is gained by evaluating the IG value of each attribute in the dataset. It is defined as the difference of the entropy of the dataset before the split and the entropy of the dataset after the split. Entropy of the entire dataset determines the amount of uncertainty in the information that needs to be assessed. IG measure works by associating a weight to each of the word obtained after pre-processing and then the top-N scoring words are selected as the indicator terms. In our work, the value of N is considered as 10 as the paper by Cleland-Huang et al [9] showed that good results were achieved when top-10 words were considered as compared to the results achieved when top-5 words or all the words were considered together. Table 3 depicts the top-10 indicator terms in decreasing order of IG measure, corresponding to each of the nine NFR types.
3. Vector Space Model: Once feature selection has been done using IG measure, we will have total number of N indicator terms which can be represented as $t_1, t_2, \ldots, t_N$. Each ith document is then represented as a N-dimensional vector consisting of N values written as $(X_{i1}, X_{i2}, \ldots, X_{iN})$. Here, X_{ij} is a TF-IDF (Term Frequency Inverse Document Frequency) weight measuring the importance of the jth term t_j in the ith document. The complete set of vectors corresponding to all the documents under consideration is called a Vector Space Model (VSM).

4.3. Development of prediction models

Once the indicator terms for each NFR type are identified, prediction models are developed for the classification of future NFR descriptions whose NFR type is not known. As depicted in Figure 4, nine datasets are developed corresponding to each NFR type from an initial dataset.

This initial dataset is the original NFR document that consists of 326 NFR descriptions belonging to one of the nine NFR types. Each dataset has the total number of instances as 326 with top-10 indicator terms of that particular NFR type as independent variables and a binary dependent variable.

Table 3. Top-10 indicator terms specific to each NFR type sorted by IG measure

Rank	A	LF	L	MN	O	PE	SC	SE	US
1	achiev	simul	regul	updat	environ	second	simultan	access	easi
2	hour	ship	compli	mainten	interfac	respons	handl	author	train
3	day	sound	disput	chang	window	time	year	ensur	understand
4	pm	interfac	legal	nfl	server	longer	capabl	authent	intuit
5	time	appear	rule	season	user	minut	support	prevent	instruct
6	long	appeal	histori	releas	web	return	number	allow	select
7	onlin	shot	requir	integr	establish	fast	expect	logon	realtor
8	avail	color	complianc	code	oper	flow	concurr	secur	learn
9	technic	compli	conform	pattern	second	add	increas	polici	symbol
10	year	access	standard	offer	custom	prepaid	launch	malici	natur

These top-10 indicator terms (extracted using IG measure) specific to each NFR type are shown in Table 3. Dependent variables will have the value of 1 or 0 depending on the type of NFR. For instance, dataset 1 is pertaining to "A" NFR type, so it will have the value of the dependent variable as 1 for all the NFR descriptions pertaining to "A" NFR type and the value of 0 for all other remaining NFR descriptions. Corresponding to each of the nine datasets, 8 prediction models are developed by employing 8 ML techniques (J48 decision tree, RF, Logitboost, Adaboost, MLP, RBF network, Bagging, NB) on each of the datasets. These prediction models can be used for the classification of future NFR descriptions whose NFR type is not known.

5. Result analysis

This section presents the results of eight different ML techniques when applied to nine different models developed with respect to their corresponding NFR types. The performance measures which have been used in evaluating the performance of these ML techniques are AUC and recall.

In this section, we will broadly discuss the following two Research Questions (RQs):
RQ1: Which ML technique is best for predicting each type of NFR such as performance, security, look-and-feel, etc.?
RQ2: Which NFR has been best predicted in terms of classification and misclassification rate?

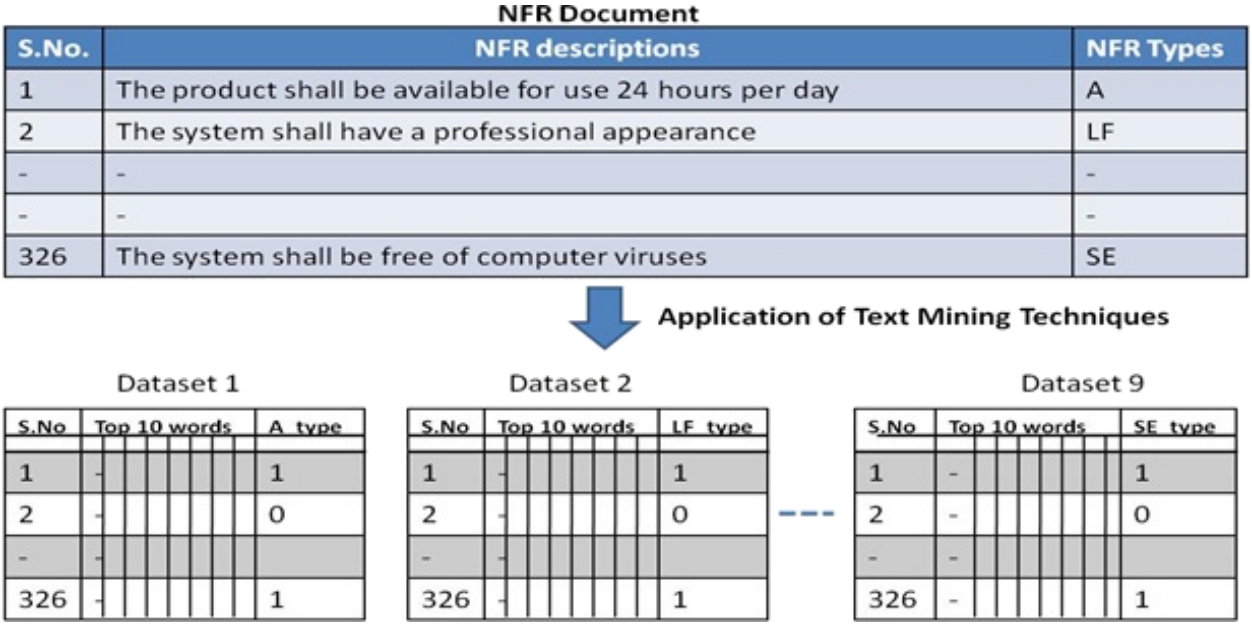

Figure 4. Process involved in NFR classification

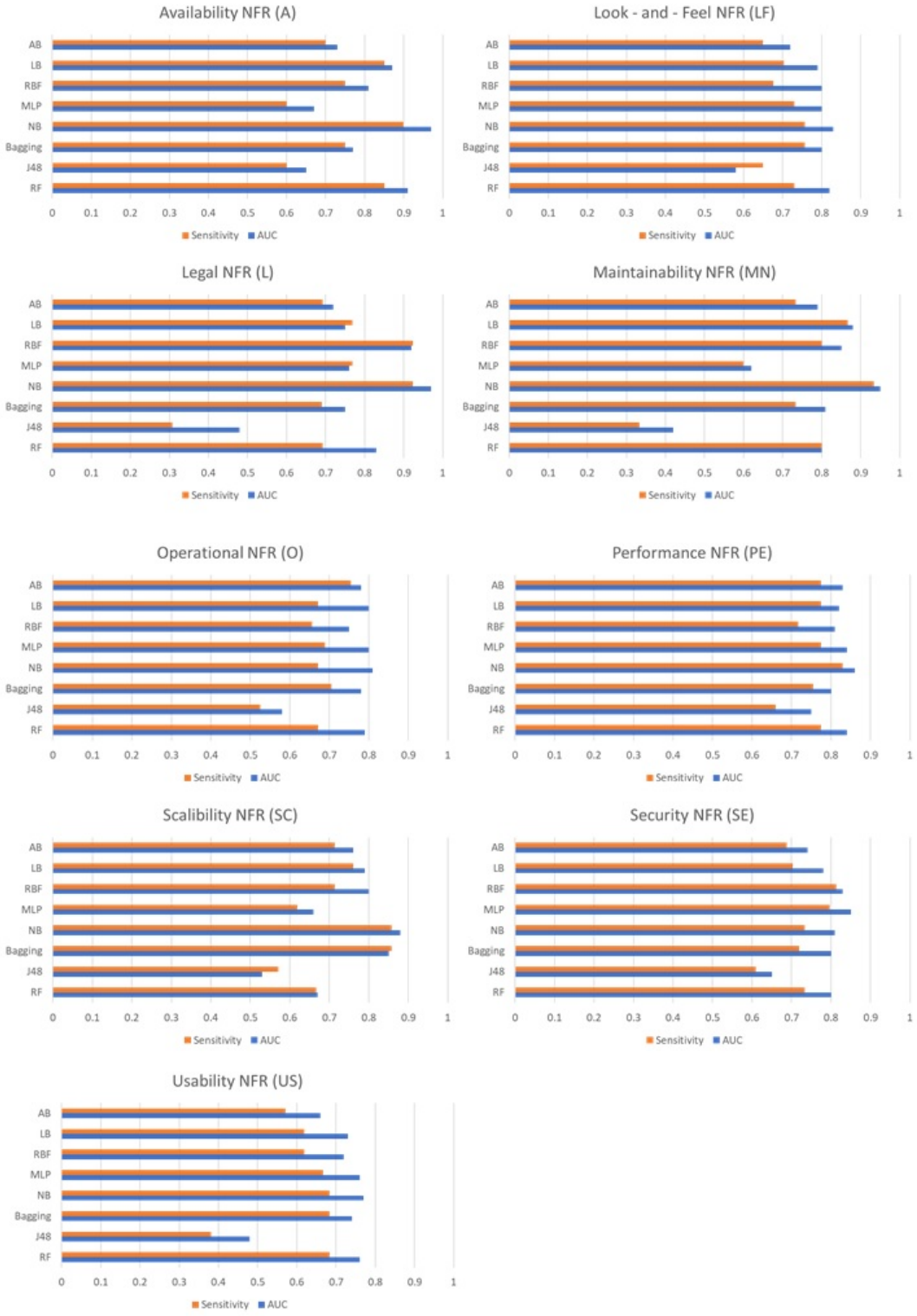

Figure 5. Graphical representation depicting the performance of ML techniques

5.1. Analysis of RQ1

To address the RQ1, the performance of the ML classification models to predict each category of NFR is depicted in Figure 5. Figure 5 depicts the comparative analysis of the ML models and determines how well these models have performed in predicting different types of NFR. The data values corresponding to each figure are shown in appendix (Tables A1–A4). We can observe from the Figure 5 that different NFR types responded differently to each classification method. For example, the performance of "A" NFR in terms of AUC when predicted by different classifiers is quite different, ranging from 0.65 to 0.97. Similar observations can be seen with all other NFR types. In other words, if a particular ML model has given a high value of AUC in predicting a particular type of NFR, it may not be necessary that it is also giving high accuracy in predicting other NFR types. This may be due to the reason that each NFR is very different from the others and thus, the top-10 words selected corresponding to each NFR are very different. Since the classification models are based on these top-10 words, the same classifier is performed differently on different NFRs. Given this scenario, the identification of a suitable classifier to predict each type of NFR will be highly beneficial for researchers and academicians. Figure 5 shows that the NFR "A" has been best predicted by NB giving AUC values as high as 0.97 and recall value as 90.0%. This is followed by RF giving AUC of 0.91 and recall as 85.0%. The graphs show that all NFRs except "SE" are best predicted by NB classifier. NB gives the highest AUC of 0.97, 0.83, 0.97, 0.95, 0.81, 0.86, 0.88, and 0.77 for "A", "LF", "L", "MN", "O", "PE", "SC" and "US" types of NFR, respectively. Their corresponding recall values are also high in the majority of the cases. The probable reason of NB performing well could be due to its assumption of attributes to be independent given the value of class variable [40]. Moreover, NB does not fit nearly as much, so there is no need to prune or process the network. For "SE" NFR, MLP has given the highest AUC of 0.85. After NB technique, RF and RBF techniques have shown the second highest AUC for the majority of the NFRs. The AUC of RF and RBF lies in the range of 0.67 to 0.91 and 0.72 to 0.92, respectively. The performance of the bagging technique can be considered as an average in predicting the NFR descriptions of all types with the values of AUC lying in the range of 0.74 to 0.85. Similar performance has been depicted by LB and AB techniques. The performance of these techniques has been overall good in predicting the NFR descriptions with the values of AUC lying in the range of 0.73 to 0.88 corresponding to LB and the AUC value falls in the range of 0.72 to 0.83 with respect to AB. On the contrary, J48 decision tree technique has not performed well in classifying the NFR descriptions into their respective types as the highest value of AUC obtained is 0.75 with 66.0% recall value. J48 has shown the lowest performance (in terms of both AUC and recall) in predicting all types of NFRs. From the above analysis, it is summarized that the performance of NB has outperformed all other

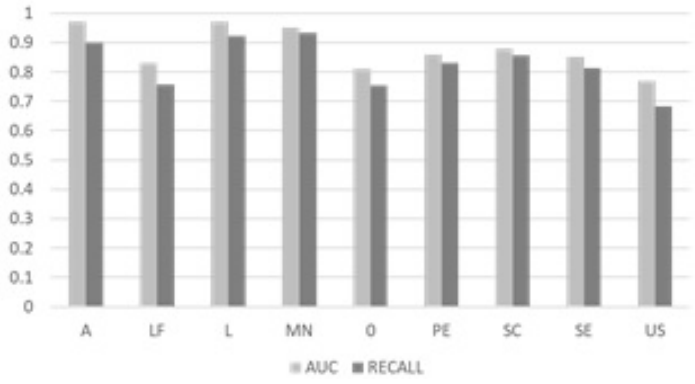

Figure 6. Bar graph showing comparison amongst NFRs in terms of AUC and recall

Table 4. Number of misclassifications of NFRs

NRF Type	Misclassifications			Misclassifications done by ML model
	False Negative (*FN*)	False Positive (*FP*)	Total (*FN*+*FP*)	
Availability (A)	7 (2.01%)	8 (2.30%)	15 (4.32%)	NB
Legal (L)	8 (2.30%)	5 (1.44%)	13 (3.74%)	NB
Look-and-Feel (LF)	8 (2.30%)	22 (6.34)	30 (8.64%)	NB
Maintainability (MN)	4 (1.15%)	5 (1.44%)	9 (2.59%)	NB
Operational (O)	15 (4.32%)	36 (10.37%)	55 (15.85%)	NB
Performance (PE)	25 (7.20%)	7 (2.01%)	32 (9.22%)	NB
Scalability (SC)	15 (4.32%)	8 (2.30%)	23 (6.62%)	NB
Security (SE)	8 (2.30%)	35 (10.08%)	43 (12.39%)	MLP
Usability (US)	1 (0.28%)	29 (8.35%)	30 (8.64%)	NB

classifiers and it is overall best in predicting the NFRs. We suggest researchers and academicians to use NB models for predicting the NFRs.

5.2. Analysis of RQ2

To address RQ2, NFRs are compared across each other by comparing the classification and misclassification ability of the best ML model for each type of NFR found in RQ1. The classification ability of the ML model is found by comparing the NFRs using AUC and recall values. In this study, the keywords form the basis for the identification of NFR documents into their respective types of NFRs. Thus, the performance of the ML model also depends on the values of these keywords. The NFRs which is predicted with high AUC and recall implies that the keywords pertaining to that NFR type are of great significance for the particular ML model to perform classification. It also implies that the dataset in turn consists of a good number of these requirements and thus the ML models are trained well on such datasets.

Whereas, the misclassifications by the ML model are found by determining the number of False Negative (*FN*) and False Positive (*FP*) of each type of NFR. As we have discussed in Section 3.2, *FN* occurs when the document actually belongs to a category "A" (positive) and is predicted by the model in category "Not A" (negative). Whereas, *FP* occurs when the document actually belongs to a category "Not A" (negative) and is predicted by the model to be in category "A". It is also discussed in Section 3.2 that *FN* is more significant that *FP* and thus, a model with lower *FN* is more desirable. The classification ability of the best ML model in predicting each type of NFR can be seen from Figure 6. On the other hand, the number of misclassifications done by the best ML classification model for each type of NFR is shown in Table 4. Table 4 shows the misclassifications for each type of NFR in terms of number of *FN* and *FP* along with their percentages. It can be seen from Figure 6 that "A", "L" and "MN" (in this order) have shown high AUC values, followed by "PE", "SC" and "SE". High recall values are shown by "MN", "L", and "A" (in this order). Thus, we can say that overall (taking AUC and recall together) "L" has been best predicted amongst all NFRs. This might be due to the reason that the "L" type of NFR is comparatively easy to understand and collect and thus, the data consisted of a good number of correctly elicited requirements pertaining to this NFR. In addition to this, Table 4 shows that only 13 "L" type NFRs are misclassified by NB model which is amongst the lowest compared to other misclassification rates. However, amongst these 13 misclassifications, there are more number of *FN* than *FP*, which is not desirable. Hence, such models may not be used for future unknown predictions. In contrast, the misclassification rates to predict "Ă" and "MN" type of NFR are also low (4.32% and 2.59%, respectively). "MN" in fact has been least misclassified amongst all the

NFRs. Also, the number of *FN* is less than *FP* for both "A" and "MN". Thus, overall, we can say that NB model has performed best in predicting "MN" and "A" types of NFRs when both classification and misclassification are taken together. The NFR which is predicted with lowest AUC and recall is "US" as can be seen from Figure 6. It can also be seen from Table 4 that 30 "US" type NFR have been misclassified which is amongst the highest. NB has given the highest misclassification rate (15.85%) for "O" type of NFR, implying that 55 "O" types of NFRs have been misclassified as some other NFRs. In terms of AUC and recall also "O" has been predicted with low values. The possible reason of misclassification may be due to the ambiguities caused by the indicator terms (keywords). These keywords form the basis for the identification of NFR documents into their respective types of NFRs and hence need to be carefully analysed to avoid misclassification. However, it has been observed that some of the NFR specifications of different NFR types have been described by the same set of keywords. In other words, some keywords tend to occur across multiple requirements of different NFR types. This gives rise to ambiguities, thus leading to false classification. Similar observations were given by Cleland-Huang et al. [9] and Sharma et al. [5]. Let us understand this with the help of a suitable example. Consider the keyword "colour" which is an indicator term of look-and-feel NFR. Presence of the term "colour" in the requirement sentence "The application shall match the colour of the schema set forth by Department of Homeland Security" clearly shows that the requirement is about look-and-feel NFR. However, the presence of the term "colour" in the requirement sentence "The list of dispute cases must be colour coded for easy identification of dispute cases based upon the dispute case status" does not necessarily represent any look-and-feel NFR, but rather represent "usability" NFR. The hint to identify look-and feel type NFR is the presence of other terms/patterns in the requirement sentence. For example, in the first sentence, the term "match" puts a constraint on "colour of the schema". However, in the second sentence, no such constraint is there to guarantee that it is a look-and-feel NFR. Thus, identification of the NFR using keywords may lead to false classification and a detailed analysis of semantic patterns and structure of NFR descriptions could be done to improve the results. Thus, future work will therefore investigate the possibility of using categories of indicator terms or extended training to improve these retrieval results.

5.3. Comparison with state-of-art

In this section, we discuss the implication of the results where we provide important insights inferred. We have compared our results to the state of the art [9, 12] and have qualitatively examined the wrongly classified cases to generate some useful insights. We have also discussed what could be done which may lead to improved performance of the proposed models.

The comparison in terms of recall values is shown in Table 5. For the purpose of comparison, we have considered the highest value of recall given by the different ML models for predicting each type of NFR. The bar chart in Figure 6 shows the highest recall value (given by different ML models) for each type of NFR. The authors, Rashwan et al. (2013) [12] have used the same dataset as used in our study and have classified the NFR descriptions using SVM classifier. From Table 5, it can be clearly seen that the prediction models proposed in this study have classified NFR descriptions of all types with higher recall values than the model developed by Rashwan et al. [12] using SVM classifier. This is despite the fact that SVM classifier is one of the most popularly used ML techniques in the field of NFR classification as can be seen from the literature survey (Table 1). We can even observe from the table that for some of the NFR types, viz. "MN" and "SC", the model of Rashwan et al. [12] has given extremely low values of recall (below 0.5). A recall value of 0.5 means that for every correct prediction, the next prediction is incorrect. There is no practical usage of such classifiers and they are known as random classifiers. The study [12] has shown that the SVM classifier has performed even worse than a random classifier for "MN" and "SC" NFR types. On the contrary, in this study, "MN" has been predicted with the highest recall value

Table 5. Comparison with state-of art

NRF Type	Recall (State-of-Art)		Recall (Our Model)
	Cleland, Huang et al. [9]	Rashwan et al. [12]	
Availability (A)	0.89	0.66	0.90
Legal (L)	0.70	0.61	0.92
Look-and-Feel (LF)	0.51	0.63	0.76
Maintainability (MN)	0.88	0.41	0.93
Operational (O)	0.72	0.66	0.75
Performance (PE)	0.62	0.70	0.83
Scalability (SC)	0.72	0.38	0.86
Security (SE)	0.81	0.70	0.81
Usability (US)	0.98	0.62	0.68

of 0.93 amongst all other NFRs. Similar observations are made when the recall values of NFRs in this study are compared with the recall values obtained in another study by Cleland-Huang et al. [9]. Table 5 shows that the recall values of all NFRs except "US" are higher than the recall values obtained in [9]. Thus, overall we can conclude that the results in this study are higher than the results obtained in both the compared studies [9] and [12].

Next, we analyze the NFR type which has been predicted with the lowest recall value in this study. As can be seen from Table 5, "US" NFR type has been predicted with the lowest recall.This might be due to the reason that NFR descriptions pertaining to "US" NFR type may be comparatively difficult to understand and collect. This is also evident from the dataset used in this study which shows that there are only 3% of NFR descriptions pertaining to "US" type of NFR. This low percentage of "US" type NFR descriptions give a low value of recall. Another implication which can be drawn from our results is that our models have not shown exceptional performance with respect to few of the NFR types. This might be because of the reason that NFR identification was done on the basis of indicator terms (keywords). In other words, the indicator terms form the basis of the classification of an unknown NFR type into its correct type. However, it has been observed that some keywords tend to occur across multiple requirements of different NFR types, leading to false classification. Similar observations were given by Cleland-Huang et al. [9] and Sharma et al. [5]. The illustration to explain this has been given in Section 5.2 (last paragraph).

6. Discussion

This section provides a discussion of the results in terms of cost/benefit analysis and how the results retrieved from this study can be useful for industry practitioners. The work in this paper is concerned with the development of nine different classification models specific to each type of NFR with the aim to classify NFRs into the respective types based on their descriptions specified in the requirement specification document. During the elicitation process, requirements analysts may generate large amounts of unstructured SRS documents consisting of the requirement specifications being scattered throughout the document in a random and adhoc fashion. The descriptions are extracted from the document and applied to a series of text mining steps to retrieve indicator terms specific to each NFR, leading to the detection and classification of NFR in the initial phases of SDLC process.

6.1. Analyzing returns on investment

We conducted cost-benefit analysis to evaluate the effectiveness of prediction models used for predicting the type of NFRs. The result section (Section 5) concludes the best ML technique for predicting each type of NFR amongst the various ML techniques used. In other words, we suggest that researchers, practitioners, and academicians may use those ML techniques for predicting the required NFR type. In this section, we discuss the cost of using the proposed model and the cost of not using the proposed model.

To calculate the cost of the model, we have considered a cost matrix having the value of 1 unit for false positives and false negatives (incorrect predictions), whereas 0 unit for true positives and true negatives (correct prediction). The values of 1 unit and 0 unit for incorrect and correct decisions, respectively, are considered as it is well understood that one may need to pay the price when the model takes wrong decisions while there is no cost involved when the model takes correct decisions. Therefore, the cost of using the model is calculated using the formula: (*No. offalsepositives*×1)+(*No. offalsenegatives*× 1). To conduct the cost-benefit analysis, the models are run at different threshold values and the performance of the models is analysed through the confusion matrix at each threshold value. The cost of the model, measured in terms of false positives and false negatives changes as the threshold changes. The minimum cost obtained is considered as the cost of the classification model (shown in Table 6). This cost is compared to the cost incurred without using the model. The cost incurred without using the model is found by selecting the same number of instances at random [58]. The difference between the values of the cost function by random selection and the value of the cost from the model is called gain. The Gain can be interpreted as the profit obtained by using the classification model instead of random selection of the same number of instances. We can observe from Table 6 that in all the cases, the cost incurred without using the model is more than the cost incurred by using the proposed model. In other words, we suggest that any professional from the industry who would use our model for classifying the NFRs into their types would be in profit from the gain as shown in Table 6.

For the purpose of demonstration, a plot of "cost/benefit curve" and "threshold curve" is depicted for MLP model used for predicting "SE" type of NFR. We have shown the plot corresponding to MLP model as it has given the highest gain of 69 units. Likewise, we can draw for all other models and infer the similar meaning. The "cost/benefit curve" is a plot of sample size (part of selected samples) on X-axis and cost/benefit on the Y-axis [59]. The "threshold curve" is a plot of true positive rate on Y-axis and sample size on X-axis. Threshold curve corresponds to the part of the selected instances ("Sample Size"). In other words, the threshold curve depicts the dependence of the part of "positive" samples retrieved during virtual screening upon the part of the samples selected from the whole dataset used for screening. It should be noted that only those samples are selected during virtual screening, for which the estimated probability of being "positive" exceeds the chosen threshold. The "cost/benefit curve" in Figure 7 shows that the minimum cost is 34 which is increasing reaching the maximum value of 283. The cross symbol denoted by "X" denotes the sample size retrieved during virtual screening. This sample size is important as we take the same number of instances to calculate the cost incurred without using the model. Similar inferences can be drawn for all other "cost/benefit curves" and "threshold curves".

6.2. Implications from industry viewpoint

The results of this work will be of interest to researchers as well as practitioners from the industry, who are interested in finding the type of NFR based on their descriptions in the early phases of software development, thus improving software quality. Timely identification of quality requirements would be of great benefit to the software developers from the industry as these quality requirements play a critical role in the design and architecture of the system. The architecture of the system built acts as the scaffolding in which the functionality of the system is delivered, thus ensuring that the system delivered meets the customer's functional expectations and needs. Consider a situation where a NFR remains undiscovered or is not elicitated properly during the early phase of software development, and is discovered at the later stages of development or when the software is released. In such a situation, the entire technical architecture has to be redesigned, leading to the wastage of limited resources in terms of time, money, and manpower [60, 61]. Thus, to avoid this situation, the models proposed in this study can

Table 6. Cost incurred with/without using proposed models

NRF Type	Proposed ML Model	Cost of the Model	Cost without Model	Gain	Profit/Loss
Availability	NB	12	41	29	Profit
Look-and-Feel	NB	24	59	36	Profit
Legal	NB	10	25	15	Profit
Maintainability	NB	9	27	19	Profit
Operational	NB	45	99	55	Profit
Performance	NB	28	80	52	Profit
Scalability	NB	21	28	7	Profit
Security	MLP	34	103	69	Profit
Usability	NB	30	85	55	Profit

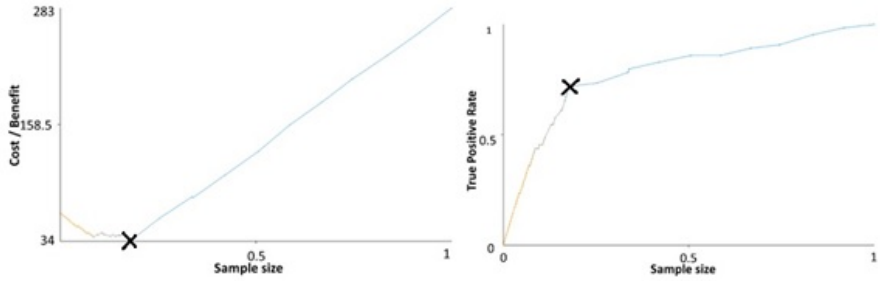

Figure 7. Cost/benefit curve

be used in the early phases for the identification and classification of NFRs. When the software is released and the customer finds that all his requirements (both functional as well as nonfunctional) are met, the customer feels satisfied and happy. This leads to the increasing the reputation and status of the software organization (in which the software is developed) in the market. Thus, customer satisfaction which is of utmost importance in today's scenario is met. Furthermore, practitioners from industry will be able to detect and classify NFRs from a previously uncategorized requirement specification in an automated way, thus avoiding the need for manual evaluation which like all human activities has a tendency to be error prone. It will also help researchers from the industry to extract viewpoints for different NFR qualities of interest. For example, a security analyst could issue a request to retrieve all descriptions related to security issues, or a GUI designer could issue a request to retrieve information about stakeholders" usability or look-and-feel concerns.

Thus, we have seen how industries can use the proposed models for identifying the NFRs and predicting the unknown NFR with its NFR type. Instead of using the proposed models, the identification of NFRs can also be done manually with the help of a human analyst. We have briefly discussed the effort required to conduct the work manually and have shown that the manual identification and prediction of NFRs is not feasible.

Human analyst can be considered as the requirement engineers in this study. Since the SRS documents are large enough and the NFRs are scattered throughout the document, it is very difficult or not feasible for the requirement engineers (human analyst) to perform the task of identifying the NFRs manually. Thus, this motivated the authors to automate this process with the help of algorithms and tools. This automation is done with the help of text mining techniques, which are used for extracting useful information (in the form of indicator terms specific to each NFR type) from a large number of documents (a large document corpus) without requiring humans to actually read and summarize the text. The automation of identification and classification of NFRs broadly consists of the following steps:

1. Use of text mining steps to retrieve the independent variables: A series of text mining steps were applied on the NFRs descriptions to retrieve the independent variables (indicator terms). These steps begin with pre-processing (tokenization, stop words removal, stemming) followed by application of feature selection method and finally applying TF-IDF weighting. Each of these steps can be studied in detail from Section 4.2. In this work, a dataset consisting of a total of 326 NFR descriptions which are categorized into 9 types of NFR was considered. For each type of NFR, indicator terms (top-10 words) were retrieved. So, we had to run our text mining module nine number of times and each time the above steps of text mining were done. An automated text mining module takes as input all the 326 NFR descriptions at a time which had not been the case if it was done by human analyst. Text mining steps had to be applied to each of the 326 NFR descriptions individually if it was done manually by human analyst. The same procedure had to be done 9 times for each NFR type. This is humanly impossible as each of the above text mining steps is complex in nature, thus consuming a lot of time. Pre-processing step involves natural language complications as we deal with textual requirements which are purely written in a natural language which is followed by feature selection and TF-IDF weighting that involves mathematical calculations. It is therefore not possible to do manual computation on each NFR to retrieve top-10 words and then calculate TF-IDF values for these words.
   Thus, we have automated this process with the help of a tool developed by the authors. The input to this tool is a set of 326 NFRs descriptions and the output is top-10 words sorted on the basis of Information-Gain measure (feature selection method). Thus, the entire process is automated and the time taken to produce the output is less than approximately 1 minute. Whereas, to perform the same activities manually, it would take us hours and may not be even feasible for large datasets.
2. Development of prediction models using machine learning classifiers: Once indicator terms (independent variables) for each NFR type are retrieved, then dataset corresponding to each of the 9 types of NFR is made consisting of a binary dependent variable (NFR type) having the value of 1 or 0 depending on the type of NFR. Corresponding to each dataset, eight different prediction models were developed by employing eight ML classifiers on each dataset viz. J48 decision tree, Random Forest (RF), Logitboost, Adaboost, Multi-Layer Perceptron (MLP), Radial Basis Function (RBF) network, Bagging, Naïve Bayes (NB). These ML classifiers are implemented through an open source tool, WEKA. There were in total 72 models (9 datasets * 8 ML classifiers= 72 prediction models) and it was observed that to run each model, CPU time of approximately 30 sec to 1:30 minutes was required.
   Had the same task been done manually, the human analyst would have to use his skills and knowledge in classifying NFR description into particular type of NFR. Also, as the number of models is quite large (72), it would have taken a lot of time to complete the computation. Moreover, it would have been very difficult to achieve a high performance as achieved by our models.
   Never the less, human computation is always prone to error. Human error is inevitable and normal which may lead to system failure if not handled at the right time. Thus, we conclude that our prediction models are highly recommended in contrast to human analyst (i.e., work done manually) for classification of future NFR descriptions whose NFR type is not known, leading to less computation time and more accuracy.

7. Threats to validity

The empirical validation in this work has certain limitations which may adversely affect the validity of the results. These limitations are discussed in terms of four threats to validity, viz. construct

validity, internal validity, external validity, and conclusion validity.

1. Construct Validity
   Construct validity is one of the most important threats to validity. It is defined as the extent to which the variables (independent and dependent variables) and the performance parameters precisely measure the concept they intend to measure [62–64].
   This threat can be due to the improper collection of the dependent variable and the independent variables. The dependent variable used in the study refers to the "types of NFRs" and the independent variables are the top few words of the document which determine how important they are within that particular document and also across a group of other documents. The collection of data for the classification of NFRs into their respective types has been done by mining the descriptions of NFRs specified in SRS document using text mining techniques. These text mining techniques cannot ensure complete correctness. This is so as this module is based on a number of pre-processing steps like tokenization, stop-words removal, stemming, etc., before the application of IG measure and Tf-Idf weighting approach could be done. Now, all these pre-processing steps use a set of English vocabulary words available online as their base, which may result in arbitrariness. However, these pre-processing steps were manually evaluated to reduce the amount of randomness and uncertainty in the accuracy and preciseness of the results so obtained. In addition to this, we have used a standard performance measure, viz, Area Under the ROC Curve (AUC) to measure the performance of the models. This measure is widely used in related research and sufficiently measures the performance of the models accurately. Thus, the proper collection of independent and the dependent variable and the use of a stable performance measure minimize the threat to construct validity to a large extent.
2. Internal Validity
   "Internal validity is defined as the degree to which conclusions can be drawn about the causal effect of independent variable on the dependent variable" [64]. In this work, independent variables used are a set of pre-processed words obtained using IG measure. These independent variables are not related to each other in any way. All these words together determine the value of dependent variable (type of NFR). It is not possible to determine the causal effect of each independent variable on the dependent variable. In other words, the goal of this study is to develop prediction models for classifying NFRs into various categories rather than discovering the cause-effect relationships. Thus, the threat to internal validity does not exist in the study.
3. External Validity
   External validity is defined as the extent to which the results of the study can be generalized universally. It concerns itself with finding out whether the results produced by the study are applicable in different domains or can be replicated in different scenarios for which the results are not evaluated [65]. In other words, external validity could be ensured if we could have applied the same approach on a different dataset and produced the same results. To ensure external validity, more research is needed as in this work, authors have not used the proposed models on some other datasets to identify the type of NFR. Authors have planned to take different datasets and use the same models to identify NFR specifications in their future work. In the future, we will be comparing the results of different datasets across different projects having diverse characteristics.
4. Conclusion Validity
   Conclusion validity threats include all those threats which affect the conclusion of the study. In other words, all the threats which may lead to improper results or conclusions of the study are called as conclusion validity threats [65]. The authors in this study have not performed the statistical evaluation of the results using statistical tests. Thus, this leads to a conclusion validity threat. How-

ever, to provide strong conclusions, we have compared the performance of 8 ML classifiers to classify 9 types of NFRs. Very few studies in literature have used such a large number of ML classifiers. Thus, such comparison and evaluation may lead to fair conclusion, reducing the threat to conclusion validity.

In addition to this, conclusion validity threat may also occur since we had worked on the dataset consisting of less number of projects. The dataset used in this work for empirical analysis was the same dataset which was used by authors Cleland-Huang et al. [9]. This dataset consists of a set of 15 projects which were made by MS students at DePaul University, containing a total of 326 NFR descriptions. To eliminate this threat, we may include more such projects in the future study leading to stronger conclusions. Also, conclusion validity threat may occur since we have worked on a limited number of basic NFRs. There are more fine-grained NFRs listed by ISO standards 9126 and 25010 [65]. The inclusion of such NFRs may affect the results and change the performance of the models to some extent. However, the NFRs we considered covered most of the quality constraints enforced in our experimental setup. In addition to this, the dataset we used has the requirements pertaining to these basic NFR types. Due to these reasons, we did not feel the requirement to include fine grained NFRs. Inclusion of such fine grained NFRs may also increase the complexity of the work and thus, maybe included only when there is a requirement of extensive analysis.

8. Conclusions and future work

Classification of NFR descriptions into their respective types is very essential for software development meeting the basic quality determining features like security, scalability, maintainability, etc. The NFR descriptions in the software requirement specification document should therefore be analyzed carefully and mapped into their corresponding NFR types. In this paper, text mining steps have been incorporated to mine the NFR descriptions and thereby identify a set of few keywords. These keywords are the top few words which hold the essential information about NFR. Keywords help to classify the NFR descriptions using different ML techniques. Eight different ML techniques viz. J48, RF, LB, AB, MLP, RBF, Bagging and NB have been used to classify the NFR descriptions into nine types of NFR. The results which have been obtained from the study are summarized as follows:

1. With respect to each of the nine NFR types, eight prediction models have been developed corresponding to eight ML techniques.
2. The retrieval of the keywords specific to each type of NFR has been done using IG measure as the feature selection method. These keywords hold essential information about the NFR type and are used to develop prediction models by employing a suitable ML technique. Top-10 words sorted by IG measure have been selected corresponding to each of the nine models.
3. The study analyzes the performance of ML models in terms of classification and misclassification rate to determine the best model for predicting each type NFR descriptions.
4. The performance of each of these nine models is evaluated using ROC analysis. The results indicated that the performance of all the NFRs except “SE” is best predicted by NB classifier. NB gives the highest AUC of 0.97, 0.83, 0.97, 0.95, 0.81, 0.86, 0.88, and 0.77 for “A”, “LF”, “L”, “MN”, “O”, “PE”, “SC”, and “US” types of NFR respectively. Their corresponding recall values are also high in majority of the cases. This is so because this technique has a high bias and low variance which works well for the dataset having a small size.
5. This is followed by the performance of RBF and RF technique. The AUC values of RF and RBF lies in the range of 0.67 to 0.91 and 0.72 to 0.92, respectively. Average performance has been depicted by the remaining techniques which are LB, AB, Bagging, and MLP.
6. On the contrary, J48 decision tree technique has not performed well in classifying the NFR

descriptions into their respective types. This technique has shown the lowest performance (in terms of both AUC and recall) in predicting all the types of NFRs.

7. Among all the NFRs, it has been observed that most of the classifiers predicted "PE" and "A" type of NFR most accurately. On the other hand, "US" NFR type has been predicted with lowest accuracy as there are only 3% (lowest percentage) of NFR descriptions pertaining to this NFR type in the dataset, thus giving low value of recall.
8. Overall, we concluded that the performance of NB model has performed best in predicting "MN" and "A" type of NFRs when both classification and misclassifications are taken together.
9. Also, cost-benefit analysis was conducted from which it was concluded that the cost incurred without using our proposed models is more than the cost incurred when using the proposed models.

As our future work, we intend to replicate our empirical study across more datasets of similar type, i.e., academic datasets where requirements are written by students and researchers to obtain generalized and well-formed results. Apart from this, we also intend to conduct more experiments and provide benchmarks for future performance by exploring different types of datasets like industrial datasets where requirements are written to describe or simulate industrial products or informal datasets where requirements are written by end-users as comments, reviews, posts and requests in open source communities (sourceforge.net) or written in different Apps of varying categories (books, education, games, health, lifestyle, navigation, news, productivity, travel and utilities). Moreover, we can analyze the effectiveness of the classification models by incorporating more search-based or evolutionary algorithms instead of basic ML algorithms.

References

[1] M. Glinz, "On non-functional requirements," in *Proceedings of the 15th IEEE International Requirements Engineering Conference*. Delhi, India: IEEE, 2010, pp. 21–26.

[2] M. Glinz, "Rethinking the notion of non-functional requirements," in *Proceedings of the 3rd world congress for software quality*, Munich, Germany, 2005, pp. 55–64.

[3] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*. KluwerAcademic, 2000.

[4] S. Amasaki and P. Leelaprute, "The effects of vectorization methods on non-functional requirements classification," in *44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. Prague, Czech Republic: IEEE, 2018, pp. 175–182.

[5] V. Sharma, R. Ramnani, and S. Sengupta, "A framework for identifying and analysing non-functional requirements from text," in *Proceedings of the 4th International Workshop on Twin Peaks of Requirements and Architecture*. New York, USA: ACM, 2014, pp. 1–8.

[6] L. Hakim and S. Rochimah, "Oversampling imbalance data: Case study on functional and non functional requirement," in *Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS)*. Batu, East Java, Indonesia: IEEE, 2018, pp. 315–319.

[7] D. Méndez-Fernández, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetro, T. Conte, M. Christiansson, D. Greer, C. Lassenius, T. Mannisto, M. Nayebi, M. Oivo, B. Penzenstadler, D. Pfahl, R. Prikladnicki, G. Ruhe, A. Schekelmann, S. Sen, R. Spinola, A. Tuzcu, J. de la Vara, and R.Wieringa, "Naming the pain in requirements engineering," *Empirical software engineering*, Vol. 22, No. 5, 2017, pp. 2298–2338.

[8] R. Svensson, M. Host, and B. Regnell, "Managing quality requirements: A systematic review," in *36th EUROMICRO Conference on Software Engineering and Advanced Applications*. Lille, France: IEEE, 2010, pp. 261–268.

[9] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "Automated classification of non-functional requirements," *Requirements Engineering*, Vol. 12, No. 2, 2007, pp. 103–120.

[10] I. Hussain, L. Kosseim, and O. Ormandjieva, "Using linguistic knowledge to classify non-functional requirements in SRS documents," in *International Conference on Application of Natural Language to Information Systems*. London, UK: Springer, 2008, pp. 287–298.

[11] G. Gokyer, S. Cetin, C. Sener, and M. Yondem, "Non-functional requirements to architectural concerns: ML and NLP at crossroads," in

Proceedings of the 2008 the 3rd international conference on software engineering advances. Sliema, Malta: IEEE, 2008, pp. 400–406.

[12] A. Rashwan, O. Ormandjieva, and R. Witte, "Ontology-based classification of non-functional requirements in software specifications: A new corpus and svm-based classifier," in *37th Annual Computer Software and Applications Conference*. Kyoto, Japan: IEEE, 2013, pp. 381–386.

[13] T. Nguyen, J. Grundy, and M. Almorsy, "Rule-based extraction of goal-use case models from text," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. Bergamo, Italy: ACM, 2015, pp. 591–601.

[14] A. Casamayor, D. Godoy, and M. Campo, "Identification of non-functional requirements in textual specifications: A semi-supervised learning approach," *Information and Software Technology*, Vol. 52, No. 4, 2010, pp. 436–445.

[15] W. Zhang, Y. Yang, Q. Wang, and F. Shu, "An empirical study on classification of non-functional requirements," in *Twenty-Third International Conference on Software Engineering and Knowledge Engineering*, Miami Beach, USA, 2011, pp. 444–449.

[16] J. Slankas and L. Williams, "Automated extraction of non-functional requirements in available documentation," in *1st International Workshop on Natural Language Analysis in Software Engineering*. San Francisco, USA: IEEE, 2013, pp. 9–16.

[17] P. Singh, D. Singh, and A. Sharma, "Rule-based system for automated classification of non-functional requirements from requirement specifications," in *International conference on Advances in Computing, Communications and Informatics*. Jaipur, India: IEEE, 2016, pp. 620–626.

[18] Z. Kurtanovic and W. Maalej, "Automatically classifying functional and non-functional requirements using supervised machine learning," in *25th International Requirements Engineering Conference Workshops*. Lisbon, Portugal: IEEE, 2017, pp. 490–495.

[19] A. Hindle, N.A. Ernst, M.W. Godfrey, and J. Mylopoulos, "Automated topic naming," *Empirical Software Engineering*, Vol. 18, No. 6, 2013, pp. 1125–1155.

[20] D. Ott, "Automatic requirement categorization of large natural language specifications at Mercedes-Benz for review improvements," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Essen, Germany: Springer, 2013, pp. 50–64.

[21] M. Lu and P. Liang, "Automatic classification of non-functional requirements from augmented app user reviews," in *21st International Conference on Evaluation and Assessment in Software Engineering (EASE)*. Karlskrona, Sweden: ACM, 2017, pp. 344–353.

[22] R. Deocadez, R. Harrison, and D. Rodriguez, "Automatically classifying requirements from app stores: A preliminary study," in *25th International Requirements Engineering Conference Workshops*. Lisbon, Portugal: IEEE, 2017, pp. 367–371.

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

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

[25] E. Knauss, S. Houmb, K. Schneider, S. Islam, and J. Jürjens, "Supporting requirements engineers in recognising security issues," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Essen, Germany: Springer, 2011, pp. 4–18.

[26] M. Riaz, J. King, J. Slankas, and L.Williams, "Hidden in plain sight: Automatically identifying security requirements from natural language artifacts," in *International Conference on Requirements Engineering Conference*. Karlskrona, Sweden: IEEE, 2014, pp. 183–192.

[27] R. Jindal, R. Malhotra, and A. Jain, "Automated classification of security requirements," in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. Jaipur, India: IEEE, 2016, pp. 2027–2033.

[28] R. Malhotra, A. Chug, A. Hayrapetian, and R. Raje, "Analyzing and evaluating security features in software requirements," in *Innovation and Challenges in Cyber Security*. Noida, India: IEEE, 2016, pp. 26–30.

[29] E. Knauss and D. Ott, "(semi-) automatic categorization of natural language requirements," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Essen, Germany: Springer, 2014, pp. 39–54.

[30] R. Maiti and F. Mitropoulos, "Capturing, eliciting, predicting and prioritizing (CEPP) non-functional requirements metadata during

the early stages of agile software development," in *Proceedings of the SoutheastCon*. Fort Lauderdale, USA: IEEE, 2015, pp. 1–8.

[31] A. Mahmoud and G. Williams, "Detecting, classifying, and tracing non-functional software requirements," *Requirements Engineering*, Vol. 21, No. 3, 2016, pp. 357–381.

[32] Z. Abad, O. Karras, P. Ghazi, M. Glinz, G. Ruhe, and K. Schneider, "What works better? A study of classifying requirements," in *25th International Requirements Engineering Conference Workshops*. Lisbon, Portugal: IEEE, 2017, pp. 496–501.

[33] L. Toth and L. Vidacs, "Study of various classifiers for identification and classification of non-functional requirements," in *International Conference on Computational Science and Its Applications*. Springer, 2018, pp. 492–503.

[34] S. Uddin, A. Khan, M. Hossain, and M. Moni, "Comparing different supervised machine learning algorithms for disease prediction," *BMC Medical Informatics and Decision Making*, Vol. 19, No. 281, 2019, pp. 1–16.

[35] Y. Dengju, J. Yang, and X. Zhan, "A novel method for disease prediction: Hybrid of random forest and multivariate adaptive regression splines," *Journal of Computers*, Vol. 8, No. 1, 2013, pp. 170–177.

[36] C. Sinclair, L. Pierce, and S. Matzner, "An application of machine learning to network intrusion detection," in *Proceedings of 15th Annual Computer Security Applications Conference (ACSAC'99)*. Phoenix, AZ, USA: IEEE, 1999, pp. 371–377.

[37] E. Aleskerov, B. Freisleben, and B. Rao, "Cardwatch: A neural network based database mining system for credit card fraud detection," in *Proceedings of the IEEE/IAFE Computational Intelligence for Financial Engineering (CIFEr)*. New York, NY, USA: IEEE, 1997, pp. 220–226.

[38] S. Rathore and S. Kumar, "An empirical study of some software fault prediction techniques for the number of faults prediction," *Soft Computing*, Vol. 21, 2016, pp. 7417–7434.

[39] A. Okutan and O. Yildiz, "Software defect prediction using bayesian networks," *Empirical Software Engineering*, Vol. 19, No. 1, 2014, pp. 154–181.

[40] T. Patil and S. Sherekar, "Performance analysis of Naïve Bayes and J48 classification algorithm for data classification," *International Journal of Computer Science and Applications*, Vol. 6, No. 2, 2013, pp. 256–261.

[41] J. Qinlan, *C4.5: Programs for machine Learning*, 1st ed. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[42] M. Danham and S. Sridhar, *Data mining: Introductory Advanced Topics*, 1st ed. Person Education, 2006.

[43] Y. Freund and R. Schapire, "A short introduction to boosting," *Journal of Japanese Society for Artificial Intelligence*, Vol. 14, No. 5, 1999, pp. 771–780.

[44] B. Widro and M. Lehr, "30 years of adaptive neural networks: Perceptron, madaline, and backpropagation," *Proceedings of the IEEE*, Vol. 78, No. 9, 1990, pp. 1415–1442.

[45] D. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, Vol. 2, 1998, pp. 321–355.

[46] S. Eyheramendy, D. Lewis, and D. Madigan, "On the Naive Bayes model for text categorization," in *Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics*, 2002.

[47] A. McCallum and K. Nigam, "A comparison of event models for Naive Bayes text classification," in *Learning for Text Categorization*, M. Sahami, Ed. AAAI Press, 1998, pp. 41–48.

[48] R. Malhotra, *Empirical Research in Software Engineering – Concepts, Analysis and Applications*, 1st ed. India: CRC Press, 2015.

[49] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the Royal Statistical Society*, Vol. 36, No. 2, 1974, pp. 111–147.

[50] Y. Jiang, B. Cukic, and Y. Ma, "Techniques for evaluating fault prediction models," *Empirical Software Engineering*, Vol. 13, No. 15, 2008, pp. 561–595.

[51] H. He and E. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 21, No. 9, 2009, pp. 1263–1284.

[52] T. Menzies, A. Dekhtyar, J. Distefance, and J. Greenwald, "Problems with precision: A response to "comments on 'data mining static code attributes to learn defect predictors'"," *IEEE Transactions on Software Engineering*, Vol. 33, No. 9, 2007, pp. 637–640.

[53] F. Provost and T. Fawcett, "Robust classification for imprecise environments," *Machine Learning*, Vol. 42, 2001, pp. 203–231.

[54] R. Shatnawi, "Improving software fault-prediction for imbalanced data," in *Proc. of International Conf. on Innovations in Information Technology*. Abu Dhabi, United Arab Emirates: IEEE, 2012, pp. 54–59.

[55] T. Fawcett, "An introduction to ROC analysis," *Pattern Recogn Lett*, Vol. 27, No. 8, 2006, pp. 861–874.

[56] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, Vol. 34, No. 1, 2002.

[57] T. Menzies and A. Marcus, "Automated severity assessment of software defect reports," in *IEEE International Conference on Software Maintenance (ICSM)*. Beijing, China: IEEE, 2008, pp. 346–355.

[58] E. Arisholm and L. Briand, "Predicting fault-prone components in a Java legacy system," in *Proceedings of ACM/IEEE international symposium on empirical software engineering*. IEEE, 2006, pp. 8–17.

[59] R. Malhotra and M. Khanna, "An exploratory study for software change prediction in object-oriented systems using hybridized techniques," *Autom Softw Eng*, Vol. 24, No. 3, 2017, pp. 673–717.

[60] L. Briand, J. Wust, and J. Daly, "Exploring the relationship between design measures and software quality in object-oriented systems," *J Syst Softw*, Vol. 51, No. 3, 2000, pp. 245–273.

[61] L. Briand, J. Wust, and H. Lounis, "Replicated case studies for investigating quality factors in object oriented designs," *Empir Softw Eng J*, Vol. 6, No. 1, 2001, pp. 11–58.

[62] R. Malhotra and M. Khanna, "Threats to validity in search-based predictive modelling for software engineering," *IET Software*, Vol. 12, No. 4, 2018, pp. 293–305.

[63] A. Dean, D. Voss, and D. Draguljic, *Design and analysis of experiments*, 10th ed. New York: Springer, 1999.

[64] Y. Zhou, H. Leung, and B. Xu, "Examining the potentially confounding effect of class size on the associations between object-oriented metrics and change-proneness," *IEEE Transactions on Software Engineering*, Vol. 35, No. 5, 2009, pp. 607–623.

[65] R. Harrison, S. Counsell, and R. Nithi, "Experimental assessment of the effect of inheritance on the maintainability of object-oriented systems," *Journal of Systems and Software*, Vol. 52, No. 2–3, 2000, pp. 173–179.

A. Appendix

Table A1. Performance evaluation of random forest and J48 decision tree techniques

S. No.	Type of NFR	Random Forest			J48		
		AUC	Sens (%)	CutOff	AUC	Sens (%)	CutOff
1	A	0.91	85.0	0.02	0.65	60.0	0.02
2	LF	0.82	73.0	0.05	0.58	64.9	0.08
3	L	0.83	69.2	0.05	0.48	30.8	0.02
4	MN	0.80	80.0	0.00	0.42	33.3	0.04
5	O	0.79	67.2	0.12	0.58	52.5	0.12
6	PE	0.84	77.4	0.05	0.75	66.0	0.06
7	SC	0.67	66.7	0.01	0.53	57.1	0.05
8	SE	0.80	73.4	0.06	0.65	60.9	0.12
9	US	0.76	68.3	0.09	0.48	38.1	0.18

Table A2. Performance Evaluation of Bagging and Naïve Bayes techniques

S. No.	Type of NFR	Bagging			Naïve Bayes		
		AUC	Sens (%)	CutOff	AUC	Sens (%)	CutOff
1	A	0.77	75.0	0.03	0.97	90.0	0.10
2	LF	0.80	75.7	0.07	0.83	75.7	0.01
3	L	0.75	69.0	0.02	0.97	92.3	0.01
4	MN	0.81	73.3	0.02	0.95	93.3	0.06
5	O	0.78	70.5	0.10	0.81	67.2	0.12
6	PE	0.80	75.5	0.07	0.86	83.0	0.01
7	SC	0.85	85.7	0.04	0.88	85.7	0.01
8	SE	0.80	71.9	0.09	0.81	73.4	0.03
9	US	0.74	68.3	0.10	0.77	68.3	0.06

Table A3. Performance Evaluation of MLP and RBF network techniques

S. No.	Type of NFR	MLP			RBF		
		AUC	Sens (%)	CutOff	AUC	Sens (%)	CutOff
1	A	0.67	60.0	0.00	0.81	75.0	0.01
2	LF	0.80	73.0	0.04	0.80	67.6	0.06
3	L	0.76	76.9	0.01	0.92	92.3	0.05
4	MN	0.62	60.0	0.01	0.85	80.0	0.01
5	O	0.80	68.9	0.10	0.75	65.6	0.13
6	PE	0.84	77.4	0.06	0.81	71.7	0.06
7	SC	0.66	61.9	0.02	0.80	71.4	0.02
8	SE	0.85	79.7	0.08	0.83	81.3	0.08
9	US	0.76	66.7	0.09	0.72	61.9	0.09

Table A4. Performance Evaluation of Logitboost and Adaboost techniques

S. No.	Type of NFR	Logitboost			Adaboost		
		AUC	Sens (%)	CutOff	AUC	Sens (%)	CutOff
1	A70.87	85.0	0.05	0.73	70.0	0.02	
2	LF	0.79	70.3	0.05	0.72	64.9	0.06
3	L	0.75	76.9	0.01	0.72	69.2	0.01
4	MN	0.88	86.7	0.04	0.79	73.3	0.01
5	O	0.80	67.2	0.12	0.78	75.4	0.14
6	PE	0.82	77.4	0.09	0.83	77.4	0.06
7	SC	0.79	76.0	0.02	0.76	71.4	0.02
8	SE	0.78	70.3	0.08	0.74	68.8	0.09
9	US	0.73	61.9	0.11	0.66	57.1	0.11