2017 IEEE 25th International Requirements Engineering Conference

Automatically Classifying Functional and Non-Functional Requirements Using Supervised Machine Learning

Zijad Kurtanović
University of Hamburg
Hamburg, Germany
kurtanovic@informatik.uni-hamburg.de

Walid Maalej
University of Hamburg
Hamburg, Germany
maalej@informatik.uni-hamburg.de

***Abstract*—In this paper, we take up the second RE17 data challenge: the identification of requirements types using the "Quality attributes (NFR)" dataset provided. We studied how accurately we can automatically classify requirements as functional (FR) and non-functional (NFR) in the dataset with supervised machine learning. Furthermore, we assessed how accurately we can identify various types of NFRs, in particular usability, security, operational, and performance requirements.**

We developed and evaluated a supervised machine learning approach employing meta-data, lexical, and syntactical features. We employed under- and over-sampling strategies to handle the imbalanced classes in the dataset and cross-validated the classifiers using precision, recall, and F1 metrics in a series of experiments based on the Support Vector Machine classifier algorithm. We achieve a precision and recall up to ~92% for automatically identifying FRs and NFRs. For the identification of specific NFRs, we achieve the highest precision and recall for security and performance NFRs with ~92% precision and ~90% recall. We discuss the most discriminating features of FRs and NFRs as well as the sampling strategies used with an additional dataset and their impact on the classification accuracy.

***Index Terms*—Requirements, Classification, Machine Learning, Imbalanced Data**

I. Introduction

Requirements are often classified as functional (FR) and non-functional (NFRs) [8], [29]. While there is a broad consensus on the definition of FRs, this is rather not the case for NFRs [14]. Typically, FRs describe the system functionality, while NFRs describe system properties and constraints [7], [21]. This distinction has influenced how requirements are handled in practice during elicitation, documentation, and validation [6], [8]. NFRs are often identified and specified relatively late in the development process [4], [5] and are barely explicitly managed [28], [31]. This suggests that developers may fail to appreciate the importance of assessing NFRs and their early detection [3], [5], and assisting them to identify and manage NFRs can reduce this risk.

The automatic extraction and classification of requirements from text documents has been the focus of several requirements engineering researchers. Cleland-Huang et al. [6] presented an approach for retrieving and classifying NFRs from structured and unstructured documents. The authors first mine weighted indicator terms that they then used to classify additional requirements. For the NFRs usability, security, operational, and performance they achieve a recall between 20% and 90% and precision between 13% and 73% respectively. Knauss et al. [20] extracted clarification patterns from software team communication artifacts in the lifetime of a requirement. They employed a Naive Bayes approach to automatically detect requirements that are not progressing in a project. Slankas and Williams [28] developed an approach that examines unstructured documents using automated natural language processing. They analyzed which document types contain NFRs, assigned them to NFR categories (e.g. capacity, reliability, and security) and measured how effectively they can identify and classify NFR statements within these documents. For this, they used the Support Vector Machine and Naive Bayes algorithms. In recent years, researchers have started to mine requirements from user comments. Unlike the dataset of this challenge, user reviews are usually short, unstructured and seldom obey grammar and punctuation rules [19], [24].

In this paper we study how accurately we can automatically classify requirements as FRs and NFRs using supervised machine learning. With Support Vector Machine and using lexical features we reach a recall and precision of ~92% for both classes. Furthermore, we assess NFR binary and multi-class classifier for identifying usability, security, operational, and performance NFRs. We achieve a precision up to 93% and a recall up to 90%. With an additional dataset of usability and performance requirements, we also evaluated a classifier for these two NFR classes that use a hybrid training set derived from two different datasets. We aimed to assess whether an additional dataset derived from user comments can help deal with NFR class rarity. Despite that such a dataset is more informal compared to the NFR dataset, we hypothesized that similarities on the lexical level (e.g., adjectives) can make them usable for the aimed purpose.

The paper is structured as follows. Section II overviews the research design. Section III presents and discusses the classification results. Section IV concludes the findings.

2332-6441/17 $31.00 © 2017 IEEE
DOI 10.1109/RE.2017.82

II. Research Design

We first introduce the research questions and then describe the dataset and research methods used in this work.

A. Research Questions

RQ1 How well can we automatically classify requirements as functional (FR) or non-functional requirements (NFR)? What are the most informative features?

RQ2 How well can we automatically classify the four most frequent NFR classes in the dataset: usability, security, operational, and performance?

RQ3 Can we improve the classification accuracy using an additional dataset and various sampling strategies?

B. Dataset

Table I presents an overview of the quality attributes NFR dataset that includes 12 classes and 625 requirements. The column Length shows the average requirements length (i.e., words count) for each class. The table shows that some classes of the NFR dataset are underrepresented. The NFRs Portability (PO) and Fault Tolerance (FT) are very rare in the dataset.

TABLE I: Overview of the "Quality attributes (NFR)" dataset.

Requirements Class	#Requirements	Percent	∅ Length
Functional	255	40,80%	20
Availability (A)	21	3,36%	19
Fault Tolerance (FT)	10	1,60%	19
Legal (L)	13	2,08%	18
Look & Feel (LF)	38	6,08%	20
Maintainability (MN)	17	2,72%	28
Operational (O)	62	9,92%	20
Performance (PE)	54	8,64%	22
Portability (PO)	1	0,16%	14
Scalability (SC)	21	3,36%	18
Security (SE)	66	10,56%	20
Usability (US)	67	10,72%	22
Total	**625**	**100%**	

Potential issues with such datasets are due to the absolute rarity of some concepts as well as the within-class imbalances [2], [17]. The issue with rare instances of a target class can make the classification difficult despite the class imbalances [33]. When the concept itself has a subconcept with limited instances, additional difficulty might arise when classifying a minority concept due to within-class imbalances [18], [26]. We therefore focus only on four NFRs and evaluated an additional dataset for handling class imbalances.

C. Research Methodology

We preprocessed the dataset in three steps. First, we converted the original NFR dataset to a CSV file. Then, we experimented with various classification features and classification algorithms. Third, we pre-calculated the classification features that are independent of the classifier training phase to reduce time needed for cross-validation.

We employed sampling strategies for dealing with imbalances in data, as balanced distribution can improve the classification accuracy [9], [17], [23], [32]. In particular, we employed random under-sampling to achieve a balanced class distribution. That is, in a binary case, the majority class is under-sampled. Additionally, we evaluated an oversampling technique with an own requirements dataset derived from user comments (UC). That is, in a binary case, the minority class is over-sampled. The UC dataset contains usability and performance requirements and is derived from a sample of Amazon software reviews (SR) that was crawled between November 2014 and May 2016. This sample is a stratified random sample of the SR dataset (proportional to the Amazon ratings and software categories) that was peer-coded by humans and contains only codings where at least two coders agreed [22].

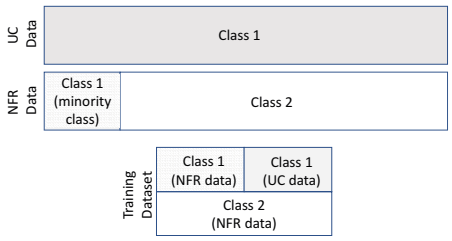


Fig. 1: Illustration of the hybrid training set: composition of the NFR and UC dataset.

To answer RQ1 we derived a sample of two classes from the dataset: the classes denoting non-functional requirements summarized as one class (NF) and the second remaining class F denoting functional requirements. We then randomly under sampled the majority class (i.e., NF in this case) to balance the two classes and obtain the training set. To assess the feature importances, we built a scoring classifier as an ensemble of tree classifiers using a F/NFR training set: Adaptive Boost [11], Extra Tree [13], Gradient Boosting [12], and Random Forest [1]. For each feature, we averaged the sum over all feature importance scores. We ranked the features according to their importances and selected the top 10.

To answer RQ2, we first filtered out the functional requirements from the dataset. Then, we assessed four binary classifier for identifying the four most frequent NFRs in the dataset: usability, security, operational, and performance. We additionally evaluated a multi-class classifier for predicting these four classes.

For RQ1 and RQ2 experiments, we assessed different classifier configurations. In addition to a baseline (manual feature selection), we also assessed the classifiers using an automatic feature selection method employing all feature types that uses statistical scoring functions. Such method reduces the feature space by removing redundant and irrelevant features, while trying not to loose much information [15]. This mitigates overfitting of the classification model and improves its generalizability. We also evaluated learning curves (using F-scores for simplicity reasons) to assess how classifiers benefit from larger trainings sets (i.e., size of labeled data).

For answering RQ3, we used under- and oversampling

TABLE II: Classification features used in the experiments.

Feature	Description
Text n-grams	N-grams of words, for $n \in \{1, 2, 3\}$
POS n-grams	N-grams of part of speech (POS) tags on the word level, based on the Penn Treebank Corpus [25], $n \in \{1, 2, 3\}$
CP unigrams	Unigrams of part of speech (POS) tags on the clause and phrase level (CP), based on the Penn Treebank corpus [25]
%Noun	Fraction of nouns
%Verbs	Fraction of verbs
%Adjectives	Fraction of adjectives
%Adverbs	Fraction of adverbs
%Modal	Fraction of modal verbs
Length	Text length
Subtree count	Sentence syntax sub-trees count
Tree height	Sentence syntax tree height

strategies to build a hybrid training set focusing on classifying usability and performance NFRs. The composition of such a training set illustrates Figure 1.

Class 1 denotes the minority class (e.g., usability) while Class 2 denotes the majority class (e.g., non-usability NFRs). The Class 1 sample of the training set for the classifier is built from the NFR dataset and user comments (UC) dataset which contains sentences addressing usability and performance requirements from software reviews written by users. An example of such a usability requirement is "It's not easy to open a group of RAW files, edit them and then save them as JPEGs"; an example of a performance requirement is "Boot up and shut down seems to take longer now".

We conducted two types of experiments for RQ3. With the first type of experiments we aimed at simulating a case of minority class rarity and how oversampling such class with the UC dataset of the same class (up to the max. of the minority class NFR sample size) affects the classification accuracy. In these experiments we under-sampled the minority class (i.e., Class 1) for the training set emphasizing its rarity. Then we oversampled this class by enlarging it with a random subsample from the UC dataset. The majority class was randomly sampled from the NFR dataset only. The final training set had an equal ratio of both classes.

In the second type of experiments, we aimed to assess whether further expanding the minority class sample with the UC dataset can improve the classification accuracy. We took the whole minority class sample from the NFR dataset and expanded it with the data from the UC dataset. We used several oversampling factors: 0.25, 0.50, 0.75, and 1.00. Again, the majority class was derived only from the NFR dataset. The final training set had an equal ratio of both classes.

D. Configuration of the Classifiers

Table II and Table III list respectively the classification features used for the classifiers and the preprocessing techniques.

We used the Natural Language Toolkit NLTK[1] and the Stanford Parser[2] to preprocess the data. In particular, we applied the tools for text preprocessing (removal of punctuations, removal of stop words, and lemmatization) and the extraction of classification features. We then built supervised classifier, employing the machine learning library SciKit[3].

TABLE III: Preprocessing techniques used in the experiments.

Preprocessing	Description
Stopwords	Removal of stopwords (using SciKit's list of stopwords)
Punctuation	Removal of punctuation
Lemmatization	Grouping of different inflected forms of a word to its base form (lemma)

We employed cross-validation to evaluate prediction performance and obtain a more accurate and robust estimate of real model's performance [27]. We evaluated the F/NFR binary classifier as well as the binary and multiclass NFRs classifier using 10-fold cross validation. The NFR classifier, trained on a hybrid dataset composed of NFR and UC dataset, was validated using a test set derived from the NFR dataset only (RQ3). For the evaluation of this classifier, we used a Monte-Carlo cross validation [34] using 10 iterations with hybrid training sets and the NFR-dataset based test sets. In particular, in each iteration we randomly sampled 10% of the NFR dataset as the test set with an equal number of both classes. This test set estimates the overall performance of the classifier against unseen NFR data. The remaining 90% of the NFR dataset and the UC dataset were used to build the final training set. Thus, in each iteration we trained the classifier using a randomly created hybrid training set (from NFR and UC datasets) and evaluated it against a randomly created test set (from the NFR dataset only). For all experiments we calculated the precision, recall, and the F1 score [10], [35] and report their mean scores.

Beside a baseline configuration using only word features that include bag of ngrams with $n \in \{1, 2, 3\}$ (single words, word bigrams, word trigrams) with filtered stopwords and words lemmatized, we also evaluated the classifiers employing all feature types listed in Table II and using only the k best features for k up to 1000 (in Tables IV and V we report only k up to 500). With this configuration we aimed to represent a rather simple model which we use as a baseline. Simple models have been shown to work better in practice than alternative (complicated) models [16], one of the reasons being that the latter easier fit to noise. The results of the cross-validated F/NFR classifier are summarized in Table IV.

III. Classification Results

In this section we report the classification results for the F/NFR binary classifier (RQ1), the binary and multi-class NFR classifier for US, SE, O, and PE (RQ2), and the NFR classifier trained using a hybrid training set for US and PE (RQ3).

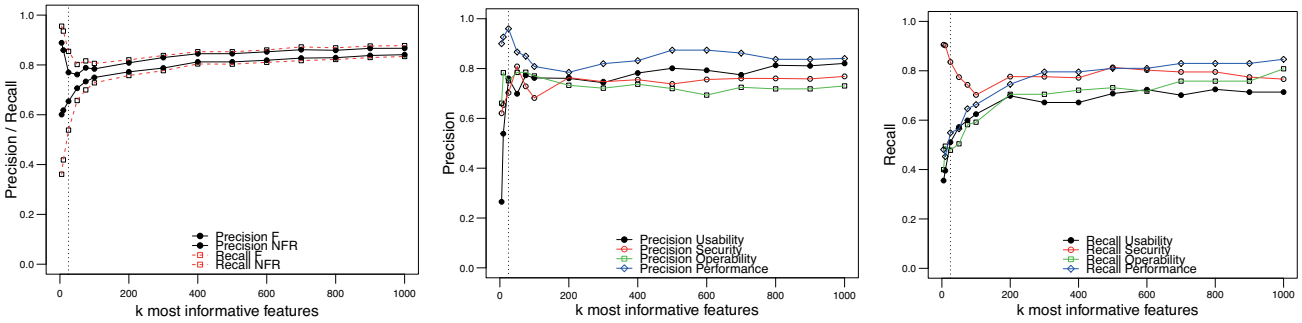
A. F/NFR Binary Classifier

The best performance is achieved using word features without automatic feature selection. Its associated classification model has a reduced feature space compared to the model that includes all feature types. The over-fitting of the associated

[1]http://www.nltk.org/, accessed on June 2017

[2]http://nlp.stanford.edu/software/lex-parser.shtml, accessed June 2017

[3]http://scikit-learn.org/, accessed on March 2017

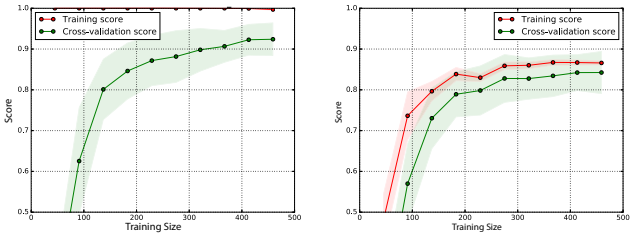


(a) Precision/Recall of the F/NFR binary classifier (b) Precision of the four NFR binary classifiers (US, SE, O, and PE) (c) Recall of the four NFR binary classifiers (US, SE, O, and PE)

Fig. 2: Precision and recall of the various classifiers using the k most informative features.

TABLE IV: 10-fold cross-validated performance metrics of the F/NFR binary classifier.

Classification techniques	F			NFR		
	Precision	Recall	F1	Precision	Recall	F1
Word features						
Word features (without autom. feature selection)	0.92	0.93	0.93	0.93	0.92	0.92
Word features (using autom. selection of k best features)						
k=100	0.86	0.51	0.63	0.65	0.92	0.76
k=200	0.89	0.68	0.77	0.74	0.92	0.82
k=300	0.91	0.74	0.82	0.78	0.93	0.85
k=400	0.91	0.75	0.82	0.79	0.93	0.85
k=500	0.92	0.79	0.85	0.82	0.93	0.87
All feature types (using autom. selection of k best features)						
k=100	0.80	0.81	0.80	0.81	0.80	0.80
k=200	0.84	0.84	0.84	0.83	0.84	0.83
k=300	0.85	0.87	0.86	0.86	0.85	0.85
k=400	0.86	0.86	0.86	0.85	0.87	0.86
k=500	0.88	0.87	0.87	0.87	0.88	0.87



(a) Learning curve using word features w/o autom. feature selection. (b) Learning curve using all feature types (only 200 most informative features).

Fig. 3: Learning curves of the F/NFR binary classifier.

classification model is additionally mitigated with increased training size, which Figure 3a illustrates.

When employing all feature types and using the automated feature selection, we achieve a precision and a recall above 80% using the 100 most informative features. The automatic feature selection reduces over-fitting, as illustrated in Figure 3b. The curves of the training scores and cross-validation scores get closer to each other with larger training size. Figures 3a and 3b show that the model's performance stabilizes with a training size bigger than 400 requirements.

When using automatic feature selection employing only word features we achieve higher recalls for classifying NFRs than when all feature types are employed. The differences between the mean recall scores are statistically significant for all evaluated k values (Welch's t-test, $p < 0.05$). In contrast, when using an automatic feature selection, the recalls for classifying FRs are higher when employing all feature types instead of employing only word features. The differences between the mean recall values are statistically significant for all k except k=500 (Welch's t-test, $p < 0.05$).

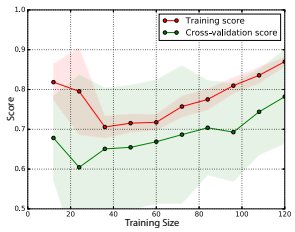
Most Informative Features: We assessed the 10 most informative features of the F/NFR classifier using all feature types and preprocessed word ngrams. Taken together, POS tags were the top scored among the top 10 features followed by the word ngrams and the feature denoting the fraction of modal verbs such as *shall* and *should*. The latter feature reflects the requirements writing convention used in the NFR dataset (*shall* indicates FRs while *should* indicates NFRs).

The single most informative feature is the POS tag CD (i.e., cardinal number). This is not surprising since numbers are mostly used in NFRs to make them concrete and measurable. The second best scored feature was the fraction of modal verbs. The two best ranked word features were 'player' and 'standard' that appear most often in FRs and NFRs respectively. Other informative single words were 'display' and 'shall' - both mostly indicating FRs. A FR having such a feature is "The system *shall* allow modification of the display".

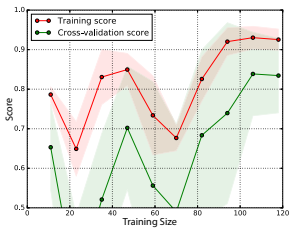
The two best ranked POS trigram were NNP-NNP-MD (i.e., two proper nouns, and modal) and VB-VBN-IN (i.e., verb, verb in past participle, and preposition). The first trigram appears mostly in FRs (e.g., in "All actions that modify an existing *dispute case must* be recorded in the case history"). The second trigram (i.e., VB-VBN-IN) predominantly appears in NFRs "The report will *be reviewed for* auditing purposes".

B. Binary and Multi-class NFR Classifier

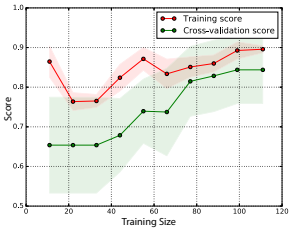
We assessed four NFR binary classifier for the NFRs usability (US), security (SE), operational (O), and performance (PE) as well as one multi-class classifier for all four NFR classes. The precision, recall, and F-scores are summarized in Table V. For the classifiers employing an automated feature



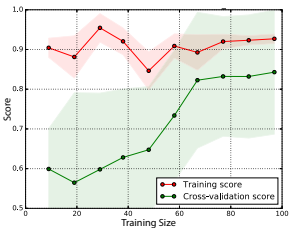
(a) US classifier learning curve.



(b) SE classifier learning curve.



(c) O classifier learning curve.



(d) PE classifier learning curve.

Fig. 4: Learning curves of the NFR binary classifiers for US, SE, O, and PE using all feature types and the 200 most informative features.

TABLE V: 10-fold cross-validated performance metrics of the binary and multi-class classifier techniques. Bold values represent the highest score for the corresponding accuracy metric per NFR class.

Classification techniques	**Usability (US)**			**Security (SE)**			**Operational (O)**			**Performance (PE)**		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Binary Classification												
Word features (w/o feature selection)	0.81	0.85	0.82	0.91	0.90	0.88	0.72	0.75	0.73	0.93	0.90	0.90
All feature types (using autom. selection of k best features)												
k=50	0.7	0.57	0.61	**0.81**	0.77	**0.74**	**0.78**	0.5	0.57	0.87	0.57	0.67
k=100	0.76	0.62	0.66	0.68	0.7	0.66	0.77	0.59	0.65	0.81	0.66	0.72
k=200	0.76	0.7	0.71	0.76	**0.78**	0.73	0.73	0.7	0.7	0.78	0.75	0.75
k=300	0.74	0.67	0.69	0.75	**0.78**	0.73	0.72	0.7	0.69	0.82	0.8	0.79
k=400	0.78	0.67	0.71	0.76	0.77	0.73	0.74	0.72	0.71	0.83	0.8	0.8
k=500	**0.80**	**0.71**	**0.74**	0.74	0.81	**0.74**	0.72	**0.73**	**0.71**	0.87	**0.81**	**0.82**
Multi-Class Classification												
Word features (w/o feature selection)	0.65	0.82	0.70	0.81	0.77	0.75	0.81	0.86	0.82	0.86	0.81	0.80
All feature types (using automated selection of k best features)												
k=50	0.49	**0.68**	0.55	0.6	0.5	0.39	0.42	0.47	0.33	0.85	0.53	0.63
k=100	0.55	**0.68**	0.59	0.6	0.39	0.46	0.41	**0.65**	0.48	0.88	0.6	0.7
k=200	0.63	0.64	0.6	0.63	0.48	0.53	0.43	0.6	0.48	0.77	0.73	0.73
k=300	0.63	0.64	0.6	0.61	**0.56**	0.55	0.45	0.57	0.47	0.8	0.68	0.71
k=400	0.63	0.65	0.6	0.63	**0.56**	**0.56**	**0.51**	0.62	0.53	0.86	**0.74**	**0.77**
k=500	**0.70**	0.66	**0.64**	**0.64**	0.53	**0.56**	0.47	0.62	**0.51**	0.81	0.74	0.76

selection, we plot the precision curves in Figure 2b and the recall curves in 2c. The NFR performance classifier achieved the overall highest precision with a smaller number of features compared to the other three evaluated NFR classifier. NFR security classifier achieved the highest recall and required less than 50 features for a recall above 0.70%, compared to the NFR classifiers of usability and performance requiring more than 100 features for the same recall.

The binary classifier had a statistically significant higher recall for classifying usability with $k \in \{300, 400, 500\}$, performance with k=300, and security with $k \in \{100, 200, 500\}$ (Welch's t-test, $p < 0.05$).

Figure 4 shows the learning curves for the four NFR binary classifiers using SVC and 200 most informative features. For O and PE the over-fitting gets less likely by larger training sample.

C. NFR binary Classifier with Hybrid Training Set

We conducted few sets of experiments with classifiers for US and PE trained on a hybrid dataset, that we derived from the NFR dataset and UC dataset and validated against a NFR-dataset based test set. With C1 we denote the minority class, i.e., one of these two classes.

First, we simulated the case of a NFR class rarity by random under-sampling the minority class C1 of the NFR dataset (of size $C1_{min}$) and then enlarging the same class using the UC dataset (up to $C1_{min}$). The results summarized in Table VI show that using the UC dataset we achieve similar classification accuracy for these two classes, as when using NFR dataset only. In particular, the NFR classifier for US trained on a hybrid set with 33% data from UC dataset (Table VI) achieves similar precision and recall values, as trained using NFR dataset only (Table V). This demonstrates the usefulness of the UC dataset for handling class imbalances.

TABLE VI: Accuracy of the NFR binary usability (US) and performance (PE) classifiers using a hybrid training set (with C1 and C2 classes) having a reduced size of the minority class (C1) and being enlarged with the UC dataset.

C1 size NFR sample	**C1 size UC sample**	**Usability (US)**			**Performance (PE)**		
		Precision	Recall	F1	Precision	Recall	F1
Baselines (NFR dataset, equal class ratio)							
33% $C1_{min}$	-	0.60	0.80	0.64	0.70	0.66	0.64
66% $C1_{min}$	-	0.80	0.85	0.81	0.89	0.88	0.86
Oversampling using UC dataset (equal class ratio)							
33% $C1_{min}$	66% $C1_{min}$	0.58	0.66	0.58	0.93	0.52	0.64
66% $C1_{min}$	33% $C1_{min}$	0.83	0.83	0.83	0.93	0.74	0.81

Second, we over-sampled the minority class C1 of the NFR dataset with the UC dataset using several factors. The results are summarized in Table VII and show that we were not able to significantly improve the classification accuracy.

TABLE VII: Accuracy of the NFR binary usability (US) and performance (PE) classifiers using a training set enlarged with the UC dataset.

Oversampling factor	Usability (US)			Performance (PE)		
	Precision	Recall	F1	Precision	Recall	F1
25%	0.80	0.86	0.81	0.91	0.84	0.87
50%	0.80	0.88	0.82	0.93	0.90	0.91
75%	0.87	0.75	0.78	0.91	0.90	0.90
100%	0.92	0.73	0.79	0.92	0.86	0.88

IV. Conclusion

In this paper, we take up the second RE17 data challenge on the "Quality attributes (NFR)" dataset. Using lexical, syntactical, and meta-data feature types we assessed how well we can predict certain classes of the provided NFR dataset.

We evaluated a F/NFR binary supervised classifiers that automatically classify requirements as functional (FR) or non-functional (NFR). We found that part of speech tags are among the most informative features, with cardinal number being the best single feature. With manually selected features employing bag of words, bigrams and trigrams, and filtering stopwords and punctuation, we achieve precision and recall of ~92%. Using automatic feature selection and employing only word features we achieve higher recalls for classifying NFRs than when employing additionally syntax and meta-data features, but lower precision.

We assessed supervised NFR classifiers to automatically identify the different types of NFRs, focusing on: usability, security, operational, and performance. Using only word features without feature selection we achieve precision and recall ranging between ~72% and ~90% with the binary classifiers. Using the 200 most informative features (~2% of the feature space) we achieve a precision and recall above 70% for these four NFR classes. Depending on the usage scenarios, precision or recall might be more relevant [30].

Finally, we demonstrate the usefulness of a dataset derived from user comments for handling class imbalances.

Acknowledgment

This work is partly funded by the H2020 EU research project OPENREQ (ID 732463).

References

[1] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, Oct. 2001.

[2] R. Caruana. Learning from imbalanced data: Rank metrics and extra tasks. In *Proc. Am. Assoc. for Artificial Intelligence (AAAI) Conf*, 2000.

[3] A. Casamayor, D. Godoy, and M. Campo. Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. *Information and Software Technology*, 52(4), 2010.

[4] L. Chung and B. A. Nixon. Dealing with non-functional requirements: three experimental studies of a process-oriented approach. In *17th International Conference on Software Engineering, ICSE 1995*. IEEE.

[5] J. Cleland-Huang, A. Czauderna, M. Gibiec, and J. Emenecker. A machine learning approach for tracing regulatory codes to product specific requirements. In *Proc. of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*. ACM, 2010.

[6] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc. The detection and classification of non-functional requirements with application to early aspects. In *Requirements Engineering, 14th IEEE International Conference*. IEEE, 2006.

[7] A. M. Davis. *Software requirements: objects, functions, and states*. Prentice-Hall, Inc., 1993.

[8] J. Eckhardt, A. Vogelsang, and D. M. Fernández. Are non-functional requirements really non-functional?: an investigation of non-functional requirements in practice. In *Proc. of the 38th International Conference on Software Engineering*. ACM, 2016.

[9] A. Estabrooks, T. Jo, and N. Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational intelligence*, 20(1), 2004.

[10] W. B. Frakes and R. Baeza-Yates. Information retrieval: data structures and algorithms. 1992.

[11] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, Aug. 1997.

[12] J. H. Friedman. Stochastic gradient boosting. *Comput. Stat. Data Anal.*, 38(4):367–378, Feb. 2002.

[13] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[14] M. Glinz. On non-functional requirements. In *Requirements Engineering Conference, 2007. RE'07. 15th IEEE International*. IEEE, 2007.

[15] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3, 2003.

[16] D. J. Hand and K. Yu. Idiot's bayes not so stupid after all? *International statistical review*, 69(3), 2001.

[17] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9), 2009.

[18] R. C. Holte, L. Acker, B. W. Porter, et al. Concept learning and the problem of small disjuncts. In *IJCAI*, volume 89. Citeseer, 1989.

[19] C. Iacob and R. Harrison. Retrieving and analyzing mobile apps feature requests from online reviews. In *Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on*. IEEE, 2013.

[20] E. Knauss, D. Damian, G. Poo-Caamaño, and J. Cleland-Huang. Detecting and classifying patterns of requirements clarifications. In *20th IEEE International Requirements Engineering Conference*. IEEE, 2012.

[21] G. Kotonya and I. Sommerville. *Requirements engineering: processes and techniques*. Wiley Publishing, 1998.

[22] Z. Kurtanović and W. Maalej. Mining user rationale from software reviews. In *Proc. of the 25rd IEEE International Requirements Engineering Conference*, 2017.

[23] J. Laurikkala. Improving identification of difficult small classes by balancing class distribution. In *Conference on Artificial Intelligence in Medicine in Europe*. Springer, 2001.

[24] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik. On the automatic classification of app reviews. *Requirements Engineering*, 2016.

[25] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, June 1993.

[26] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1), 1986.

[27] C. Schaffer. Selecting a classification method by cross-validation. *Machine Learning*, 13(1), 1993.

[28] 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*. IEEE, 2013.

[29] I. Sommerville and P. Sawyer. *Requirements Engineering: A Good Practice Guide*. John Wiley & Sons, Inc., 1st edition, 1997.

[30] L. T. Su. The relevance of recall and precision in user evaluation. *Journal of the American Society for Information Science*, 45(3), 1994.

[31] R. B. Svensson, T. Gorschek, and B. Regnell. Quality requirements in practice: An interview study in requirements engineering for embedded systems. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2009.

[32] G. Weiss and F. Provost. The effect of class distribution on classifier learning: an empirical study, tech. Technical report, Re ML-TR-44, Department of Computer Science, Rutgers University, 2001.

[33] G. M. Weiss. Mining with rarity: a unifying framework. *ACM Sigkdd Explorations Newsletter*, 6(1), 2004.

[34] Q.-S. Xu and Y.-Z. Liang. Monte carlo cross validation. *Chemometrics and Intelligent Laboratory Systems*, 56(1), 2001.

[35] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proc. of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1999.