Requirement Engineering and ML

# Introduction:

Requirement Engineering (RE) is the key phase in the development of the software. RE is the first complete phase in the proposed literature e.g. waterfall model. As the time passed, software development models get a different vision. Now, the RE is part of software life cycle, from the beginning until the end of the project e.g. agile model. In simple words, determining and managing user needs related to the software and hardware is known as requirement engineering.

Software systems are developed over millions of lines of code, number of modules and documents. The primary goal of the software system is to satisfy users by developing the software that can meet up their needs and expectations. This goal is achievable by applying different methodologies and engineering techniques. One of the key factor is to understand and identify the needs of users, also known as, software requirements. Software requirement engineering is the process that helps to determine the requirements in a systematic way to know what functionalities the targeted system should have to fulfil user needs. Formally RE is defined as [1]:

“Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families."

Software requirements play a key role in the success of a project. In USA, a survey was conducted over 8380 projects by 350 companies to know the project failure rates. The report overall results showed only 16.2% projects were completed successfully and one-half (52.7%) of project were challenged and completed with partial functionalities, time delays and over budgeted. Whereas, 31% of the projects were never completed. The main cause told by the executive managers was the poor requirement. The major problem were the lack of user involvement (13%), requirements incompleteness (12%), changing requirements (11%), unrealistic expectations (6%), and unclear objectives (5%). [6]

Software requirement engineering has mainly four phases; requirement elicitation, requirement analysis, requirement documentation and requirement verification [2]. Requirement elicitation [3] [4] helps to understand the stakeholders needs e.g. what features he wants in the software? Requirement elicitation techniques are mostly derived by the social sciences, organizational theory, knowledge engineering and practical experience. For requirements elicitation, different techniques exist in the literature that includes interviews, questioners and ethnography etc. Requirement analysis [5] is the next step after requirement elicitation. In this phase, software requirements are analyzed to check the conflicts and consistency of requirements. It is also made sure that the requirements are clear, complete and consistent. Furthermore, the agreed requirements are documented. This documentation has a clear and precise definition of the system functionalities. It also acts as an agreement between stakeholders and developers. These functionalities and requirements are documented usually as diagrams, mathematically formulae or natural languages. These documents are used until the end of the projects [ref required].

System requirements are classified into functional requirements (FR) and non-functional requirements (NFR). Functional requirements are the system requirements that include the main features and characteristics of the desired system. Non-functional requirements are the system properties and constraint [7]. NFRs set the criteria for judging the operation of the system e.g. performance, availability and reliability etc.

On the other hand, machine learning (ML) is an emerging field of this era. Artificial intelligence (AI) is a well-known and mature field in computer science domain. Machine learning is a part of AI. Machine learning helped to solve complicated and hard problems efficiently (can you give some examples here?). ML mainly relies on the data and its algorithms learns from the existing data and predict the unseen problem solution. Its learning process has a great inspiration from human learning i.e. learning from the examples. ML algorithms have proved to have a great impact in different fields e.g. business, medical, software engineering, computer security, data and communication networks etc. For the leaning process algorithm, another key factor is the feature. Features correspond to the characteristics of the learning and provide the base to the algorithm. In addition, features help to abstract the complexities of the information provided for the learning and shortening training times. In short, features help to reduce time and complexity of the models for learning.

ML algorithms are divided into two major categories; supervised learning and unsupervised learning [7]. In addition, the third category is reinforcement learning added later in ML. In supervised learning, data set is labeled that means it has some example with the defined features and output. New input or query will be predicted from the learning of the labeled data. All regression and classification algorithms come under the umbrella of supervised learning e.g. Logistic Regression, Decision Trees (DT), Support Vector Machine (SVM), Nearest Neighbors (NN), Naive Bayes, Random Forest and Artificial Neural Network (ANN) etc. [8]. On the other hand, unsupervised learning is from unlabeled data. It covers all clustering algorithms e.g. k-means clustering and hierarchical clustering etc. [Ref Required].

# Literature Review:

This section describes the current trends in RE and ML. It will show a few problems that have been solved or automated using ML.

**Requirement classification:**

The first and highlighted challenged problem in RE is the requirement analysis. It deals with the classification of functional and non-functional requirements. Requirements are usually written in the natural language. These documents contain hundreds of requirements. It is hard and time consuming to classify them manually. These classifications are not limited to only FR and NFR, but also to the sub categories of NFR and quality attributes. This problem area can be categorized on the base of used datasets in the literature. Three different domains are part of this category i.e.

* App stores
* Social Media
* Specified Application or Software

Millions of users are sharing their reviews on app stores after downloading and using the app. They just not only rate the apps but also write about the liked and disliked features. It is not easy to get the requirements in such a complicated data. New features identification, classification of FR and NFR, and summaries of the reviews for the improvements of app is done with ML [12-14]. For the automated classification of FR and NFR, [12] used a total of 932,338 online reviews of the 40 top paid and free apps on app stores from top 10 different categories. Semi supervised algorithm self- training, RASCO, Rel-RASCO for self-labelling is being used. This semi supervision technique overcome the manual annotation problem and showed that only small amount of labelled data achieved high accuracy. Naïve Bayes classification achieved the best results out of the kNN, C4.5 and SMO. Another proposed study on classification of FR and NFR was being proposed in [13], with experimentation on 6696 raw user reviews from iBook and 4400 raw user reviews from WhatsApp. It used the different concept i.e. augmentation of user reviews. It is effective to improve user reviews classification results by adding textual semantics to the sentences. The user reviews are augmented by several similar words for better classification of results. The bagging algorithm showed best results in comparison to Naïve Bayes and J.48. A classification method is produced in [14] for identifying bug reports and feature requests from user reviews. Total 146,057 reviews for 40 apps were collected from apple stores and google play stores. For further experimentation, 4400 reviews were selected. The proposed model showed the upwards of 70 percent precision and 80 percent recall could be obtained using multiple binary classifiers, as an alternative to a single multiclass classifier. For the classification, binary Naive Bayes algorithm is used. The results showed that the commonly used NLP techniques, stop word removal and lemmatization, and could negatively affect the performance of this classification task.

In the paper [15], main goal is to transform online reviews in to evolutionary requirements. Karplersky internet security 2011 from Amazon and mobile app of Tune-In Radio Pro V3.6 from the app store data is taken as a dataset. The characteristic analysis of the reviews is considered for the automated task analysis and used relation-based propagation approach (SRPA) technique for the identification of opinion about common software features. Each set manually labelled the potential software feature, opinion and the polarities in the reviews, and then classify the reviews on the base of relevant opinion semantics. For clustering, the opinion expression network algorithm Grivan Newman is used in the proposed methods S-GN. The input of the J-K means and S-K means are set of the same number of clusters produced by the S-GN. The GN algorithm produced the optimized number of the cluster considering the global network topology that reduced the cluster containing mixed categories. For the second problem, system helped developer and proposed a set of related evolutionary requirement of the system. The second problem in this study made it more interesting. By adding polarization of the reviews and helping the developer to know the related requirements. It is actually an intersection of the idea discussed above and give a new insight to this domain.

Social media e.g. Twitter and Facebook have become popular platforms to gather the requirements from user posts. Users are sharing their new features requests, feedback and bug report via tweets. Tweets as a big dataset used ML to classify theses tweets into meaningful categories e.g. new requests and bugs etc. [16-17]. These classifications and the information from theses tweets helped industry to know the user feedback for the software improvements. ALERTme [16] approach proposed for classifying, grouping and ranking tweets in software evolution process. For this, a total of 68108 tweets i.e. collection of two-month tweets of Spotify, Dropbox and Slack software dataset are used. The output is binary classification i.e. improvement request or other. For the automated classification, supervised learning algorithm Multinomial Naïve Bayes is used. Further improvement requests are considered for the grouping which helped to sort the request and summarize them accordingly. Reduce human effort to analyze each tweet for eliciting the requirements and knowing the issues in the software is a key contribution. As a last step, these summaries and tweets are ranked on the basis of high worthy tweets. A drawback of this study is the high number of manual annotations for labeling the data as request or others. In addition, the majority voting scheme is being used to solve the disagreements. Three annotators did this process and it took around 13.5 hours for each annotators to complete the task. The next study [17] classify and summarize the tweets. Total 4000 randomly selected tweets from ten different software are selected. The proposed model classify them into bug, requirement and spam using Naïve Bayes and SVM. The results showed that 50% of data contained useful technical feedback and achieved an average classification F1 of 72% using SVM that is better than the state of the art in literature [16]. The reason is the feedback dedicated to technical stakeholders i.e. developer related tweets are focused and analyzed. Tweets dataset is labelled manually. Different techniques with VSM (is it SVM?) and NB as a preprocessing e.g. stop word removing, sentimental, stemming, and Bag of words implemented. However, the results showed that these parameters did not improve to help the results of ML algorithm for classification of the tweets. Unlike the political tweets which are polarized and carry emotions, software tweets are neutral in nature (this line should be somewhere in the middle of paragraph).

The third category is dealing with requirement classification using the software product data i.e. written in SRS. This data is composed of different software function and nonfunctional requirements. The nonfunctional requirements have subcategories that include availability, fault tolerance, legal, look and feel, maintainability, operational, performance, portability, scalability, security and usability etc. This dataset is provided by the Requirement Engineering (RE) conference and named as Quality attributes (NFR) dataset. The size of the total data set has 625 requirements with 225 FR and rest with NFRs subcategories. For the automated classification of FR and NFR and identification of the subcategories of NFR [18], support vector machine (SVM) algorithm is used. The data is not equally distributed with some NFR with lowest number ignored. Data is under sampled, and for solving the under sampling problem external data i.e. user comment dataset from the amazon is added and hybrid approach is proposed with the new dataset. This dataset contains performance and usability requirements. For the identification of specific NFRs, proposed methodology achieve the highest precision and recall for security and performance NFRs with ~92% precision and ~90% recall. Two main goals are targeted [19], first is classifying the FR and NFR, and second is identification of NFR category. The data is pre-processed as a first step. Feature co-occurrence and regular expression is used to increase the weight of influential words used in NFR. The supervised learning algorithm J.48 DT is used for the classifying the FR and NFR. For achieving the categorization or classification of NFR, topic modeling using unsupervised algorithm LDA and BTM is applied. For generation of the topic result showed BNB worked better out of clustering, k-means, LDA, BTM. Another study for solving the same problem with NFR dataset additionally security dataset was conducted [20]. Software requirements are classified and the focus is more on the security related requirements. ML algorithm CNN with specific setting in Tensor-Flow helped to achieve the goal and better results. In all these classification problems, human input is involved for the annotation of the requirements. Semi-supervision requires less human effort in labeling requirements than fully supervised methods. The semi-supervised approach [21] using Naïve Bayes resulted in accuracy rates above 70%, considerably higher than the results obtained with supervised methods using standard collections of documents.

# Problem Statement:

Conventionally, software requirement elicitation and analysis is only limited to the meetings, interviews and documented data etc. All the tasks are performed manually and need more effort and time. With the recent data trend from different sources, the user satisfaction and opinion is more integrated into the industry. Recent studies have shown that the user analytics tools and techniques are helping developers and practitioner to deal with the large numbers of user feedback by filtering, classifying, and summarizing them, to decide what requirements and features they should add, change, or eliminate. Different ML algorithms have used for making it automated with different settings and parameters. Most of the studies are using the app data or Twitter data. Recent study [] has shown that data from Twitter and app stores reviews can complement each other. The use of the similar apps and tweets can give more knowledge and help to get the better results. The effect of these results in the real project is missing and not explained in the studies:

1. Reusability
2. Data from appstore Tweets and amazon software review does not exist
3. Which stakeholder has a particular interest in which topic for assigning the weight to the tweets or etc.

When the requirements engineers use the classifier to pull out and group comments on issues, those posts by people with higher expertise values for the topic might be given more weight. In fact, tools could be developed that would sort the comments by the expertise of the poster.

\*\* these security problem if identified on the right level of abstraction can be reusable across the multiple systems even as a set to meet the same security objective. Pattern and similarities in grammar or phrasing the requirement may exist and allow the security requirement to be reused across multiple software with minor tweak to content. Paper hidden in plain sight Reusability is missing in the above studies, if reusable the results and studies of there impactFor judging the expertise the question can be directed to the related person. Thoes who answered and solve the problem will scored high and and those who did not answere or poorly answered will score low and it alos define the expertise level. Software requirement elicitation and analysis was only limited to the meetings, interviews and documented data etc. All the tasks were manual and needed more effort and time. With the recent data trend from different sources the user satisfaction and opinion is more integrated into the industry. Recent Study has shown that user analytics tools and techniques are helping developers and practitioner to deal with the large numbers of user feedback by filtering, classifying, and summarizing them, to decide what requirements and features they should add, change, or eliminate. Different ML algorithm has used for making it automated with different settings and parameters. Most of the studies are using the app data or twitter data. Recent study [22] shows that data from twitter and app stores reviews can complement each other. The use of the similar apps and tweets can give more knowledge and help to get the better result. The effect of these results in the real project is missing and not explained in the studies.

RQ1: Can advance classifier perform better than the algorithm used in the existing literature.

Description: Review the performance, complexity and factor effecting the results?

RQ2: What would be the impact of using semi supervised and unsupervised learning in contrast to supervised learning?

RQ:3 How social network sites (twitter, facebook, LinkedIn) and online discussion forums (appstores, amazon, google stores) reviews complement for developing the new app or software?

The importance of the end user involvement in today’s software is so important and highlighted in the literature. However, the reusability of this large dataset is not discussed and explored completely. The addition of the social media data in the software development process in early stages can add globalist and heterogeneity perspective. The benefit of the large dataset is to gather information about specific category and see what requirement user wished to have in both functional and nonfunctional perspective. Implement this idea to a case study to know the impact and enhancement of requirement that were gained by the additional data.

RQ: 4 How useful are the social network sites (twitter, facebook, LinkedIn) and online discussion forums (appstores, amazon, google stores) reviews for requirement elicitation and what works better?

RQ:5 How accurately the information obtained from various sources be summarized?

RQ:6 What would be the best candidates tool for adding our proposed tool as a plugin?

How we can integrate the proposed tool to the current exisiting requirement elicitation tool?

# Deploying a Template and Pattern Library for Improved Reuse of Requirements Across Projects

**References:**

[1] Zave, P. (1997). Classification of Research Efforts in Requirements Engineering. ACM Computing Surveys, 29(4): 315-321.

[2] [Kotonya, Gerald](http://www.research.lancs.ac.uk/portal/en/people/gerald-kotonya(47dfa00c-3154-4732-8969-24eb56e8768b).html); Sommerville, Ian (1998). [Requirements Engineering: Processes and Techniques.](http://www.research.lancs.ac.uk/portal/en/publications/requirements-engineering-processes-and-techniques(f9a48c9f-e3b1-4373-a7f4-72491f858091).html) Chichester : John Wiley & Sons, 282 p.

[3]Goguen, J.A., Linde, C. (1993): Techniques for Requirements Elicitation, Proceedings of the IEEE International Symposium on Requirements Engineering, pp. 152-164, January 4-6, San Diego, CA. 24.

[4] Zowghi D., Coulin C. (2005) Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In: Aurum A., Wohlin C. (eds) Engineering and Managing Software Requirements. Springer, Berlin, Heidelberg,

[5] Nuseibeh, Bashar, and Steve Easterbrook(2000). "Requirements engineering: a roadmap." In Proceedings of the Conference on the Future of Software Engineering, pp. 35-46. ACM.

[6] The Standish Group (2014), “Software Chaos" https://www.projectsmart.co.uk/white-papers/chaos-report.pdf.”.

[7] A.M. Davis. “Software Requirements: objects, function, and states.” Prentice-Hall. Inc., 1993.

[8] Lison, P. (2015). “An introduction to machine learning”

[9] Kotsiantis, S.B., Zaharakis, I. and Pintelas, P. (2007). “Supervised machine learning: A review of classification techniques”. Emerging artificial intelligence applications in computer engineering, 160, pp.3-24.

[10]<http://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/>

[11] Du Zhang, Jefferey J.P.Tsai (2003). “Machine Learning and Software engineering”. Software Quality Journal, 11, 87–119.

[12] Deocadez, R., Harrison, R., & Rodríguez, D. (2017). “Automatically Classifying Requirements from App Stores: A Preliminary Study.” IEEE 25th International Requirements Engineering Conference Workshops (REW), 367-371.

[13] Mengmeng Lu and Peng Liang (2017). “Automatic Classification of Non-Functional Requirements from Augmented App User Reviews”. In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (EASE'17). ACM, New York, NY, USA, 344-35

[14] W. Maalej and H. Nabil, “Bug report, feature request, or simply praise? on automatically classifying app reviews,” in Proc. IEEE 23rd Int. Requirements Eng. Conf., 2015, pp. 116–125.

[15] Jiang W., Ruan H., Zhang L., Lew P., Jiang J. (2014). “For User-Driven Software Evolution: Requirements Elicitation Derived from Mining Online Reviews” In Advances in Knowledge Discovery and Data Mining. PAKDD 2014. Lecture Notes in Computer Science, vol 8444. Springer, Cham.

[16] E. Guzman, M. Ibrahim and M. Glinz(2017) "A Little Bird Told Me: Mining Tweets for Requirements and Software Evolution," *2017 IEEE 25th International Requirements Engineering Conference (RE)*, Lisbon, 2017, pp. 11-20

[17] G. Williams and A. Mahmoud, "Mining Twitter Feeds for Software User Requirements," 2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, 2017, pp. 1-10.

[18] Z. Kurtanović and W. Maalej (2017). "Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning," *2017 IEEE 25th International Requirements Engineering Conference (RE)*, Lisbon, 2017, pp. 490-495.

[19] [Zahra Shakeri Hossein Abad](https://arxiv.org/search?searchtype=author&query=Zahra+Shakeri+Hossein+Abad), [Oliver Karras](https://arxiv.org/search?searchtype=author&query=Oliver+Karras), [Parisa Ghazi](https://arxiv.org/search?searchtype=author&query=Parisa+Ghazi), [Martin Glinz](https://arxiv.org/search?searchtype=author&query=Martin+Glinz), [Guenther Ruhe](https://arxiv.org/search?searchtype=author&query=Guenther+Ruhe), [Kurt Schneider](https://arxiv.org/search?searchtype=author&query=Kurt+Schneider) (2017). “What Works Better? A Study of Classifying Requirements”. [**arXiv:1707.02358**](https://arxiv.org/abs/1707.02358)**.**

[20]A. Dekhtyar and V. Fong, (2017). "RE Data Challenge: Requirements Identification with Word2Vec and TensorFlow," 2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, 2017, pp. 484-489.

[21] Agustin Casamayor, Daniela Godoy, Marcelo Campo(2010). “Identification of non-functional requirements in textual specifications: A semi-supervised learning approach, Information and Software Technology”. Volume 52, Issue 4, 2010, Pages 436-445, ISSN 0950-5849.

[22] M. Nayebi, H. Cho, H. Farrahi and G. Ruhe, "App Store Mining Is Not Enough," *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, Buenos Aires, 2017, pp. 152-154.

[146] C. Iacob and R. Harrison, (2013) “Retrieving and analyzing mobile apps feature requests from online reviews,” in Proc. 10th Working Conf. Mining Software Repositories, pp. 41–44.

[178] S. Panichella, A. D. Sorbo, E. Guzman, A. Visaggio, G. Canfora, and H. Gall, “How can I improve my app? classifying user reviews for software maintenance and evolution,” in Proc. 31st IEEE Int. Conf. Softw. Maintenance Evol., 2015, pp. 281–290.