ReFAIR: Toward a Context-Aware Recommender for Fairness Requirements Engineering

Carmine Ferrara
University of Salerno
Fisciano (SA), Italy
carferrara@unisa.it

Francesco Casillo
University of Salerno
Fisciano (SA), Italy
fcasillo@unisa.it

Carmine Gravino
University of Salerno
Fisciano (SA), Italy
gravino@unisa.it

Andrea De Lucia
University of Salerno
Fisciano (SA), Italy
adelucia@unisa.it

Fabio Palomba
University of Salerno
Fisciano (SA), Italy
fpalomba@unisa.it

ABSTRACT

Machine learning (ML) is increasingly being used as a key component of most software systems, yet serious concerns have been raised about the fairness of ML predictions. Researchers have been proposing novel methods to support the development of fair machine learning solutions. Nonetheless, most of them can only be used in late development stages, e.g., during model training, while there is a lack of methods that may provide practitioners with early fairness analytics enabling the treatment of fairness throughout the development lifecycle. This paper proposes ReFair, a novel context-aware requirements engineering framework that allows to classify sensitive features from User Stories. By exploiting natural language processing and word embedding techniques, our framework first identifies both the use case domain and the machine learning task to be performed in the system being developed; afterward, it recommends which are the context-specific sensitive features to be considered during the implementation. We assess the capabilities of ReFair by experimenting it against a synthetic dataset—which we built as part of our research—composed of 12,401 User Stories related to 34 application domains. Our findings showcase the high accuracy of ReFair, other than highlighting its current limitations.

CCS CONCEPTS

• **Software and its engineering → Requirements analysis**.

KEYWORDS

Software Fairness; Machine Learning; Requirements Engineering.

ACM Reference Format:
Carmine Ferrara, Francesco Casillo, Carmine Gravino, Andrea De Lucia, and Fabio Palomba. 2024. ReFAIR: Toward a Context-Aware Recommender for Fairness Requirements Engineering. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3597503.3639185

ICSE '24, April 14–20, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0217-4/24/04.
https://doi.org/10.1145/3597503.3639185

1 INTRODUCTION

In today's software production, data-driven machine learning (ML) algorithms are more and more employed to support decision-making activities performed by individuals and companies [65], other than to automate repetitive tasks, reducing human's workload [51]. Successful applications have been showcased in multiple domains [23] like loan management [46], hiring decisions [43], healthcare [2], biology [56], and more.

Despite the benefits brought by ML, researchers have been reporting on the implications that those algorithms may have on ethics and fairness [38]: the reliance on historical data may lead an ML algorithm to gather biased knowledge about the relations ruling a phenomenon, which might lead to unfair predictions and recommendations that, in turn, might reiterate discrimination and injustice [3]. As such, the definition of methods and tools able to reduce risks due to ethical concerns represent a key challenge [3, 18].

Recently, the *Software Engineering for Artificial Intelligence* (SE4AI) research community has been actively working on this matter, arguing the need for novel engineering processes to treat fairness throughout the software lifecycle. However, operationalizing this need is complex because of the intrinsic nature of fairness. On the one hand, fairness strictly depends on the application domain and the specific task an ML system is designed for [39, 48], e.g., a feature may be sensitive in one context and not in another. On the other hand, fairness represents a multi-faceted aspect, and, indeed, multiple definitions targeting various perspectives of software fairness have been proposed in literature [60].

At the current stage, most of the existing approaches focus on avoiding discrimination from data. For instance, Chakraborty et al. [14] proposed Fair-SMOTE, an oversampling algorithm able to rebalance training data according to sensitive attribute groups. On a similar note, other researchers attempted to analyze the best data preprocessing actions to keep fairness under control [6], how to diversify data to reduce fairness concerns [44], and optimize training data to balance fairness and accuracy [17, 29]. At the same time, automated fairness testing procedures have been defined [25].

Recognizing these advances, Soremekun et al. [54] pointed out the need for novel requirements engineering techniques that may let practitioners be aware of sensitive features since the project inception. Those instruments may have a fundamental impact on practice: being able to provide early recommendations on sensitive features, they may inform the entire ML engineering lifecycle,

possibly making all the involved stakeholders aware of the most suitable bias mitigation strategies to put in place to reduce risks due to unfairness. In addition, the outcome of such a recommender may complement existing bias mitigation approaches, empowering the whole ML pipeline by making it more fairness-aware.

In this paper, we perform the first step toward this objective and propose ReFair, an automated requirements engineering framework that employs natural language processing (NLP) and word embedding techniques to classify sensitive features from User Stories (USs).[1] We design ReFair to be *context-aware*. As such, it can classify application domains and ML tasks to be implemented before recommending the sensitive features to consider, hence addressing the needs brought by the intrinsic nature of software fairness.

To experiment with ReFair, we create a *synthetic* dataset of 12,401 ML-related USs pertaining to 34 different application domains. The results of our study showcase the capabilities of ReFair, which can (1) classify application domain and ML tasks within the USs with an F1-score of 97% and 90%, respectively, and (2) recommend sensitive features with high precision.

To sum up, our paper provides three key contributions:

(1) ReFair, a novel context-aware automated framework to support fairness requirements engineering;
(2) the empirical validation of ReFair, which showcase the capabilities of our framework;
(3) a publicly available replication package [1] which includes
  (a) the implementation of ReFair,
  (b) the dataset and scripts used to assess the framework,
  (c) a technical report discussing the additional analyses conducted to assess its capabilities.

Section 2 discusses the related literature; Section 3 describes the construction of the synthetic dataset; our framework is presented in Section 4; its evaluation is shown in Section 5; Section 6 elaborates the limitations of the work; Section 7 outlines our future research.

2 BACKGROUND AND RELATED WORK

Verma and Rubin [60] grouped fairness definitions based on the (1) probability to make a correct prediction among different sensitive groups, *e.g.*, *Statistical Parity*, (2) similarity-based prediction relations among different individuals, *e.g.*, *Fairness through Unawareness*, and (3) causal relation among features and outcomes *e.g.*, *Counterfactual Fairness*. Mehrabi et al. [39] mapped fairness definitions onto (1) group-based discrimination avoidance, *e.g.*, *Equalize Odds*, and (2) individual reasoning prediction monitoring, *e.g.*, *Fairness through awareness*. Based on these definitions, most previous works defined bias mitigation strategies by preprocessing or manipulating training data. For instance, Biswas and Rajan [6] compared 37 different pre-processing pipelines against three datasets, verifying that specific data transformations may impact fairness in training data. Nargesian et al. [45] proposed training data augmentation as a solution to limit unfairness, while Moumoulidou et al. [44] exploited data diversity under fairness constraints, observing how the representativeness of different data groups is key to guarantee fairness in data and avoiding model discrimination. On a similar line, Chakraborty et al. [14] proposed Fair-SMOTE, a data balancing algorithm able to rebalance the internal distribution of training data according to sensitive attribute groups.

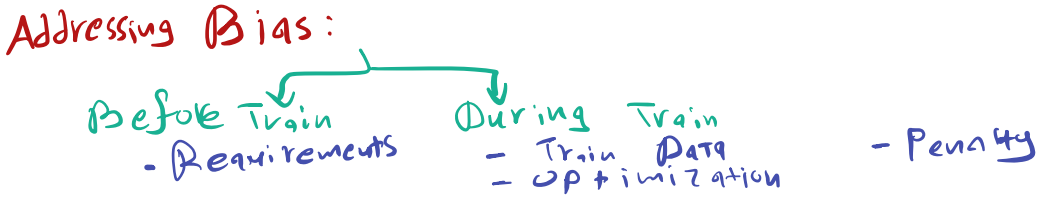
Another line of research refers to optimizing ML model behaviors based on trade-offs. Hort et al. [29] proposed a mutation approach, coined Fairea, to benchmark bias mitigation methods according to specific fairness-accuracy trade-offs. Similarly, Chen et al.[17] proposed MAAT, an ensemble learning approach to jointly improve fairness and model performance metrics. Finally, Galhotra et al. [25] introduced Themis, a test generation approach highlighting unfairness caused in data according to specific fairness metrics and trying to solve them by using specific resampling strategies.

Rather than focusing on data, we propose an automated requirements engineering approach that can classify sensitive features from user stories. As such, our approach may complement existing ones and inform how data preprocessing and manipulation algorithms should optimize training data to reduce fairness concerns. When focusing on requirements engineering, Soremekun et al. [54] pointed out only a few research works aiming to assess fairness during requirements specification and analysis. This is due to the lack of standards and guidelines to manage ethics in requirements engineering [5] and the fact that the analysis of multiple mutually exclusive definitions of fairness in the same requirements specification might be counterintuitive [10]. In addition, ethical concerns are tightly dependent on the specific context a system should operate, both in terms of application domain and ML task that should be performed [23]. As proof of that, previous work [13, 24] highlighted that different notions of fairness are difficult to optimize among customers due to the specificity of the application domains. Our research builds on top of the current knowledge in fairness requirements engineering and represents the *first attempt to elicit ethical concerns from requirements specification automatically*. In particular, our work exploits recent advances in the area of automatic classification of non-functional requirements, where techniques based on natural language processing and word embeddings have been shown to be effective for the classification of non-functional attributes from user stories. For instance, Hey et al. [27] and Casillo et al. [12] have recently showcased successful applications for the classification of function and privacy requirements, respectively. In addition, our work complements research in the field of artificial intelligence, where some preliminary efforts have been made to build tools, e.g., Twitter post analyzers [30], able to provide analytics on the ethical concerns within general-purpose text [11, 15, 28, 35].

3 BUILDING A DATASET OF USER STORIES

One of our work's challenges was the identification of a dataset of ML-enabled system's requirements. It should have reported *domain-specific* requirements of ML-enabled systems, it should have been *diverse* enough to investigate software fairness in various domains, *large* enough to experiment with our approach, and *generic* enough not to be explicitly tailored on fairness analysis. Unfortunately, the current literature does not offer any off-the-shelf solution. Hence, we proceeded with the creation of a *synthethic* dataset, which we built by considering (i) the contemporary requirements engineering processes [31], (ii) the knowledge on the domains where fairness impacts ML solutions [54], and (iii) the trustworthiness of the generation process [63]. Figure 1 overviews the generation process.

[1] The framework analyzes US titles rather than the entire structure of a US; yet, for the sake of readability, we use the term "User Story" throughout the manuscript.



3.1 Requirements Format Selection

Among the available standards [9, 19, 53], we focused on *User Stories* (USs). These represent a widely adopted instrument to describe features of a software system from the perspective of the users that will interact with the system being developed [31]. Requirements engineering processes of ML-enabled systems are typically performed by means of USs [61]. USs enclose three main elements [19]: (1) the *actor*, who is the main user interested in performing a specific task; (2) the *action*, i.e., the activity to perform using the system under development; and (3) the *benefits*, i.e., the advantages the actor (and the application domain) has from acting on the environment through the feature. Among the main advantages of using USs, we identified the possibility to enclose as actions the specific ML tasks that a software system should enable [34]. In addition, the actor and benefits may typically enclose information on the domain where the system should act, hence possibly providing insights into the context-dependent fairness aspects to consider.

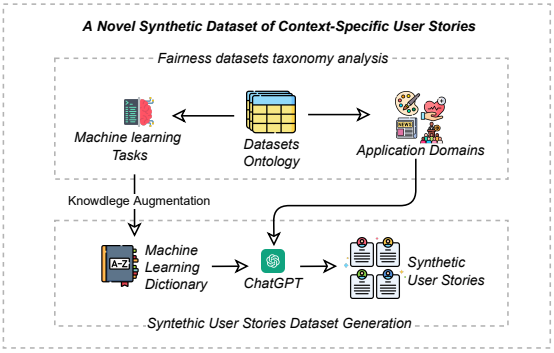


Figure 1: An overview of the USs Dataset Generation Process.

3.2 A Taxonomy of Fairness-Related Application Domains and ML Tasks

The second step toward the definition of our synthetic dataset consisted of mapping the existing knowledge concerned with the application domains and ML tasks for which software fairness concerns arise. This step was key in our case to inform the US generation process, i.e., without such a systematic mapping, we could not know for which domains and ML tasks USs should have been generated. In particular, this was a two-step process that included (1) the manipulation of an existing ontology describing fairness-critical application domains and ML tasks [23]; and (2) the augmentation of such an ontology with the specific ML techniques able to operationalize the various ML tasks.

Mapping the Existing Knowledge. We first exploited the OWL ontology developed by Fabris et al. [23], which maps over 250 fairness-related datasets onto the application domains and ML tasks for which they were used and the corresponding sensitive features representing possible causes of unfairness. To the best of our knowledge, this represents the most updated resource available that describes software fairness in different contexts. The ontology was ideal for our case, as it provides information on (i) the domains we should have considered while generating the synthetic dataset of USs and (ii) the ML tasks that might produce fairness concerns and that we should have further analyzed during the generation process. To make the ontology functional to our purposes, the first author of this article—a SE research assistant with two years of expertise in ML, SE4AI, and ethical AI—manually converted it into a database reporting (1) the application domains and ML tasks classified by Fabris et al. [23]; and (2) the sensitive features impacting fairness within each of the application domains and each of the ML tasks. Overall, the database included 34 application domains and 25 ML tasks, along with the sensitive features that impact them. We made the converted ontology available in our online appendix [1].

Augmenting the Existing Knowledge. While the original ontology provided us with an extensive amount of application domains, the set of ML tasks was quite restrictive. The ontology classified the tasks based on a high-level categorization, *e.g.*, ML classification or regression. However, those tasks could be implemented using a variety of models and algorithms. Building a synthetic dataset solely relying on such a high-level classification might have negatively impacted the conclusion validity of our study.

Indeed, USs may be defined by specifying either the problem or the solution of the ML tasks to be developed [61], for instance they may either indicate the general classification task to be performed (*e.g.*, classification) or the specific technique that will be used to implement a requirement (*e.g.*, a *Naive Bayes* classifier).

To tackle this problem, we performed a data augmentation process aiming at enlarging the set of ML tasks considered by the original ontology. We exploited the AI dictionary proposed by Duran Silva et al. [22]: this is a collection of 599 specific words widely used in different artificial intelligence areas, such as machine learning and natural language processing. This dictionary was built using advanced language models and various large knowledge datasets such as arXiv, Dpedia, Wikipedia, and Scopus. Experts from several universities have validated the final collection of keywords. Among the keywords of the dictionary, 457 of them relate to ML or natural language processing techniques, *i.e.*, they report about learning algorithms or models. The rationale behind the use of the dictionary was that of exploiting this knowledge to link finer-grained techniques to the higher-level tasks reported by the original ontology, hence creating a comprehensive taxonomy that reports, for each application domain, both high-level ML tasks and low-level ML techniques that might lead to fairness-related concerns.

The mapping was manually performed by the first author of the paper, who is referred as "the inspector" in the following. For each of the 457 relevant ML keywords of the dictionary by Duran Silva et al. [22], the inspector (1) verified that the keyword was actually related to a ML or NLP technique, *i.e.*, the keyword matched an algorithm or model—otherwise, the keyword was discarded; and (2) mapped the technique onto one of the 25 higher-level ML tasks. When the inspector was unfamiliar with the specific technique considered, online material, books, or the other paper authors could be consulted to understand how the mapping should have been performed, *i.e.*, which higher-level task would have better suited the technique. The inspector did not find cases where the mapping could not be done: all techniques reported in the dictionary could be successfully mapped, increasing the confidence in the choice of relying on the work by Duran Silva et al. [22] to augment the original ontology by Fabris et al. [23].

The other authors of the paper then verified the consistency of the mapping. The disagreements cases were discussed and solved before proceeding to the next stages. As an outcome of this stage, we could rely on a comprehensive, multi-level taxonomy that reported the application domains where fairness concerns arise along with the high-level ML tasks and low-level techniques that may possibly induce the emergence of fairness issues. In addition, it is worth remarking that—exploiting the knowledge collected through the original ontology—those pieces of information are *directly* mapped onto the specific sensitive features that may cause fairness issues: in other terms, by design, our augmented taxonomy can be used as a basis to create USs that actually provide insights into the application domains and ML techniques that may typically lead ML engineers to deal with sensitive features.

Figure 2 reports a snippet of the taxonomy, showcasing examples of mapping between ML tasks and sensitive features and between application domains and sensitive features. The whole taxonomy is available in our online appendix [1].

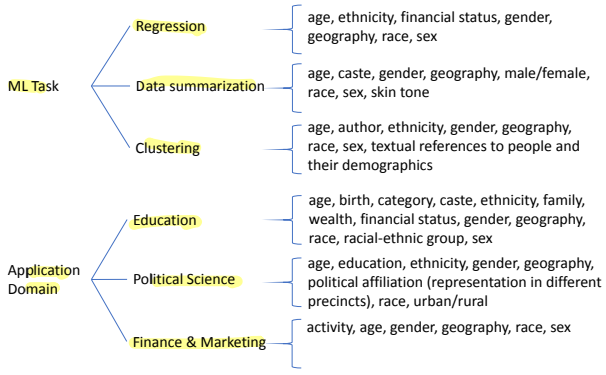


Figure 2: Snippet of the augmented taxonomy built.

3.3 Synthetic Generation of User Stories

The taxonomy represented the input of the final step of the synthetic dataset generation process. To generate USs as realistic as possible, we exploited the capabilities of large language models (LLMs) [8] and, in particular, of ChatGPT.[2] This is one of the most powerful LLMs currently available and is based on the GPT-3.5 architecture.[3] A key challenge was represented by the so-called *prompt engineering* [66], i.e., the definition of the most suitable query that would have allowed ChatGPT to properly process the input and output USs that would have effectively mimicked the actions done by a software engineer. We experimented with multiple prompts, finally coming up with the one whose structure is reported below.

[2]Link to ChatGPT: https://openai.com/blog/chatgpt.
[3]More on GPT 3.5: https://platform.openai.com/docs/models/gpt-3-5.

can be improved using GPT Docs. or providing Example

Prompt employed to generate USs.
Considering the following:
[High-level machine learning task]
OR *[Low-level machine learning technique]*
in the field of *[machine learning]*
OR *[natural language processing]*.
Can you provide me with specific user stories
for the following application domains?
[List of Relevant Application Domains]

The prompt was employed to systematically query ChatGPT and generate a number of domain-specific USs equals to the number of ML tasks/techniques that may induce fairness-related issues, according to the taxonomy built in the previous step. Such a procedure aimed at emphasizing the *context-dependent* nature of ML fairness, putting a strong focus on domain and machine learning task specificity that serve as the foundation of our work. The generation process was *supervised*, i.e., we did not blindly rely on the USs generated by ChatGPT as these might have been unrealistic or erroneous, affecting the overall reliability and representativeness of the dataset. Every time a new US was generated, the first author manually verified its consistency and degree of realism, discarding low-quality USs. As a result of this manual validation, about 3,000 USs targeting 81 of the ML tasks originally considered in the augmented taxonomy were removed, as they were deemed too specific for generating USs with a structure and expressiveness close to the ones produced in real-world development environments.

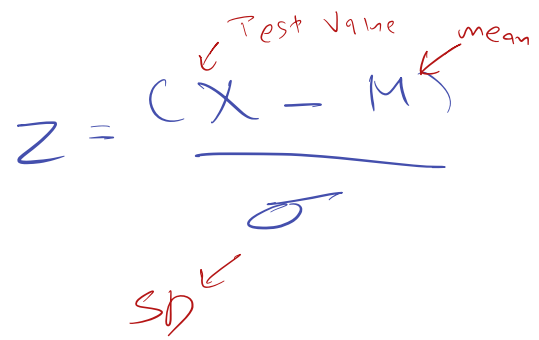
The same Inspector

Overall, the dataset generation process required around 120 person/hours and produced 12,401 synthetic USs related to 34 different application domains. Upon completion of the generation process, the second author double-checked the operations conducted by the first author on a statistically significant sample of 375 synthetic USs (confidence level=95%, margin of error=5% - min 373 instances) to (i) further ensure the reliability of the produced dataset and (ii) to assess its suitability in the subsequent experimental phases. More specifically, the second author conducted a qualitative evaluation on the statistically significant sample of synthetic USs which involved a review against predefined criteria, including clarity, completeness, and relevance to the anticipated experimental conditions. The analysis revealed no inconsistencies. Our online appendix provides access to the dataset, other than to additional reports and examples on the dataset generation process and its validation [1].

→ They Assessed The Data based on 375 Samples. The 375 were statistic Significant - 95% Accuracy

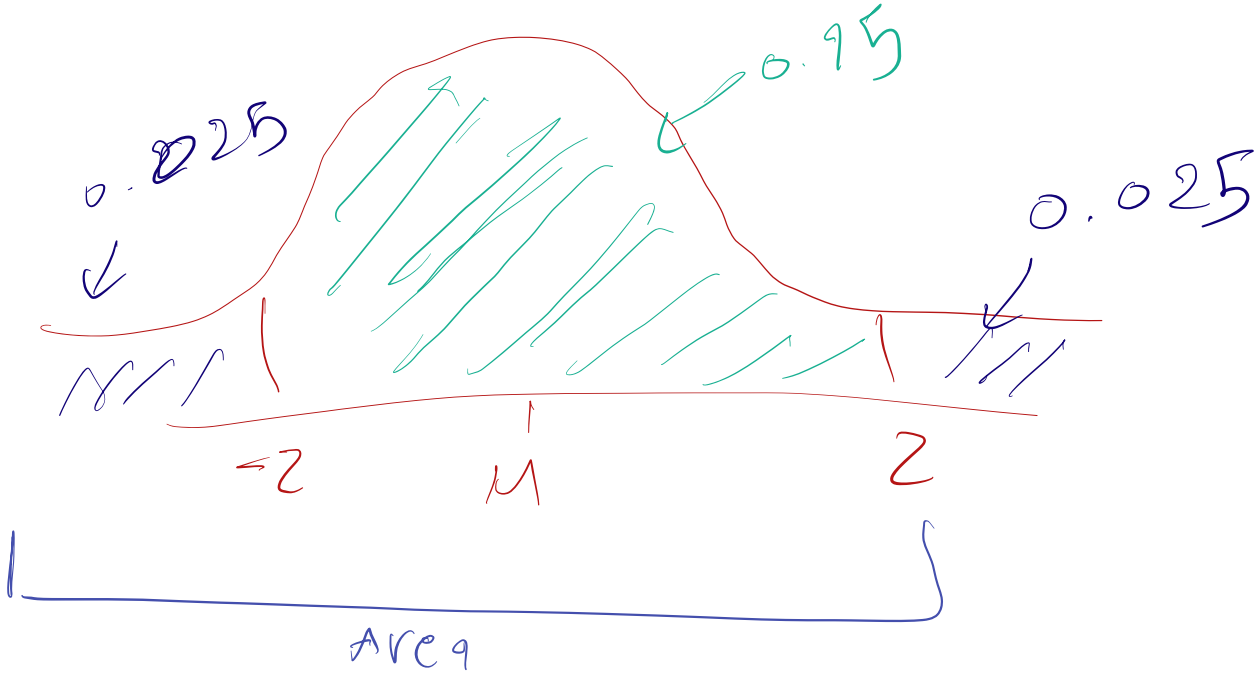
The prompt provided to ChatGPT may generate synthetic USs having different levels of granularity, i.e., they may specify the solution or the problem of the ML tasks to be developed. The motivation behind the choice of the granularity of the generated USs comes from existing literature in the requirements engineering domain [34] that showed that the granularity of USs may largely vary in practice. The original Cohn's user story template [19] indeed described a template to write user stories but it did not provide stringent constraints on how to write them. As a consequence, any development team is free to decide on the level of granularity based on multiple factors like the knowledge of the domain, the information made available by the client, etc. [37]. Our synthetic dataset generation procedure embedded these considerations and attempted to simulate the behavior of the largest population of requirements engineers. As such, we kept the generation as broad as possible, foreseeing the possibility of having a diverse set of USs.

z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
0.0	.5000	.5040	.5080	.5120	.5160	.5199	.5239	.5279	.5319	.5359
0.1	.5398	.5438	.5478	.5517	.5557	.5596	.5636	.5675	.5714	.5753
0.2	.5793	.5832	.5871	.5910	.5948	.5987	.6026	.6064	.6103	.6141
0.3	.6179	.6217	.6255	.6293	.6331	.6368	.6406	.6443	.6480	.6517
0.4	.6554	.6591	.6628	.6664	.6700	.6736	.6772	.6808	.6844	.6879
0.5	.6915	.6950	.6985	.7019	.7054	.7088	.7123	.7157	.7190	.7224
0.6	.7257	.7291	.7324	.7357	.7389	.7422	.7454	.7486	.7517	.7549
0.7	.7580	.7611	.7642	.7673	.7704	.7734	.7764	.7794	.7823	.7852
0.8	.7881	.7910	.7939	.7967	.7995	.8023	.8051	.8078	.8106	.8133
0.9	.8159	.8186	.8212	.8238	.8264	.8289	.8315	.8340	.8365	.8389
1.0	.8413	.8438	.8461	.8485	.8508	.8531	.8554	.8577	.8599	.8621
1.1	.8643	.8665	.8686	.8708	.8729	.8749	.8770	.8790	.8810	.8830
1.2	.8849	.8869	.8888	.8907	.8925	.8944	.8962	.8980	.8997	.9015
1.3	.9032	.9049	.9066	.9082	.9099	.9115	.9131	.9147	.9162	.9177
1.4	.9192	.9207	.9222	.9236	.9251	.9265	.9279	.9292	.9306	.9319
1.5	.9332	.9345	.9357	.9370	.9382	.9394	.9406	.9418	.9429	.9441
1.6	.9452	.9463	.9474	.9484	.9495	.9505	.9515	.9525	.9535	.9545
1.7	.9554	.9564	.9573	.9582	.9591	.9599	.9608	.9616	.9625	.9633
1.8	.9641	.9649	.9656	.9664	.9671	.9678	.9686	.9693	.9699	.9706
1.9	.9713	.9719	.9726	.9732	.9738	.9744	.9750	.9756	.9761	.9767
2.0	.9772	.9778	.9783	.9788	.9793	.9798	.9803	.9808	.9812	.9817
2.1	.9821	.9826	.9830	.9834	.9838	.9842	.9846	.9850	.9854	.9857
2.2	.9861	.9864	.9868	.9871	.9875	.9878	.9881	.9884	.9887	.9890
2.3	.9893	.9896	.9898	.9901	.9904	.9906	.9909	.9911	.9913	.9916
2.4	.9918	.9920	.9922	.9925	.9927	.9929	.9931	.9932	.9934	.9936
2.5	.9938	.9940	.9941	.9943	.9945	.9946	.9948	.9949	.9951	.9952
2.6	.9953	.9955	.9956	.9957	.9959	.9960	.9961	.9962	.9963	.9964
2.7	.9965	.9966	.9967	.9968	.9969	.9970	.9971	.9972	.9973	.9974
2.8	.9974	.9975	.9976	.9977	.9977	.9978	.9979	.9979	.9980	.9981
2.9	.9981	.9982	.9982	.9983	.9984	.9984	.9985	.9985	.9986	.9986
3.0	.9987	.9987	.9987	.9988	.9988	.9989	.9989	.9989	.9990	.9990
3.1	.9990	.9991	.9991	.9991	.9992	.9992	.9992	.9992	.9993	.9993
3.2	.9993	.9993	.9994	.9994	.9994	.9994	.9994	.9995	.9995	.9995
3.3	.9995	.9995	.9995	.9996	.9996	.9996	.9996	.9996	.9996	.9997
3.4	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9998



95% → 1.96 ≈ (1.9 + 0.06)

Area = (1 + 0.95) / 2 = 0.975



1. **Confidence Level**: They chose a **95% confidence level**.
2. **Margin of Error**: They set the **margin of error** at **5%**.
3. **Population Size**: The population size in their case was **12,401** User Stories.
4. **Estimated Proportion (p)**: If no prior knowledge of the proportion is available, the most conservative estimate is used, which is **50% (p = 0.5)**. This assumes the largest variability in the population.

$$n = \frac{Z^2 \cdot p \cdot (1-p)}{e^2} \times \frac{N}{N + Z^2 \cdot p \cdot (1-p) - 1}$$

Step 1: Calculate the initial sample size without considering the population size:

$$n_0 = \frac{1.96^2 \cdot 0.5 \cdot (1-0.5)}{0.05^2}$$

$$n_0 = \frac{3.8416 \cdot 0.25}{0.0025} = \frac{0.9604}{0.0025} = 384.16$$

This would be the sample size if the population were infinite. However, since the population size is finite, we need to adjust this using the **finite population correction**.

Step 2: Apply finite population correction:

$$n = \frac{384.16 \cdot 12,401}{12,401 + (384.16 - 1)}$$

$$n = \frac{4,762,841.6}{12,784.16} = 372.55$$

Step 3: Round up the result:

Rounding **372.55** to the nearest whole number gives **373**.

Second Author Did Qualitative Assess

1. Clarity:

- **Definition**: Clarity refers to how easily understandable the User Story is. A clear User Story should have well-defined language, with no ambiguity in the wording, making it easy for any stakeholder (developer, analyst, or client) to grasp the intention behind the requirement.
- **Evaluation**: In the assessment, the authors manually reviewed whether the generated User Stories were free from vague or confusing language. The goal was to ensure that the generated USs would be interpretable and actionable by software teams.

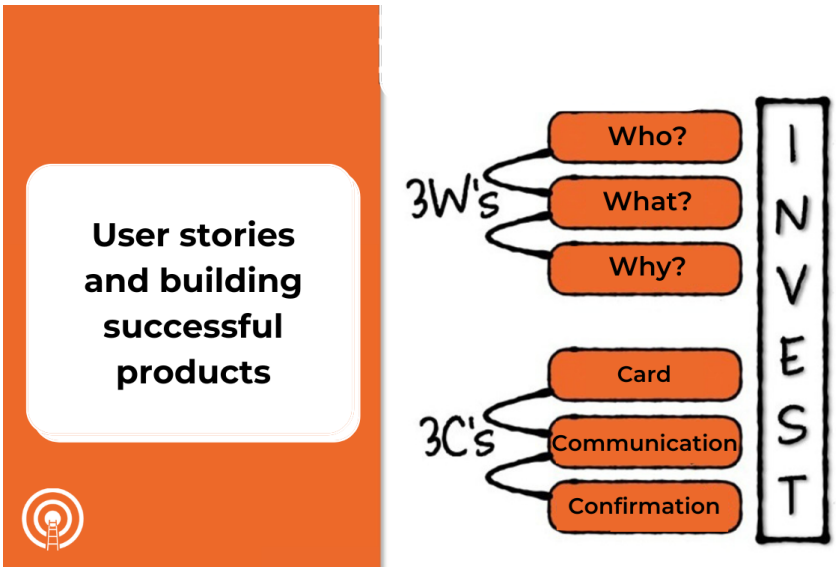
2. Completeness:

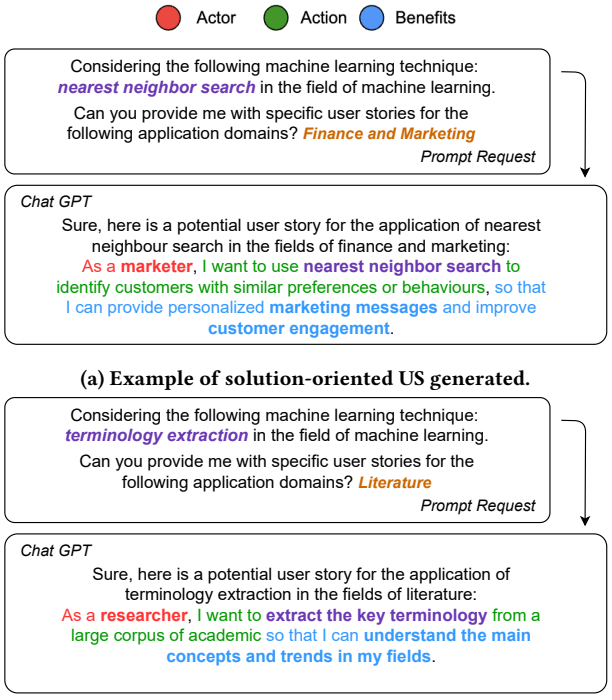
- **Definition**: Completeness in this context refers to whether the User Story contains all the necessary information to fully capture the requirement. This includes the actor (user), action (what they want to achieve), and the benefit (why it matters).
- **Evaluation**: The authors checked if each User Story had all required components and could stand on its own without needing additional clarification. Incomplete User Stories were flagged for further refinement.

3. Relevance:

- **Definition**: Relevance measures how closely aligned the User Story is to the domain or task it's intended for. A relevant US should address a genuine need or functionality within the system being developed.
- **Evaluation**: The authors assessed whether the generated User Stories were contextually appropriate for the machine learning tasks and application domains they were intended to describe. They reviewed whether the USs were meaningful and could feasibly apply to real-world development scenarios.

User Story Temp





(a) Example of solution-oriented US generated.

(b) Example of problem-oriented US generated.

Figure 3: Examples of synthetic USs Generation.

To showcase the levels of granularity of the USs in our dataset, let consider the two examples depicted in Figures 3a and 3b. In both cases, the US pattern defined by Cohn [19] is fully met. The former describes a solution-oriented US concerned with the application domain named *"Finance and Marketing"*, where a rather specific ML task is specified, i.e., *"Nearest Neighbor Search"*. The latter, instead, refrains from specifying a ML task and revolves around the problem of *"terminology extraction"*, which can be later implemented using different methods. In this instance, the "what" conveyed is the user's desire to *"extract the key terminology from a large corpus of academic"*, maintaining a high-level abstraction of the ML task.

3.4 Synthetic Dataset Validation

The *supervised* dataset generation process ensured that the synthetic USs closely resembled the ones that a software engineer would produce, as the internal assessments suggested. This validation instilled a high level of confidence in the dataset's reliability. Nevertheless, to further safeguard against potential subjectivity in the internal inspection, we undertook an additional and comprehensive validation of the synthetic dataset. In particular, we let practitioners assess the USs's quality in our dataset. First, we extracted a statistically significant different sample than the one used for the internal inspection of 375 synthetic USs (confidence level=95%, margin of error=5% - min 373 instances). Second, we randomly split the 375 USs into 75 groups, each containing five USs. Third, we designed 75 online surveys - one for each USs group - to (1) present each of the five USs and inquire the respondents about *comprehensibility*, i.e., the degree to which the US is understandable, *realism*, i.e., the degree to which the US is written as a real user story, and *actionability*, i.e., the degree to which the US can be used to drive the development of a ML-enabled project (the respondents judged these properties for each proposed US through a 5-point Likert scale [4]); and (2) ask respondents to provide feedback on how to improve the US deemed unrealistic. To avoid respondents being biased, in either positive or negative fashion, we did not reveal the synthetic nature of the USs under evaluation. We administered the survey through Prolific[4], following the guidelines by Reid et al. [52] to prevent invalid responses, and applying quality assessment to discard unreliable responses. We cross-validated two reports for each USs group, thus involving 150 practitioners with experience in software design and ML. The results of the surveys corroborated those of the internal validation: the involved practitioners (i) gave the USs very high scores for all indicators, achieving at least ≈4 on the Likert scale; (ii) provided recommendations on how 46 unique USs (12%) could be improved. These recommendations were minor and addressed ambiguous terminology affecting the readability of the USs, not altering their core structure or intent. As such, we did not require to propagate the changes to the other USs in the sample nor to the entire dataset. The external validation is more detailed in the technical report released in our online appendix [1].

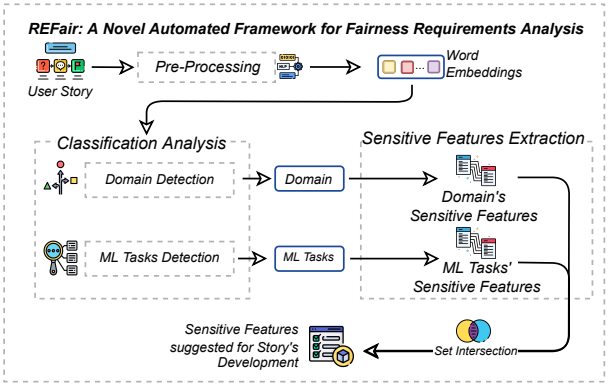


Figure 4: ReFair: An overview of the proposed approach.

4 THE REFAIR FRAMEWORK

The key idea behind ReFair is to analyze USs of ML-enabled systems with the aim of (1) *classifying* the application domain of the system being developed; (2) *classifying* the ML task(s) that will be employed to implement the US; and (3) *mapping* those pieces of information onto the specific sensitive features to account when working under the classified application domain and ML task.

As such, ReFair supports the requirements engineer by providing recommendations that may inform the follow-up development activities of the potential fairness concerns to take into account. We are aware that other external factors, e.g., laws and regulations, may influence the identification of sensitive features. On the one hand, ReFair is designed to work with software engineering artifacts, i.e., User Stories, rather than with elements that may be hardly

[4]The Prolific administration platform: https://www.prolific.co/

extracted because of their tight relation with the specific customs or regulations of the society where a system is being developed. On the other hand, our framework does not aim at replacing the requirements engineer but provide insights that may be further elaborated. From a technical standpoint, ReFair exploits the *base ontology* built in Section 3 to learn how to detect sensitive features based on application domains and ML tasks, and is fed with the set of synthetic USs coming from the dataset generation process.

We made ReFair working with the base ontology rather than with the augmented taxonomy as we aimed at designing a *generalized* framework that could not limit itself to the analysis of the currently existing ML techniques, but that would rather allow the classification of a set of general high-level tasks that can be adapted to multiple application domains, hence recommending sensitive features independently from the specific ML techniques that engineers will use. Such a generality has an additional implication: the best ML technique to use in a given context may result from experimental analyses performed after the requirements engineering phase. Relying on a higher-level task classification can better inform how such investigations should be performed. Hence, we argue that such a design choice better fits the ML engineers' needs. On the contrary, feeding and experimenting the framework with the USs coming from the augmented taxonomy was key to understand the robustness of the framework. Those USs were based on different levels of granularity, hence simulating multiple conditions arising in reality which might challenge the capabilities of our framework.

Figure 4 overviews the main steps of the proposed approach. A US represents the input of ReFair. This is preprocessed with the aim of producing a word embedding representation of the elements of the US. Those embeddings will feed two different ML models:

the first will be responsible for classifying the application domain of the US, while the second will classify the most likely ML task(s) that may be employed to implement the US. The outcomes of these two models will then be used to map the classified application domain and ML tasks to the corresponding sensitive features. These will be finally presented to the user. The next sections provide more details on each of the steps of our approach.

4.1 User Story Preprocessing

The first step of ReFair allows to produce a N-dimensional space representation of the input US [33]: in turn, such a representation enables the extraction of features out of the text that natural language models can use for classification purposes. In other terms, this step allows ReFair to transform the text contained within a US into a real-valued vector that can be used in the following steps. Our framework supports multiple word embedding techniques such as TF-IDF [59], BERT [20], Word2Vec [42], FastText [7], and GloVe [47]. The assessment reported in Section 5 aimed at experimenting with those techniques and identifying the best one.

4.2 Classification Analysis

The word embeddings are then taken as input by the classification analysis modules of ReFair. This comprises two main components:

Application Domain Classification. This component is responsible for classifying the most likely application domain of the US among the 34 domains available in the ontology.

Multiclass

We have modeled the domain detection problem as a multi-class classification task [32], where (1) the features are represented by the real-valued vector of the word embeddings and (2) the classification labels correspond to the application domains of the augmented taxonomy. Our framework supports 25 ML algorithms that make different assumptions on the underlying data as well as have different advantages and drawbacks in terms of execution speed and overfitting. For instance, ReFair provides users with the possibility to run probabilistic algorithms, *e.g.*, Gaussian Naive-Bayes, entropy-based classifiers, *e.g.*, Random Forest, semi-supervised approach, *e.g.*, Label Propagation, and others. Our online appendix [1] reports the complete list of ML techniques supported.

multi-label Class

Machine Learning Tasks Classification. This is responsible for classifying the ML tasks likely to be employed when implementing the US. We have modeled the problem as a multi-label classification task [57], as a US may be operationalized using multiple ML techniques. For instance, the term *"artificial neural network"* may refer to tasks of regression, classification, clustering, and more. As such, we designed the framework to be *conservative* enough and identify all the potential ML tasks that may lead to unfairness. From a practical perspective, this choice may allow the users to receive a larger set of sensitive features, hence favoring recall over precision: this was done on purpose, as we preferred to provide users with actionable feedback that might be later interpreted rather than with a more restrictive set of sensitive features that may have overlooked some relevant pieces of information. The implications of these design choices are later analyzed as part of the empirical study. In this case, the features are represented by the real-valued vector of the word embeddings, while the classification labels consist of the high-level ML tasks of the augmented taxonomy.

As for the actual classification, ReFair supports established multi-label techniques such as Binary Relevance (BR) [36], Classifier Chain (CC) [50], and Label Powerset (LP) [55]. We rigorously tested these techniques with popular ML algorithms such as Logistic Regression (LR), Random Forest (RF), Gaussian Naive Bayes (GNB), Linear Support Vector Classification (LSVC), K-Nearest Neighbors (KNN), and Decision Tree (DT). We did not consider the analysis of more advanced ML solutions, *e.g.*, Deep Learning (DL) neural networks, as (1) shallow ML classifiers are more interpretable and explainable, possibly providing ReFair with an additional, relevant feature that would increase its practical usability; (2) DL models may not be required if the performance of shallow ML classifiers are already high.

4.3 Sensitive Features Recommendation

The application domain and ML tasks classified in the previous step are finally used to recommend sensitive features. ReFair exploits the base ontology [23] to identify the sensitive features connected to both the application domain and ML tasks concerned with the the classified domain. The intersection of those sensitive features represents the final outcome of the framework, *i.e.*, the outcome comprises the set of sensitive features relevant when jointly considering the application domain and the learning tasks.

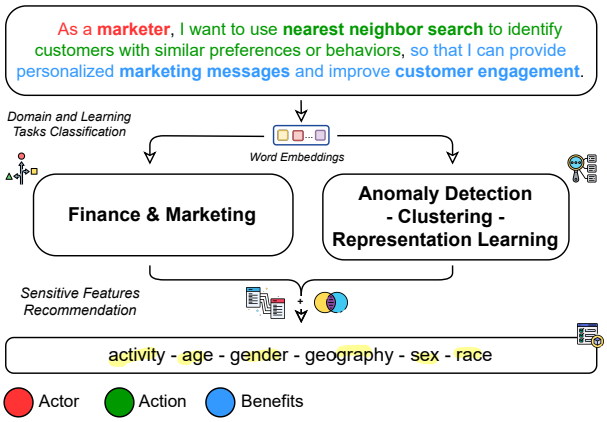


Figure 5: ReFair: Running Example.

4.4 Prototypical Implementation

We released the source code of the prototypical implementation of ReFair in our online appendix [1]. For the sake of understandability, Figure 5 reports a running example of our framework, which shows how it could recommend sensitive features for the US generated in Figure 3a. The example shows a successful classification from the empirical study discussed later in the paper, in partucular the results obtained when configuring the framework with (1) the XGBoost classifier for the domain classification and (2) Linear Support Vector Machine for the ML tasks classification. The approach correctly classifies *Finance & Marketing* as domain and identifies *Anomaly Detection*, *Clustering*, and *Representation Learning* as possible ML tasks through which the US may be implemented. The classification of these tasks is consistent, as these are widely applied in the context of financial applications [23]. Finally, for each ML task, ReFair maps relevant sensitive features, recommending *working activity*, *age*, *gender*, *geography*, *sex*, and *race* as the set of sensitive features to consider in the subsequent implementation steps of the US. The recommendations can be considered both complete and consistent, as these are the well-known attributes that should be considered while developing financial ML applications.

5 EMPIRICAL EVALUATION

The *goal* of our study was to assess ReFair, with the *purpose* to measure the extent to which it may support fairness requirements engineering. The *perspective* is of researchers and practitioners. The former are interested in assessing the viability of automatically detecting sensitive features from User Stories. The latter are interested in assessing whether it may be actually employed in practice.

Specifically, we first focused on the capabilities of ReFair in classifying application domains and machine learning tasks. These are indeed the two aspects that determine the final accuracy of the recommendations provided, i.e., if ReFair correctly classifies application domains and machine learning tasks, the outcome will be correct *by definition*, as the sensitive features recommended would directly map onto the base ontology reporting the ground truth on the fairness attributes to consider in that domain and for those ML tasks. We formulated two research questions (**RQs**):

RQ$_1$. *To what extent can ReFair classify ML-specific application domains from User Stories?*

RQ$_2$. *To what extent can ReFair classify ML-specific tasks from User Stories?*

After assessing the classification components of ReFair, we moved toward the evaluation of the sensitive feature recommender. When either the application domain or the ML tasks are misclassified, ReFair may recommend sensitive features that are inconsistent for a given US. Hence, we formulated a third research question:

RQ$_3$. *To what extent can ReFair recommend sensitive features from User Stories?*

In terms of reporting we followed the guidelines by Wohlin et al. [64], other than the *ACM/SIGSOFT Empirical Standards*.[5] The *context* of the study was represented by the synthetic dataset described in Section 3. For each application domain, the dataset contained 365 USs which were used to experiment with our framework.

5.1 Addressing RQ$_1$: The ReFair Application Domain Classification Performance

We addressed **RQ$_1$** by experimenting with the word embeddings and domain classifiers supported by our framework.

Experimental Setting. We designed an empirical study to identify the best classifier among the 25 ML multiclass classification algorithms available within ReFair. Such an experiment was performed through the use of *Lazy Predict*,[6] a Python library that facilitates the comparison of multiple models and does not require manual parameter tuning. All the algorithms were evaluated using the five word-embedding techniques listed in Section 4.1. Specifically, the US dataset was first represented by using the i^{th} embedding technique considered. Afterwards, we applied a ten-fold cross validation [26] to split the dataset in ten folds and let *Lazy Predict* assess the performance of each of the 25 algorithms on each fold. To effectively manage the computational demands associated with testing multiple combinations of word embeddings and ML techniques, we represented USs using a fixed vector of 100 tokens, independently from the size of the requirements or the embedding method used. This approach reduced the overall computational time required for testing while keeping the results informative and meaningful. The results were then analyzed using *pandas* [62], which allowed us to group them and obtain the average performance for all models over all folds. As the experiment focused on domain detection, which we modeled as a multiclass classification task, we used F1-Score and accuracy as evaluation criteria [49]. F1-Score is a commonly used metric for evaluating the performance of multiclass classification models [59]. It is computed as the harmonic mean of precision and recall, which provides a balance between these two measures, where precision measures the proportion of correct positive predictions among all positive predictions and recall measures the proportion of correct positive predictions among all

[5] Available at: https://github.com/acmsigsoft/EmpiricalStandards. Given the nature of the study and the standards currently available, we employed the *"General Standard"*, the *"Data Science"*, and the *"Engineering Research"* guidelines.

[6] The *Lazy Predict* library: https://github.com/shankarpandala/lazypredict

Comparison of selected embedding techniques for the Domain Detection classifier.															
TF-IDF			BERT			Word2Vec			FastText			Glove			
Model	F1-Score	Accuracy	Model	F1-Score	Accuracy	Model	F1-Score	Accuracy	Model	F1-Score	Accuracy	Model	F1-Score	Accuracy	
ET	0.80	0.80	**XGBC**	**0.98**	**0.98**	CCCV	0.91	0.91	SVC	0.94	0.94	CCCV	0.91	0.91	
SVC	0.80	0.80	BC	0.98	0.98	LR	0.91	0.91	CCCV	0.94	0.94	LR	0.91	0.91	
CCCV	0.80	0.80	DT	0.98	0.98	LSVC	0.90	0.90	LR	0.94	0.94	LDA	0.90	0.90	

Table 1: Domain classifier selection - Experimental Results
ET = Extra Trees, SVC = Support Vector Classification, CCCV = Calibrated Classifier CV, XGBC = XGB Classifier, BC = Bagging Classifier, DT = Decision Tree, LR = Logistic Regression, LSVC = Linear SVC, LDA = Linear Discriminant Analysis

actual positive instances. Accuracy measures the overall proportion of correct predictions, regardless of the class. It is computed as the ratio of correctly classified instances to the total number. At the end of this analysis, the best model was subject to a hyperparameters fine-tuning step to obtain the model that best fits the supplied data. Once we had identified the best performing classifier, we further refined it by running the RandomizedSearchCV algorithm: this is an automated configuration instrument provided by Scikit-learn, which involves testing random combinations of values from a range, as opposed to predefined values in classical Grid Search. This allowed us to (1) understand whether the results obtained through *LazyPredict* were reliable and (2) carry out another round of cross-validation, consolidating the results obtained and preventing them from being facilitated by the split used for the previous analysis.

Experimental Results. We evaluated 125 combinations of word embeddings and classification algorithms. While the detailed results for all models are available in our online appendix [1], Table 1 presents the top-3 models for each embedding technique in terms of accuracy and F1-Score. Notably, the XGB Classifier [16]—an implementation of gradient-enhanced decision trees designed for speed and performance—using BERT as word embedding emerged as the best-performing model, reaching 98% of F1-Score and accuracy. Two other models, i.e., Bagging Classifier and Decision Tree, also achieved the same performance, highlighting the effectiveness of BERT as word embedding for the textual representation of our task. Nonetheless, the combination of BERT and XGB Classifier provided the best compromise between performance and efficiency: hence, we deemed this model as the best one resulting from the application domain model selection step.

In the second step, we configured XGB Classifier by considering *max-depth*, *learning-rate*, *subsample*, and *n_estimators* as hyperparameters. The best configuration was the following: {*learning-rate: 0.087, max-depth: 3, n-estimators: 80, subsample: 0.924*}. The performance of the XGB Classifier were similar to those obtained without hyper-parameter configuration. On the one hand, this confirmed that we could classify the application domain within USs with high accuracy. On the other hand, our findings suggest that an additional, possibly costly fine-tuning refinement would not be strictly required to obtain high performance.

Summary of the Results. We empirically evaluated 125 ML algorithms and word embedding combinations to address **RQ$_1$**. The combination of BERT and XGB Classifier exhibited Accuracy and F1-Score close to 98% in the domain detection task.

5.2 Addressing RQ$_2$: The ReFair Machine Learning Tasks Classification Performance

Similarly to what done previously, we addressed **RQ$_2$** by experimenting with the word embeddings and machine learning tasks classification mechanisms supported by our framework.

Experimental Setting. As described in Section 4.2, the classification of the ML task associated to a fairness-critical application domain may be challenging, as multiple tasks can be used during the development of ML-enabled systems. As such, we modeled a multi-label classification problem, wherein USs may relate to multiple ML tasks. We leveraged Scikit-multilearn [55], a BSD-licensed library for multi-label classification that is based on the well-known Scikit-Learn ecosystem. We exploited *MultiLabelBinarizer* [55] to transform the output to be predicted. Unlike a single value indicating the class to which it pertains, in this case, the output is represented by an array of values indicating which classes the US belongs to. Similarly to **RQ$_1$**, we experimented with each combination of multi-label technique, classification model, and word embedding method considered by ReFair, by performing a ten-fold cross validation [26]. The results were examined using *pandas*, which allowed us to cluster the results and derive the mean performance for all possible combinations across all folds. As evaluation metrics, we considered F1-Score and Hamming Loss values for every combination. Both metrics have been widely adopted in the context of multi-label classification problems, especially when the number of labels is large [57]. In particular, the F1-Score was computed for each label separately, taking the average as a final performance indicator. As for the Hamming Loss, this measures the fraction of misclassified labels. It was computed as the average number of labels that were incorrectly predicted for each instance.

Experimental Results. Overall, we empirically evaluated 90 combinations of word embeddings and ML task classification algorithms. While the complete results are in online appendix [1], Table 2 reports the top-3 performing models for each embedding technique. The combination of Glove, Label Powerset, and Linear Support Vector Classification obtained the highest F1-Score and the lowest Hamming score. Nevertheless, unlike **RQ$_1$**, where the word-embedding technique impacted the classification results, in this case, we discovered that the multilabel classification technique made the difference, achieving excellent results regardless of the word-embedding technique. More particularly, Label Powerset, in combination with different classification algorithms, achieves results above 70% in terms of F1-Score with a Hamming Loss of at most 15% in the worst case, even with different techniques

Comparison of selected embedding techniques for ML Task Detection classifier.

TF-IDF			BERT			Word2Vec			FastText			Glove		
Technique + Model	F1-Score	Hamming Loss	Technique + Model	F1-Score	Hamming Loss	Technique + Model	F1-Score	Hamming Loss	Technique + Model	F1-Score	Hamming Loss	Technique + Model	F1-Score	Hamming Loss
LP + RF	0.88	0.06	LP + DT	0.86	0.07	LP + LSVC	0.86	0.07	LP + LSVC	0.81	0.09	**LP + LSVC**	**0.90**	**0.05**
LP + DT	0.82	0.09	LP + RF	0.84	0.08	LP + GNB	0.76	0.12	LP + GNB	0.81	0.10	LP + GNB	0.72	0.15
BR + RF	0.80	0.09	BR + DT	0.78	0.11	BR + KNN	0.68	0.15	BR + KNN	0.74	0.12	BR + KNN	0.67	0.15

Table 2: Machine Learning Task classifier selection - Experimental Results

for text representation. The best results were finally obtained by combining Label Powerset, Glove, and LSV: the F1-Score was slightly above 90%, while the Hamming Loss scored 5%.

Summary of the Results. We assessed 90 combinations of multilabel techniques, ML algorithms, and word-embedding methods. The combination of Label Powerset, Glove, and Linear Support Vector Classification reached an F1-Score higher than 90% and Hamming Loss of 5% on the ML tasks classification.

5.3 Addressing RQ$_3$: The ReFair Sensitive Feature Recommendation Capabilities

We addressed **RQ$_3$** by comparing the output of ReFair against an oracle reporting the actual set of sensitive features to be provided.

Experimental Setting. To address **RQ$_3$**, we first built an oracle reporting the correct set of sensitive features for each US of our dataset. This was done by exploiting the base ontology [23]. As explained in Section 3.2, it provided sensitive features for each application domain and ML task: as the considered USs were concerned with an actual application domain and ML task, we could use the base ontology to label each US with the corresponding set of sensitive features. Afterward, we compared such an oracle against the recommendations of ReFair. This evaluation allowed us to measure how much the misclassifications assessed in **RQ$_1$** and **RQ$_2$** altogether influenced the capabilities of our framework, hence providing a final assessment of the support that ReFair may provide during requirements engineering. We computed the MoJo distance [58] as the evaluation metric, which is a widely accepted method to measure the distance between two partitions of the same set. This metric ranges from 0 (indicating a completely identical set) to 1 (representing two completely different sets). It is based on the number of "moves" required to make the two compared sets identical. Here, a "move" refers to either shifting a single data point from one cluster to another or swapping the cluster assignments of two data points. To evaluate the distance between the set of features recommended by ReFair and the oracle for each US, we employed a specific MoJo variant that relies on the Jaccard's and the Mean sets overlap indexes [40]. In addition, we analyzed the results from a more qualitative standpoint by computing the *amount* of sensitive features erroneously recommended by ReFair: to this aim, we leveraged the token-level Levenshtein distance [41], considering each token as a sensitive feature.

Experimental Results. The average MoJo distance computed by comparing all the sets of sensitive features against the oracle reached 0.04, meaning that the output of ReFair was just 4% far from the ideal one. This result indicates that the misclassifications in terms of application domain and ML task have a marginal impact on the overall capabilities of ReFair, hence making it a potentially suitable instrument to support requirements engineering activities. Going deeper, the Levenshtein distance [41] analysis revealed that ReFair acted as a *perfect recommender* on 11,969 USs (97%). Of the remaining 432 (3%), in 41 cases (less than 1%) the feature sets of ReFair and the oracle differed by only 1 feature, 70 (0.6%) differed by 2 features, and 321 (less than 3%) differed by more than 2 features. By analyzing the latter cases more closely, we observed that (1) these would have still led ReFair to provide some support in a real case scenario, as the set of recommendations were partially correct, i.e., wrong recommendations were up to 52.5% of the recommendations provided by ReFair; and (2) they were mostly due to specific application domains, e.g., *Health*, and ML tasks, e.g., *Classification* which were those more often misclassified. This suggests that further improvements of ReFair could revolve around the addition of targeted data samples or the application of data augmentation methods able to provide the framework with a more consistent knowledge. We report additional analyses into the capabilities of ReFair per application domain and ML task as part of the technical report that accompanies this submission [1].

Summary of the Results. The MoJo distance showed a near-perfect match (0.04) between the set of sensitive features recommended by ReFair and by the oracle. Our framework acts as a perfect recommender in 90% of the cases, providing mostly wrong recommendations in just 5% of the cases.

6 THREATS TO VALIDITY

Our study suggested that ReFair may represent a valuable instrument, yet various aspects might have biased the conclusions drawn.

Threats to External Validity. A first threat is represented by the user story format employed in the study. While this possibly limits the applicability of ReFair, previous studies showed that these are widely used in the requirements engineering of machine learning-enabled systems [61]. As part of our future research agenda, we plan to generalize the framework to other formats.

Our framework was built on top of the current knowledge on the fairness-critical application domains and machine learning tasks [21, 23]. As such, its application is limited to such a knowledge. However, ReFair was designed to be easily extended; the source code is publicly available so that researchers can build on top of it.

The generalizability of the results might have also been threatened by the granularity of the USs automatically generated and, more in general, by the use of synthetic USs to experiment with ReFair. As explained in Section 3.3, we accounted for the liberty developers have in real-case scenarios [34], hence enabling the generation of both solution- and problem-oriented USs that might have properly simulated a realistic use case for ReFair. Nonetheless, further investigations into the generalizability of these USs should be pursued. To partially mitigate such a threat to validity and preliminarily assess how ReFair may work in a realistic environment, we conducted a *qualitative* experimentation involving real-world machine learning engineers. The goal was to understand the capabilities of our framework when run against manually-written requirements specifications. We involved 20 machine learning engineers from our contact network and asked them to develop requirements specifications that could be later employed to assess the soundness of our framework. The participants had between two and five years of professional experience, had knowledge on both software engineering and artificial intelligence. We involved them through e-mails, by asking for a volunteer participation.

Upon confirmation of their participation, we sent them a link to an online questionnaire which comprised three sections. The first presented the informative consent: we clarified that the answers would have kept anonymous to preserve privacy and that their responses would have been used for a research submission. The second aimed at collecting demographic data. The third proposed a problem statement concerned with a specific machine learning domain among those investigated in this study [23]. For instance, one of the problem statements revolved around the definition of a machine learning-based software system able to classify cancer types based on genomic data. While each participant was assigned to a problem statement pertaining to a different domain, we could not assess all 34 domains considered in the study because of the lack of participants. The problem statements were crafted by the first author of the paper, who elaborated them with the help of online resources and existing projects, in an effort of producing realistic cases to propose to the participants. The full set of problem statements are available in our online appendix [1].

Participants were asked to carefully read the problem statement they were assigned to and produce up to ten requirements involving the machine learning solutions that may be employed in the context. We gave them ten days to deliver, collecting a set of 119 requirements that could be classified by ReFair, which was configured according to the empirical results presented in Section 5.

We could not compute precise performance metrics because of the unavailability of a ground truth for these requirements, yet we manually went through a qualitative analysis and assessed whether these might have been considered similar or meaningful. We noticed that the major differences between the manually- and automatically-generated USs were due to two aspects. On the one hand, the participants' knowledge on the problem statements presented: when participants were not familiar enough with the domain, they indeed developed more generic USs that were hard for ReFair to correctly handle. On the other hand, the machine learning engineers' experience with requirements engineering: we noticed that more experienced practitioners were able to develop higher-quality USs that better characterized the elements to be implemented, hence allowing ReFair to properly classify potential sensitive features. In conclusion, the qualitative investigation confirmed the value of ReFair, yet discovered the potential boundaries that may affect its capabilities, namely *domain familiarity* and *experience* of the practitioners that engage with the framework. As part of our future research agenda, we aim at performing larger-scale studies that might better assess the differences between manually- and automatically-written USs, other than the implications for the classification performance of ReFair.

Threats to Construct Validity. Our study was designed to take the context-dependent nature of software fairness into account, i.e., some sensitive features might be considered as such depending on the context: as such, we trained and tested ReFair on USs coming from a large variety of fairness-critical domains [23]. Additional, hardly operationalizable external factors, e.g., laws and regulations, may influence the identification of sensitive features: as any recommendation system, ReFair must be considered as an assistant rather than a tool to replace the requirements engineer by providing insights that might be relevant for the development of a certain US.

As for the synthetic dataset, we built it on top of the existing knowledge on requirements engineering [54, 61] and software fairness [23], favoring the generation of USs having different levels of granularity [34] and employing a reliable Large Language Model like ChatGPT, which we inquired only after experimenting with multiple prompts. The internal validation of the dataset quality, other than the external validation conducted with practitioners, increased our confidence on its validity and suitability for our purposes. Replications of our study on real-world datasets would still be desirable and part of our future research agenda.

We built a prototypical implementation of ReFair by experimenting with a wide set of shallow machine learning algorithms to classify application domains and tasks within user stories. We did not make use of advanced artificial intelligence solutions, *e.g.*, deep learning algorithms. While they might have offered additional insights into the capabilities of our framework, we favored the analysis of simpler models which are less demanding in terms of computation costs and training data, other than being more interpretable. The high classification performance obtained by those simpler models increased the confidence of our design choices, even though we plan to investigate the contribution of more complex classification models as part of our future research agenda.

Threats to Conclusion Validity. We assessed the classification components of ReFair by experimenting with multiple classifiers and word embedding techniques, computing well-established metrics, i.e., F1-Score and accuracy, that have been widely used to comprehensively assess multiclass and multilabel classification algorithms [57, 59]. As for the sensitive feature recommender, we defined evaluation metrics that could well represent the capabilities of ReFair in recommending appropriate sensitive features.

Another threat concerns the use of the same dataset to train and test our approach. In this respect, there are three considerations to make. First, we had no alternatives than using the synthetic dataset, given the lack of alternatives in literature. Second, we made sure not to mix training and testing data by performing a ten-fold cross validation: in each iteration one fold was retained as the test set and left untouched, while the remaining folds were used for training.

By doing that, we ensured the ReFair was always experimented against unseen data. Last but not least, after addressing our research questions relying on the synthetic dataset, we performed a more qualitative investigation into the performance of ReFair on a manually-generated dataset of USs. This qualitative analysis was explicitly designed to verify how ReFair may work when applied on a dataset different than the one used for training. The conclusions drawn in our qualitative study still pointed out the promising performance of our approach: of course, we are aware of the need for further, larger-scale evaluations of ReFair but, at the same time, we believe that the results provided so far represent a valuable contribution to the research community.

7 CONCLUSION

We proposed and assessed ReFair, an automated framework to support fairness requirements engineering. Our findings showed that ReFair can accurately classify sensitive features from USs.

Our future research agenda includes further experimentation on ReFair, including (1) a larger-scale qualitative and industrial assessment of the framework; (2) extensions making the framework independent from the format used to develop requirements specifications; and (3) the integration of ReFair with bias mitigation strategies applied in later development stages. Also, we plan to conduct further investigations into the generalizability of the automated generation of USs for requirements engineering research.

ACKNOWLEDGMENT

This work has been partially supported by the QUAL-AI and FRINGE national research projects funded by the MUR under the PRIN 2022 and PRIN 2022 PNRR programs (Contracts 2022B3BP5S and P2022553SL, respectively). The work has been also partially supported by the EMELIOT national project funded by the MUR under the PRIN 2020 program (Contract 2020W3A5FY).

REFERENCES

[1] 2024. *ReFAIR: Toward a Context-Aware Recommender for Fairness Requirements Engineering*. Zenodo. https://doi.org/10.5281/zenodo.10470916

[2] Muhammad Aurangzeb Ahmad, Carly Eckert, and Ankur Teredesai. 2018. Interpretable machine learning in healthcare. In *Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics*. 559–560.

[3] Solon Barocas, Moritz Hardt, and Arvind Narayanan. 2017. Fairness in machine learning. *Nips tutorial* 1 (2017), 2.

[4] Dane Bertram. 2007. Likert scales. *Retrieved November* 2, 10 (2007), 1–10.

[5] Seblewongel Esseynew Biable, Nuno Manuel Garcia, Dida Midekso, and Nuno Pombo. 2022. Ethical Issues in Software Requirements Engineering. *Software* 1, 1 (2022), 31–52. https://doi.org/10.3390/software1010003

[6] Sumon Biswas and Hridesh Rajan. 2021. Fair Preprocessing: Towards Understanding Compositional Fairness of Data Transformers in Machine Learning Pipeline. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Athens, Greece) *(ESEC/FSE 2021)*. Association for Computing Machinery, New York, NY, USA, 981–993. https://doi.org/10.1145/3468264.3468536

[7] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606* (2016).

[8] Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large Language Models in Machine Translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Association for Computational Linguistics, Prague, Czech Republic, 858–867. https://aclanthology.org/D07-1090

[9] Bernd Bruegge and Allen H Dutoit. 2010. *Object-Oriented Software Engineering Using UML, Patterns, and Java™ Third Edition*. by Pearson Education, Inc.,.

[10] Yuriy Brun and Alexandra Meliou. 2018. Software fairness. In *Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*. 754–759.

[11] Marc-Etienne Brunet, Colleen Alkalay-Houlihan, Ashton Anderson, and Richard Zemel. 2019. Understanding the Origins of Bias in Word Embeddings. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 803–811. https://proceedings.mlr.press/v97/brunet19a.html

[12] Francesco Casillo, Vincenzo Deufemia, and Carmine Gravino. 2022. Detecting privacy requirements from User Stories with NLP transfer learning models. *Information and Software Technology* 146 (2022), 106853. https://doi.org/10.1016/j.infsof.2022.106853

[13] L. Elisa Celis, Lingxiao Huang, Vijay Keswani, and Nisheeth K. Vishnoi. 2019. Classification with Fairness Constraints: A Meta-Algorithm with Provable Guarantees. In *Proceedings of the Conference on Fairness, Accountability, and Transparency* (Atlanta, GA, USA) *(FAT* '19)*. Association for Computing Machinery, New York, NY, USA, 319–328. https://doi.org/10.1145/3287560.3287586

[14] Joymallya Chakraborty, Suvodeep Majumder, and Tim Menzies. 2021. Bias in machine learning software: why? how? what to do?. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 429–440.

[15] Kai-Wei Chang, Vinodkumar Prabhakaran, and Vicente Ordonez. 2019. Bias and fairness in natural language processing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): Tutorial Abstracts*.

[16] Tianqi Chen and Carlos Guestrin. 2016. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. https://doi.org/10.1145/2939672.2939785

[17] Zhenpeng Chen, Jie M. Zhang, Federica Sarro, and Mark Harman. 2022. MAAT: A Novel Ensemble Approach to Addressing Fairness and Performance Bugs for Machine Learning Software. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Singapore, Singapore) *(ESEC/FSE 2022)*. Association for Computing Machinery, New York, NY, USA, 1122–1134. https://doi.org/10.1145/3540250.3549093

[18] Alexandra Chouldechova and Aaron Roth. 2020. A snapshot of the frontiers of fairness in machine learning. *Commun. ACM* 63, 5 (2020), 82–89.

[19] Mike Cohn. 2004. *User stories applied: For agile software development*. Addison-Wesley Professional.

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]

[21] Mengnan Du, Fan Yang, Na Zou, and Xia Hu. 2020. Fairness in deep learning: A computational perspective. *IEEE Intelligent Systems* 36, 4 (2020), 25–34.

[22] Nicolau Duran-Silva, Enric Fuster, Francesco Alessandro Massucci, César Parra-Rojas, Arnau Quinquillà, Fernando Roda, Bernardo Rondelli, Nicandro Bovenzi, and Chiara Toietta. 2021. A controlled vocabulary for research and innovation in the field of Artificial Intelligence (AI). https://doi.org/10.5281/ZENODO.5591987

[23] Alessandro Fabris, Stefano Messina, Gianmaria Silvello, and Gian Antonio Susto. 2022. Algorithmic fairness datasets: the story so far. *Data Mining and Knowledge Discovery* 36, 6 (Sept. 2022), 2074–2152. https://doi.org/10.1007/s10618-022-00854-z

[24] Anthony Finkelstein, Mark Harman, S Afshin Mansouri, Jian Ren, and Yuanyuan Zhang. 2008. "Fairness analysis" in requirements assignments. In *2008 16th IEEE International Requirements Engineering Conference*. IEEE, 115–124.

[25] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. 2017. Fairness testing: testing software for discrimination. In *Proceedings of the 2017 11th Joint meeting on foundations of software engineering*. 498–510.

[26] A. Géron. 2019. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media. https://books.google.ch/books?id=HnetDwAAQBAJ

[27] Tobias Hey, Jan Keim, Anne Koziolek, and Walter F. Tichy. 2020. NoRBERT: Transfer Learning for Requirements Classification. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*. 169–179. https://doi.org/10.1109/RE48521.2020.00028

[28] Frederic R. Hopp, Jacob T. Fisher, Devin Cornell, Richard Huskey, and René Weber. 2020. The extended Moral Foundations Dictionary (eMFD): Development and applications of a crowd-sourced approach to extracting moral intuitions from text. *Behavior Research Methods* 53, 1 (July 2020), 232–246. https://doi.org/10.3758/s13428-020-01433-0

[29] Max Hort, Jie M. Zhang, Federica Sarro, and Mark Harman. 2021. Fairea: A Model Behaviour Mutation Approach to Benchmarking Bias Mitigation Methods. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Athens, Greece) *(ESEC/FSE 2021)*. Association for Computing Machinery, New York, NY, USA, 994–1006. https://doi.org/10.1145/3468264.3468565

[30] Xiaolei Huang, Linzi Xing, Franck Dernoncourt, and Michael J. Paul. 2020. Multilingual Twitter Corpus and Baselines for Evaluating Demographic Bias in Hate

Speech Recognition. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*. European Language Resources Association, Marseille, France, 1440–1448. https://aclanthology.org/2020.lrec-1.180

[31] Irum Inayat, Siti Salwah Salim, Sabrina Marczak, Maya Daneva, and Shahaboddin Shamshirband. 2015. A systematic literature review on agile requirements engineering practices and challenges. *Computers in human behavior* 51 (2015), 915–929.

[32] Ammar Ismael Kadhim. 2019. Survey on supervised machine learning techniques for automatic text classification. *Artificial Intelligence Review* 52, 1 (2019), 273–292.

[33] Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the association for computational linguistics* 3 (2015), 211–225.

[34] Olga Liskin, Raphael Pham, Stephan Kiesling, and Kurt Schneider. 2014. *Why We Need a Granularity Concept for User Stories*. Springer-Verlag, Berlin, Heidelberg, 110–125.

[35] Nicholas Lourie, Ronan Le Bras, and Yejin Choi. 2021. Scruples: A corpus of community ethical judgments on 32,000 real-life anecdotes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 13470–13479.

[36] Oscar Luaces, Jorge Díez, Jose Barranquero, Juan del Coz, and Antonio Bahamonde. 2012. Binary relevance efficacy for multilabel classification. *Progress in Artificial Intelligence* 1 (12 2012). https://doi.org/10.1007/s13748-012-0030-x

[37] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn E. M. van der Werf, and Sjaak Brinkkemper. 2016. The Use and Effectiveness of User Stories in Practice. In *Requirements Engineering: Foundation for Software Quality*.

[38] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–35.

[39] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A Survey on Bias and Fairness in Machine Learning. *ACM Comput. Surv.* 54, 6, Article 115 (jul 2021), 35 pages. https://doi.org/10.1145/3457607

[40] Marina Meilă. 2007. Comparing clusterings—an information based distance. *Journal of Multivariate Analysis* 98, 5 (2007), 873–895. https://doi.org/10.1016/j.jmva.2006.11.013

[41] Ahmed Metwally and Chun-Heng Huang. 2019. Scalable Similarity Joins of Tokenized Strings. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 1766–1777. https://doi.org/10.1109/ICDE.2019.00193

[42] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs.CL]

[43] Claire Cain Miller. 2015. Can an algorithm hire better than a human. *The New York Times* 25 (2015).

[44] Zafeiria Moumoulidou, Andrew McGregor, and Alexandra Meliou. 2020. Diverse Data Selection under Fairness Constraints. *arXiv preprint arXiv:2010.09141* (2020).

[45] Fatemeh Nargesian, Abolfazl Asudeh, and H. V. Jagadish. 2022. Responsible Data Integration: Next-Generation Challenges. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 2458–2464. https://doi.org/10.1145/3514221.3522567

[46] Parmy Olson. 2011. The algorithm that beats your bank manager. *CNN Money March* 15 (2011).

[47] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1532–1543. https://doi.org/10.3115/v1/D14-1162

[48] Dana Pessach and Erez Shmueli. 2022. A Review on Fairness in Machine Learning. *ACM Comput. Surv.* 55, 3, Article 51 (feb 2022), 44 pages. https://doi.org/10.1145/3494672

[49] David M. W. Powers. 2020. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. arXiv:2010.16061 [cs.LG]

[50] Jesse Read, Bernhard Pfahringer, Geoffrey Holmes, and Eibe Frank. 2021. Classifier Chains: A Review and Perspectives. *Journal of Artificial Intelligence Research* 70 (feb 2021), 683–718. https://doi.org/10.1613/jair.1.12376

[51] Jörg Rech and Klaus-Dieter Althoff. 2004. Artificial intelligence and software engineering: Status and future trends. *KI* 18, 3 (2004), 5–11.

[52] Brittany Reid, Markus Wagner, Marcelo d'Amorim, and Christoph Treude. 2022. Software Engineering User Study Recruitment on Prolific: An Experience Report. *arXiv preprint arXiv:2201.05348* (2022).

[53] Ian Sommerville. 2010. *Software Engineering* (9 ed.). Addison-Wesley, Harlow, England.

[54] Ezekiel Soremekun, Mike Papadakis, Maxime Cordy, and Yves Le Traon. 2022. Software fairness: An analysis and survey. *arXiv preprint arXiv:2205.08809* (2022).

[55] P. Szymański and T. Kajdanowicz. 2017. A scikit-based Python environment for performing multi-label classification. *ArXiv e-prints* (Feb. 2017). arXiv:1702.01460 [cs.LG]

[56] Adi L Tarca, Vincent J Carey, Xue-wen Chen, Roberto Romero, and Sorin Drăghici. 2007. Machine learning and its applications to biology. *PLoS computational biology* 3, 6 (2007), e116.

[57] Grigorios Tsoumakas and Ioannis Katakis. 2007. Multi-label classification: An overview. In *International Conference on Data Mining and Knowledge Discovery*. Springer, 1–15.

[58] V. Tzerpos and R.C. Holt. 1999. MoJo: a distance metric for software clusterings. In *Sixth Working Conference on Reverse Engineering (Cat. No.PR00303)*. 187–193. https://doi.org/10.1109/WCRE.1999.806959

[59] C. J. van Rijsbergen. 1979. Information Retrieval. In *ACM SIGSPATIAL International Workshop on Advances in Geographic Information Systems*.

[60] Sahil Verma and Julia Rubin. 2018. Fairness Definitions Explained. In *Proceedings of the International Workshop on Software Fairness* (Gothenburg, Sweden) *(FairWare '18)*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/3194770.3194776

[61] Andreas Vogelsang and Markus Borg. 2019. Requirements engineering for machine learning: Perspectives from data scientists. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. IEEE, 245–251.

[62] Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman (Eds.). 56 – 61. https://doi.org/10.25080/Majora-92bf1922-00a

[63] Jules White, Sam Hays, Quchen Fu, Jesse Spencer-Smith, and Douglas C Schmidt. 2023. ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design. *arXiv preprint arXiv:2303.07839* (2023).

[64] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science and Business Media.

[65] Jianlong Zhou and Fang Chen. 2018. *Human and Machine Learning*. Springer.

[66] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910* (2022).