

Technical Report of MANILA: A Framework to Democratize the Quality-Based Machine Learning Development Through Extended Feature Models

Giordano d'Aloisio*, Antinisca Di Marco, Giovanni Stilo

*^aDepartment of Information Engineering and Information Sciences and Mathematics,
University of L'Aquila, Via Vetoio, L'Aquila, 67100, Italy*

Abstract

Machine learning (ML) systems have become an essential tool for experts of many domains, data scientists, and researchers, helping them to find answers to many complex business questions starting from raw datasets. Nevertheless, the development of ML systems able to satisfy the stakeholders' needs requires an appropriate amount of knowledge about the ML domain. Over the years, several solutions have been proposed to automate the development of ML systems. However, an approach taking into account the new quality concerns needed by ML systems (like fairness, effectiveness, privacy, and others) is still missing.

We target the *fairness* and *expressiveness* quality attributes by proposing MANILA, a novel low-code application to support the development of fair and effective ML systems. The main idea behind the proposed approach is that we can think of the quality-based development process of ML systems as a Software Product Line (SPL) implemented through the Extended Feature Model formalism. In this work, we extend our previous publication [1] by evaluating MANILA's *expressiveness* and *correctness* by replicating three different real-world fairness evaluations and showing how the results of the experiments implemented with MANILA are statistically equal to the original ones. In addition, we conduct a user evaluation of our approach to highlight and discuss its strengths and main points of improvement for its usability.

*Corresponding author

Email addresses: `giordano.daloisio@graduate.univaq.it` (Giordano d'Aloisio), `antinisca.dimarco@univaq.it` (Antinisca Di Marco), `giovanni.stilo@univaq.it` (Giovanni Stilo)

Keywords: Machine Learning System, Fairness, Low-code development, Software Product Line

1. Introduction

Machine Learning (ML) systems are increasingly becoming used instruments, applied to all application domains, and affect our real life. The development of ML systems requires a high knowledge of the underlying ML approaches to choose the best techniques, models, and measures to solve the target problem. In recent years, many methods have been developed to automate some phases of ML system development and help non-technical users (e.g., [2, 3, 4]). However, these techniques do not consider the *quality properties* essential for ML systems, like the privacy of the data set or the interpretability, explainability and fairness of the model [5, 6, 7].

Indeed, if we consider the impact that ML applications have on our lives, it is clear how ensuring that these quality properties are satisfied is of paramount importance. This importance is also highlighted by some of the 17 Sustainable Development Goals (SDG) proposed by the United Nations [8], in primis SDG 5 (Gender Equality) or SDG 10 (Reduced Inequalities). In addition, the relevance of these quality properties is also highlighted by the *AI Act* proposal of the European Commission, where *effectiveness* (i.e., how good an ML model is in predicting outcomes [9]), and *fairness* (i.e., the absence of any prejudice or favouritism toward an individual or group based on their inherent or acquired characteristics [10]) are described as key quality properties for *high-risk* AI-enabled systems [11].

In this paper, we target these two quality properties (i.e., *effectiveness* and *fairness*) by presenting MANILA (**M**odel **bA**sed developme**N**t of mach**I**ne **L**earning systems with qu**A**lity), a novel approach that will democratise the quality-based development of ML systems by means of a low-code platform [12]. The goal of our approach is to provide an environment for the automatic configuration and implementation of experiments that automatically selects the ML system (i.e., the ML Algorithm and the fairness enhancing method) to better satisfy a given quality requirement. The requirement is satisfied by finding the best trade-off between effectiveness and fairness. This will simplify the work of data scientists who are not practical in the quality-based development of ML systems, and will guarantee fairness by reducing inequalities in contexts where applying ML systems is ethically critical (e.g., justice [13] or education [14]).

Note that this paper extends our previous publication [1] by adding the following new contributions:

- We extend the discussion of related works by including new papers employing a software engineering approach to model fairness experiments;
- We extensively evaluate MANILA in terms of both *expressiveness* and *correctness*;
- We conduct a user evaluation of MANILA to highlight its strengths and limitations, and to identify its main points of improvement for its usability.

The rest of this paper is structured as follows: Section 2 presents a list of related works distinguishing among motivating papers and related approaches; Section 3 presents a general quality evaluation workflow that we use as a reference in the rest of the paper; in Section 4 we describe the architecture of MANILA and its components; Section 5 is devoted to describing the *correctness* and *expressiveness* evaluation and the user evaluation of MANILA; Section 6 discusses some possible threats to validity of our approach, and, finally Section 7 concludes the paper also presenting some future works.

2. Related Work

In this section, we describe related work related of our proposed approach. We first describe a list of papers that motivate the need for quality assurance in ML systems, and we describe how our approach aims to solve some of the underlying concerns. Next, we describe a set of works similar to our proposed approach, highlighting for each one the differences with MANILA.

2.1. Motivating Papers

The problem of quality assurance in machine learning systems has gained much relevance in the last years. Many articles highlight the need to define and formalise new standard quality attributes for machine learning systems [5, 6, 7, 15, 16, 17]. Most of the work in the literature focuses on either the identification of the most relevant quality attributes for ML systems or on the formalisation of them in the context of ML system development.

Concerning the identification of quality attributes in ML systems, the authors of [18, 19] identify three main components in which quality attributes

can be found: **Training Data**, **ML Models** and **ML Platforms**. The quality of **Training Data** is generally evaluated with properties such as *privacy*, *bias*, *completeness* and *missing values*, *expressiveness*. For **ML Model**, the authors mean the trained model used by the system. The quality of this component is generally evaluated by *fairness*, *explainability*, *interpretability*, *security*. Finally, **ML Platform** is the implementation of the system, which is mainly affected by *security*, *performance reliability*, and *availability*. Mucini et al.[5] identified a set of quality properties as constraints of stakeholders and highlighted the need to consider them during the *Architecture Definition* phase. The quality attributes are: *data quality*, *ethics*, *privacy*, *fairness*, *ML models' performance*, etc. Martinez-Fernández et al. also highlights in [6] the need to formalise quality properties in ML systems and update the software quality requirements defined by ISO 25000 [20]. The most relevant properties highlighted by the authors concern *ML safety*, *ML ethics*, and *ML explainability*. In our work, we focus on quality properties that arise during the development of ML systems such as *fairness*, *explainability*, *interpretability*, and datasets *privacy*, while we leave other quality properties (e.g., *performance*) that arise during other phases (e.g., deployment) for future work. In particular, in this paper we target *fairness*, also in relation to *expressiveness*.

Many solutions have been proposed to formalise and model the standard quality assurance process in ML systems. Amershi et al. have been the first authors to identify a set of common steps that identify the development of each ML system [21]. In particular, each ML system is identified by nine stages that go from data collection and cleaning, to model training and evaluation, and finally to the deployment and monitoring of the ML model. Their work has been the foundation of many subsequent papers on the quality modelling of ML systems. *CRISP_ML* (*Cross-Industry Standard Process model for Machine Learning*) is a process model proposed by Studer et al. [22], extending the more well known *CRISP_DL* [23] process model to ML systems. They identify a set of common phases for the building of ML systems, namely: *Business and Data understanding*, *Data preparation*, *Modelling*, *Evaluation*, *Deployment*, *Monitoring and Maintenance*. For each phase, the authors identify a set of functional quality properties to ensure the quality of such systems like *robustness* (i.e., the ability of an ML model to guarantee a constant level of correctness in the predictions during time), *explainability*, and *computational complexity*. Similarly, the *Quality for Artificial Intelligence (Q4AI)* consortium proposed a set of guidelines [24] for the quality assurance of ML systems for specific domains: *generative systems*,

operational data in process systems, voice user interface system, autonomous driving and AI Optical Character Recognition. For each domain, the authors identify a set of properties and metrics to ensure quality properties like *robustness*, and *explainability*. Concerning the modelling of quality requirements, Azimi et al. proposed a layered model for the quality assurance of machine learning systems in the context of the Internet of Things (IoT) [25]. The model consists of two layers: *Source Data* and *ML Function/Model*. For *Source Data*, a set of quality attributes is defined: *completeness, consistency, conformity, accuracy, integrity, timeliness*. Machine learning models are instead classified into *predictors, estimators* and *adapters* and a set of quality attributes are defined for each of them: *accuracy, correctness, completeness, effectiveness, optimality*. Each system is then influenced by a subset of quality characteristics based on the type of ML model and the required data. Ishikawa proposed, instead, a framework for the quality evaluation of an ML system [26]. The framework defines these components for ML applications: *dataset, algorithm, ML component* and *system*, and, for each of them, proposes an argumentation approach to assess quality. Finally, Siebert et al. [27] proposed a formal modelling definition for quality requirements in ML systems. They start from the definition of the process in [23] and build a metamodel for the description of quality requirements. The meta-model is made of the following classes: *Entity* (which can be defined at various levels of abstraction, such as the whole system or a specific component of the system), *Property* (also expressed at different levels of abstraction), *Evaluation* and *Measure* related to the property. Starting from this meta-model, the authors build a tree model to evaluate the quality of the different components of the system. From this analysis, we can conclude that there is a robust research motivation in formalising and defining new quality attributes of the ML systems.

From this review, we can conclude how many attempts have been proposed to address quality constraints of ML systems, and several quality properties, metrics, and definitions of ML systems can now be extracted from the literature. However, a framework that guides the data scientist through developing ML systems satisfying quality properties is actually still missing. In this paper, we go in this direction by proposing MANILA, a novel approach that will democratise the quality-based development of ML systems through a low-code platform. In particular, we model a general workflow for the quality-based development of ML systems as a Software Product Line (SPL) [28] through the ExtFM formalism [29, 30]. Next, we demonstrate how it

is possible to generate an actual implementation of such workflow from a low-code experiment configuration and how this workflow is actually able to suggest the best methods to satisfy a given quality requirement. Recalling the ML development process of [21], MANILA focuses on the *model training* and *model evaluation* development steps by guiding the data scientist in selecting the ML system (i.e., ML algorithm and quality improvement method) that better meets a given quality attribute. In particular, in this paper, we focus on the *fairness* quality attribute, while other attributes will be explored in future works.

2.2. Related Approaches

Several approaches have been proposed in the last few years regarding either the adoption of Feature Models to model ML systems or helping users in the quality-based development of ML systems.

Regarding the adoption of Feature Models to model ML systems, a similar approach was used by Di Sipio et al. in [31]. In their work, the authors used feature models to model ML pipelines for Recommender Systems. The variation points are identified by all the components needed to implement a recommender system (e.g., the ML algorithm to use or the Python libraries for the implementation). However, they do not consider quality attributes in their approach.

In terms of helping users assess quality attributes in ML systems, there is an intense research activity mainly related to the fairness domain [32]. In general, the problem of fairness assurance can be defined as a search-based problem among different ML algorithms and fairness methods [32]. An example of a search-based approach to identify the best combination of the ML classifier and the fairness enhancing method is the *Fairkit-learn* library proposed by Johnson and Brun in [33]. In their work, the authors present a Python library (similar to the well-known *scikit-learn* library [34]) that uses a grid-search approach to identify the best combination of the fairness enhancing method and ML classifier, along with the best configuration of hyper-parameters for the given classifier and fairness enhancing methods. In our work, we adopt a similar grid search-based approach to select the best combination of the ML classifier and the fairness method. However, instead of a library that may be difficult to use for non-technical users [35], we propose a low-code application to guide non-expert users in the quality-based development of ML systems. In addition, MANILA guides in the

definition and generation of experiments that are always executable (i.e., not leading to errors).

Instead, a tool that is close to our approach is the model-driven bias assessment and mitigation framework proposed by Yohannis and Kolovos in [36]. In particular, the authors have first defined a meta-model to represent bias measurement and mitigation scripts. Then, they defined a Domain Specific Language (DSL) to specify such scripts. Finally, from a script model defined using such a DSL, the proposed tool automatically generates a Python implementation of the script. The authors have also presented two decision trees that guide users in selecting, respectively, the most accurate fairness metrics and methods based on the purpose of the experiment. MANILA differs from this approach in three aspects: *(i)* our approach relies on the Extended Feature Model formalism to model the quality-based ML development process as an SPL; *(ii)* MANILA is designed to be easily extended to include other quality attributes and find the best trade-off among them; *(iii)* MANILA guides users in selecting features (i.e., ML algorithms and fairness-enhancing methods) that always lead to an executable experiment script (i.e., that does not lead to execution errors). However, we will also include the purpose-orientated feature selection approach of [36] in our future works. In this way, MANILA will guide users in selecting features that are both compliant with already selected ones and also compliant with the purpose of their experiments.

3. Considered Quality Evaluation Process

Consider now a data scientist who needs to understand which is the best ML system able to satisfy a given quality constraint. What they typically will do is perform a quality evaluation ad-hoc process to assess the set of ML and quality enhancing techniques able to satisfy the quality constraint better. They assume that the best possible quality result can be achieved by applying one of the available techniques in isolation.

Algorithm 1 reports the pseudo-code of a generic process to assess a generic QA during the development of an ML system.

The first step is selecting the dataset to use and possibly preparing it for the experiment (in this work, we assume that the dataset has already been preprocessed and is ready to train the ML model). Next, the data scientist selects the ML algorithms, the methods enhancing a QA, and the appropriate quality metrics for the evaluation. Then, for each of the chosen

Algorithm 1: Quality Evaluation Process pseudo-code

Input: Dataset d , ML Algorithms ML , QA Methods Q , QA Metric M

```
for  $m \in ML$  do
    for  $q \in Q$  do
        if  $q$  works on  $d$  then
            | apply  $q$  on  $d$ ;
        if  $q$  works on  $m$  before training then
            | apply  $q$  on  $m$ ;
         $f = \text{train } m$ ;
        if  $q$  works on  $f$  then
            | apply  $q$  on  $f$ ;
        compute selected metrics  $M$  on  $f$ ;
choose report technique;
evaluate the results;
 $Q = \text{select best QA Method}$ ;
 $M = \text{select best ML Algorithm}$ ;
 $F = \text{train } M \text{ with full dataset applying } Q$ ;
return  $F$ 
```

ML algorithms, they apply the selected quality methods and, according to their type, there can be the following options:

- if the quality method works on the training set, it has to be applied to the dataset before training the ML algorithm
- if the quality method works on the ML algorithm before training, then it has to be applied to the ML algorithm before the training phase
- if the method works on the trained ML algorithm, then it has to be applied after the training of the ML algorithm

Finally, the data scientist computes the selected metrics for the specific pair of ML and QA methods. After repeating the process for all the selected methods, they choose a reporting technique (e.g., table or chart), evaluate the obtained results collected in the report, and train with the entire dataset the ML algorithm performing better by applying the quality method that

better achieves the QA. If the data scientist has a threshold to achieve, then they can verify if at least one of the ML and quality methods combinations satisfies the constraint. If so, one of the suitable pairs is selected. Otherwise, they have to relax the threshold and repeat the process.

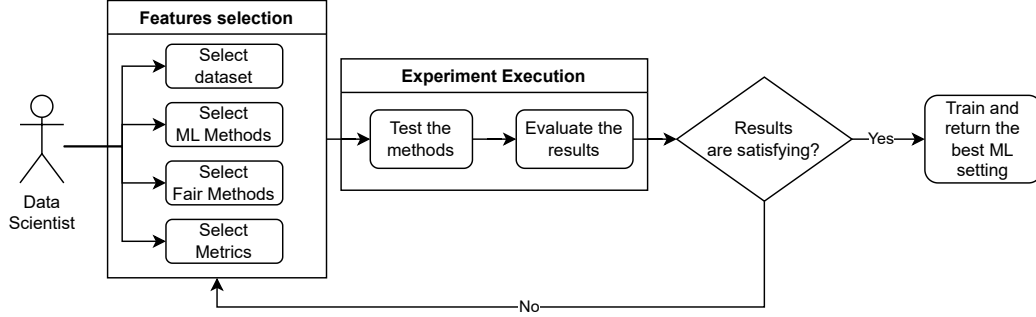


Figure 1: Manual execution of the Quality Evaluation Process

The Quality Evaluation Process described in Algorithm 1 can be generalized as a process of common steps describing any experiment in the considered domain with the same assumptions. Figure 1 sketches such a generalization. First, the data scientist selects all the experiment features, i.e., the dataset, the ML Methods, the methods assuring a specific QA and the related metrics (*Features Selection* step). Next, she runs the quality methods using the general approach described in algorithm 1 and evaluates the results (namely, *Experiment Execution*). If the results are satisfying (i.e., they satisfy the quality constraints), then the method with the best QA is returned. Otherwise, the data scientist has to repeat the process, possibly relaxing the QA constraints or working on the dataset.

The described Quality Evaluation Process is the foundation of MANILA that aims to formalise and democratise it by providing an SPL and ExtFM-based low-code framework that supports data scientists in developing quality ML systems.

4. MANILA Approach

In this section, we describe the new web-based version of MANILA, a low-code framework to formalise the quality-based development of ML systems. This work is based on the general experiment workflow described in Section 3, and on the results of the user evaluation described in Section 5.2.

Our approach aims to automate and ease the quality-based development of ML systems. We achieve this goal by proposing a framework to automatically generate a configuration of an experiment to find the ML system (i.e., ML algorithm and quality enhancing method) better satisfying a given QA. This framework will accelerate the quality-based development of ML systems making it accessible also to not experts.

Recalling the experimental workflow described in section 3, the set of ML models, quality methods and metrics can be considered *variation points* of each experiment, differentiating them from one another. For this reason, we can think of this family of experiments as a Software Product Line (SPL) specified by a Feature Model [37]. Indeed, Feature Models allow us to define a template for families of software products with standard features (i.e., components of the final system) and a set of variability points that differentiate the final systems [29, 38]. Features in the model follow a tree-like parent-child relationship and could be *mandatory* or *optional* [38]. Sibling features can belong to an *Or-relationship* or an *Alternative-relationship* [38]. Finally, there could be *Cross-tree* relationships among features not in the same branch. These relationships are expressed using logical propositions [38]. However, traditional Feature Models do not allow associating attributes to features, which are necessary in our case to represent a proper experiment workflow (for instance, to specify the label of the dataset or the number of rounds in cross-validation [39]). Hence, we relied on the concept of Extended Feature Models [29, 30] to represent the family of experiments workflows.

Figure 2 details a high-level picture of MANILA, where each rounded box represents a step in the quality-driven development process, while square boxes represent artefacts. MANILA has been implemented as a low-code web application through which all the steps of the quality-based development workflow can be performed. Near each artefact, we report the tools involved in its implementation.

The first step in the development process is the feature selection, in which the data scientist selects all the components of the quality-testing experiment through a dedicated web form. Next, a Python script implementing the experiment is automatically generated from the selected features. The generated experiment can be executed directly in the web application, or it can be downloaded and executed locally or embedded in other pipelines. After its execution, for each QA selected, the experiment returns:

1. a quality report reporting for each quality method and ML algorithm

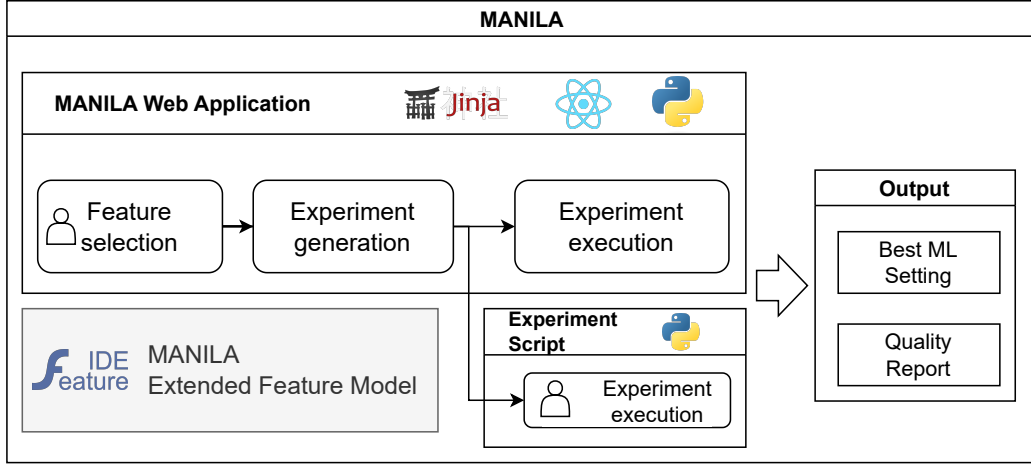


Figure 2: MANILA approach

the related metrics;

2. the ML algorithm with the applied quality enhancing method that better performs with the given QA, trained and ready for production.

The architecture of MANILA makes it easy to extend. In fact, adding a new method or metric to MANILA translates to adding a new feature to the web form and adding the proper code implementing it.

The basis of MANILA is the Extended Feature Model (ExtFM). The ExtFM is the template of all possible experiments a data scientist can perform and guides her through the quality-based development of an ML system. We used the ExtFM as a formalism to reason about the different relationships and constraints among features before implementing them in the web application.

MANILA is publicly available in the SoBigData research infrastructure (RI) [40] (after registration)¹, and its source code is available on GitHub². SoBigData is a European project that aims to develop and share analyses and tools in the field of Big Data following the Open Science principle. Its RI is built on top of the D4Science platform [41] and provides datasets, analyses, and methods on several data science topics. In this perspective,

¹<https://sobigdata.d4science.org/group/sobigdata.it/manila-univaq>

²<https://github.com/giordanoDaloisio/manila-web>

MANILA is provided as an application to assist data scientists in performing fairness evaluations using the datasets already available in the platform or by uploading new ones.

In the following, we detail first the ExtFM; then, we describe the web application and how each workflow step has been implemented.

4.1. Extended Feature Model

As already mentioned, the ExtFM is the basis of MANILA approach since it defines the template of all possible experiments a data scientist can generate. It has been implemented using *FeatureIDE*, an open-source graphical editor which allows the definition of ExtFMs [42].

Figure 3 shows a short version of the implemented ExtFM³. In particular, each experiment is defined by six macro features, which are then detailed by children’s features.

The first mandatory feature is the Dataset. The Dataset has a file extension (e.g., CSV, EXCEL, JSON, and others), and a Label which can be Binary or Multi-Class. The Label feature has two attributes specifying his name and the positive value (used to compute fairness metrics). The Dataset could also have one or more sensitive variables that identify sensitive groups subject to unfairness [10]. The sensitive variables have a set of attributes to specify their name and the privileged and unprivileged groups [10]. Finally, there is a feature to specify if the Dataset has only positive attributes. This feature has been included to define a cross-tree constraint with a scaler technique that requires only positive attributes (see table 1). All these features are modelled as abstract since they do not have a concrete implementation in the final experiment. The next feature is a Scaler algorithm, which is not mandatory and can be included in the experiment to scale and normalize the data before training the ML model [43]. Different scaler algorithms from the *scikit-learn* library [34] are listed as concrete children of this feature. Next, there is the macro-feature representing the ML Task to perform. This feature has not been modelled as mandatory since there are two fairness methods (i.e. Gerry Fair and Meta Fair [44, 45]) that embed a fair classification algorithm and so, if they are selected, the ML Task can not be specified. However, we included a cross-tree constraint requiring the selection of ML Task if any of these two methods are selected ($\neg \text{Gerry Fair} \wedge \neg \text{Meta Fair} \Rightarrow \text{ML Task}$).

³The whole picture can be downloaded here <https://bit.ly/3ZyvMrk>

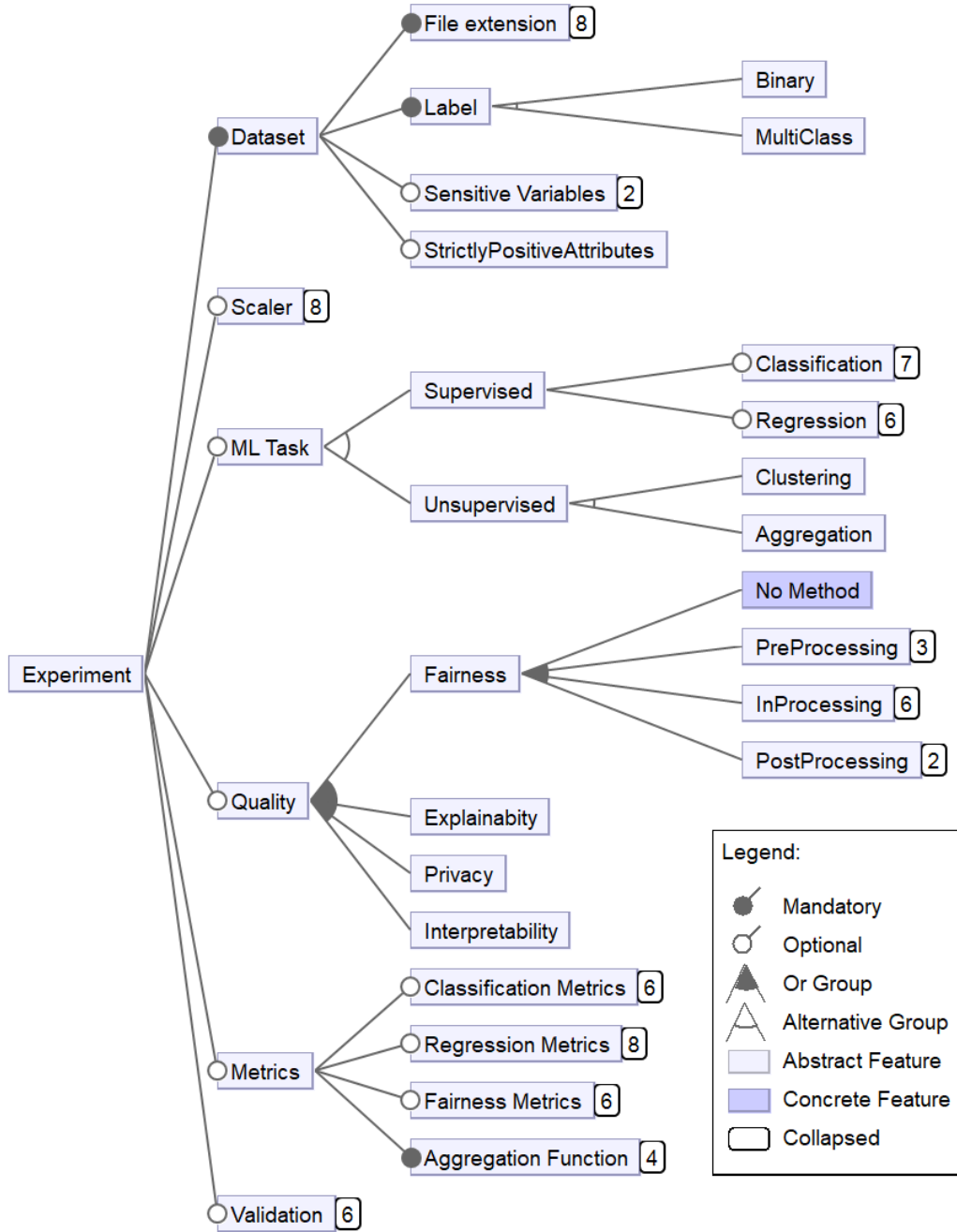


Figure 3: Short version of the implemented Extended Feature Model

An ML Task could be Supervised or Unsupervised. A Supervised task could be a Classification task or a Regression task and has an attribute to specify the size of the training set. These two abstract features are then detailed by a set of concrete implementations of ML methods selected from the *scikit-learn* library [34]. The Unsupervised learning task could be a Clustering or an Aggregation task. At this stage of the work, these two features have not been detailed and will be explored in future works. Next is the macro feature representing the system’s Quality Attributes. This feature is detailed by the four quality attributes elicited in our previous paper [1]. Effectiveness is not included in these features since it is an implicit quality of the ML methods and does not require adding other components (i.e. algorithms) in the experiment. At the time of this paper, the Fairness quality has been detailed, while the other properties will be deepened in future works. In particular, Fairness methods can be *Pre-Processing* (i.e. strategies that try to mitigate the bias on the dataset used to train the ML model [10, 46, 47]), *In-Processing* (i.e. methods that modify the behaviour of the ML model to improve fairness [10, 48]), and *Post-Processing* (i.e. methods that re-calibrate an already trained ML model to remove bias [10, 49]). These three features are detailed by several concrete features representing fairness-enhancing methods. In selecting such algorithms, we selected methods with a solid implementation, i.e., algorithms integrated into libraries such as *AIF360* [50] or *Fairlearn* [51] or algorithms with a stable source code such as *DEMV* [52] or *Blackbox* [49]. All these quality features have been implemented with an *Or-group* relationship. The last macro feature represents the Metrics to use in the experiment. Metrics are divided among Classification Metrics, Regression Metrics and Fairness Metrics. Each metric category has a set of concrete metrics selected from the *scikit-learn* library [34] and the *AIF360* library [50]. Based on the ML Task and the Quality Attributes selected, the data scientist must select the proper metrics to assess Correctness and the other Quality Attributes. This constraint is formalized by cross-tree relationships among features (see table 1). In addition, a set of Aggregation Functions must be selected if more than one metric is selected. The aggregation function combines the value of the other metrics to give an overall view of the method’s behaviour. Forward, there is the optional macro feature identifying the Validation function. Validation functions are different strategies to evaluate the Quality Attributes of an ML model [53]. Several Validation functions are available as children features, and there is an attribute to specify the number of groups in case of cross-validation [53].

Table 1: Extended Feature Model cross-tree constraints

Cross-tree constraints	Description
Fairness \Rightarrow Sensitive Variables	If the fairness QA is selected, then the sensitive variables must be specified
Fairness \Rightarrow Fairness Metrics	If the fairness QA is selected, then at least a fairness metric must be selected
Multiple Sensitive Var $\Rightarrow \neg$ Sampling $\wedge \neg$ Blackbox $\wedge \neg$ DIR	If multiple sensitive variables are selected, then disable the fairness enhancing methods not supporting more than one sensitive variable
MultiClass $\Rightarrow \neg$ Reweighing $\wedge \neg$ DIR $\wedge \neg$ Optimized Preprocessing $\wedge \neg$ LFR $\wedge \neg$ Adversarial Debiasing $\wedge \neg$ Gerry Fair $\wedge \neg$ Meta Fair $\wedge \neg$ Prejudice Remover $\wedge \neg$ Calibrated EO $\wedge \neg$ Reject Option	If a multi-class label is selected, then disable the fairness methods not supporting multi-class classification
Regression $\Rightarrow \neg$ PostProcessing $\wedge \neg$ Reweighing $\wedge \neg$ DIR $\wedge \neg$ DEMV $\wedge \neg$ Optimized Preprocessing $\wedge \neg$ LFR $\wedge \neg$ Adversarial Debiasing $\wedge \neg$ Gerry Fair $\wedge \neg$ Meta Fair $\wedge \neg$ Prejudice Remover	If the regression task is selected, then disable all the fairness methods not supporting this task
Exponentiated Gradient \vee Grid Search $\Rightarrow \neg$ MLP Classifier $\wedge \neg$ MLP Regressor	Exponentiated Gradient and Grid Search fairness methods do no work with MLP Classifier and MLP Regressor ML methods
\neg GerryFair $\wedge \neg$ MetaFair \Rightarrow ML Task	If not GerryFair and MetaFair fairness methods are selected, then an ML Task must be selected
Classification \iff Classification Metrics $\wedge \neg$ Regression Metrics	If the classification ML task is selected, then at least one classification and no regression metrics must be selected, and vice versa
Classification Metrics $\iff \neg$ Regression Metrics	If a classification metric is selected, then a regression metric must not be selected and vice versa
Regression \iff Regression Metrics $\wedge \neg$ Classification Metrics	If the regression ML task is selected, then at least one regression and no classification metrics must be selected, and vice versa
Box-Cox Method \Rightarrow Strictly Positive Attributes	If the Box-Cox scaler method is selected, then the dataset must have strictly positive attributes

Finally, table 1 lists the cross-tree constraints among features. These constraints have been then implemented in the web form of MANILA to guide the data scientist in selecting features always leading to a correct (i.e., executable) experiment.

4.2. Web Application

The features and the constraints defined in the ExtFM have been implemented into a low-code web application which is the core of MANILA. Through the web application, is it possible to perform all the steps of the quality-based development workflow described in Section 3. The web application has been implemented using the React framework for the front end and the Flask Python library for the back end. In the following, we detail how each workflow step has been implemented in the application.

4.2.1. Feature Selection

The feature selection step has been implemented in MANILA through a web form that includes all the features and the constraints defined in the ExtFM.

MANILA
Select the features that comprise your experiment

Dataset

File Extension
☒ CSV ☐ Parquet ☐ Excel ☐ JSON ☐ Text ☐ HTML ☐ XML ☐ HDF5

Label
☒ Binary ☐ MultiClass

Label Name * Positive Value *

Sensitive Variables *
☒ Single Sensitive Variable

Variable Name * Unprivileged value * Privileged value *

☐ Multiple Sensitive Variables

Variable Names * Unprivileged values * Privileged values *

Comma separated list of values Comma separated list of values Comma separated list of values

Figure 4: MANILA Web Form

Figure 4 shows a portion of the web form. The form comprises six sub-forms (one for each macro feature defined in the ExtFM). Each sub-form

includes all the *concrete* (i.e., non-abstract) children features of the relative macro feature in the ExtFM. For instance, in figure 4 is shown the sub-form relative to the *Dataset* macro feature, and it includes all the *Dataset* children features like *File Extension*, *Label*, and so on. Children with an *alternative* relationship in the ExtFM are implemented in the form either through a radio group (like the *File Extension* or *Label* fields in figure 4) or by a logical condition among fields (for instance, in figure 4 the *Multiple Sensitive Variables* field has been disabled because the *Single Sensitive Variable* field has been selected). In all other cases, features in the ExtFM have been implemented as checkbox fields in the form. Additional attributes related to the features (like the *Label Name* or *Positive Value* fields) have been implemented as text fields, which may be mandatory or not, depending on the case.

The cross-tree constraints defined in the ExtFM have been implemented as logical constraints among the different form fields. Figure 5 shows an example of such constraints. In the figure, it can be seen how the *Regression* field has been disabled. This is due to two reasons: first, the *Classification* ML task has already been selected, and second, the *Regression* task is incompatible with the fairness methods included so far. Hence, since the *Fairness* quality property has been selected, regression ML methods can not be selected; otherwise, they would lead to a non-executable experiment. This constraint is also shown to the user through a message saying that *Regression task is incompatible with fairness methods*. In addition, note how the *Reweighting* fairness method has been disabled as well. This is because the *Reweighting* method is not compatible with the *MLP Classifier* ML method that has already been selected. This constraint is also reported to the user through a message saying that *Reweighting is not compatible with MLP Classifier or MLP Regressor*.

Concerning the selection of fairness metrics, we included a set of questions (inspired from [36]) to help the data scientist select the proper ones. Figure 6 reports the set of questions. The first question aims at identifying if the fairness definition to assess is an *individual* (i.e., similar individuals should be treated similarly) or *group* (i.e., individuals of a specific group should not be discriminated) fairness definition [10]. If the data scientist selects *individual*, then the three individual fairness metrics implemented in the `aif360` library (i.e., *Euclidean Distance*[54], *Manhattan Distance*[55] and *Mahalanobis Distance*[56]) are shown. Instead, if the data scientist chooses the *group* fairness definition, another question aims to identify the specific category of group

☒ Classification
 ☐ Regression

Classification Methods

- ☒ Logistic Regression
- ☐ Support Vector Classifier
- ☐ Gradient Descent Classifier
- ☐ Gradient Boosting Classifier
- ☒ MLP Classifier
- ☐ Decision Tree Classifier
- ☐ Random Forest Classifier

Regression task is not compatible with fairness methods


Regression Methods

- ☐ Linear Regression
- ☐ Support Vector Regressor
- ☐ Gradient Descent Regressor
- ☐ Gradient Boosting Regressor
- ☐ MLP Regressor
- ☐ Decision Tree Regressor

☐ Save Semi-Trained Model

Train size

☒ Fairness

 Select at least one fairness method

☐ No Method

Pre Processing
 These methods work on the training dataset to reduce its intrinsic bias

- ☐ Reweighing

Not compatible with MLP Classifier or MLP Regressor
- ☐ DIR
- ☐ DEMV

Figure 5: Example of web form cross-tree constraints

fairness definitions. In particular, the data scientist has to specify if she is interested in *equal* fairness (i.e., everyone should have the same probability of receiving the positive label predicted [10]), *proportional* fairness (i.e., everyone should have the positive label predicted only if the other variables,

Fairness Metrics

1. Are you dealing with bias on **individuals** (similar individuals should be treated similarly) , **groups** (individuals of a group should not be discriminated)?
 - ☐ Individual
 - ☒ Group
2. Should different groups be treated **equally** (everyone should have the same probability of receiving a positive label), **proportionally** (everyone should get a positive label only if the evidence tells that), or **other**?
 - ☐ Equally
 - ☒ Proportionally
 - ☒ Equalized Odds Difference
0 means fairness
 - ☐ Average Odds Difference
0 means fairness
 - ☐ Other

Figure 6: Fairness metric selection

different from the sensitive ones, tell that [10]), or *other* fairness definitions. Based on the selected definition, then a set of metrics is shown. In particular, following the work of [36], for the *equal* category, we have included *Statistical Parity* and *Disparate Impact* fairness metrics [57]. For the proportional category, we have included the *Equalized Odds Difference* [58] and *Average Odds Difference* [59]. Finally, for the *other* category, we included the *True Positive Difference* and *False Positive Difference* [60].

Upload dataset

Upload a file if you want to run the experiment on the server (the file extension must be the same as above).

Figure 7: File upload field and execution buttons

Finally, figure 7 shows a last field in the form that allows the user to upload their dataset to run the generated experiment on the server. Below are two buttons allowing respectively to download the generated code and run the experiment on the server. Note how in the figure, the buttons are disabled because not all the constraints defined in the form are met.

4.2.2. Experiment Generation and Execution

The experiment can be generated and executed after selecting a set of features that meet all the form’s constraints.

The generation process starts by either clicking on the *Generate Code* or *Run the experiment* buttons shown in figure 7. The first button starts the code generation process and makes the generated code available for download. In contrast, the second generates the code and executes the generated code directly on the server showing the output to the user. The code generation process is performed through the Jinja template engine [61].

In general, the experiment applies each ML algorithm with each fairness method. It returns a report with the selected metrics and the ML setting (i.e., ML method and fairness enhancing method), achieving the best fairness and effectiveness trade-off. As already reported, it is worth noticing how the *effectiveness* is an intrinsic property of the ML algorithms and so no enhancing method is required. Figure 8 reports an example of how the quality

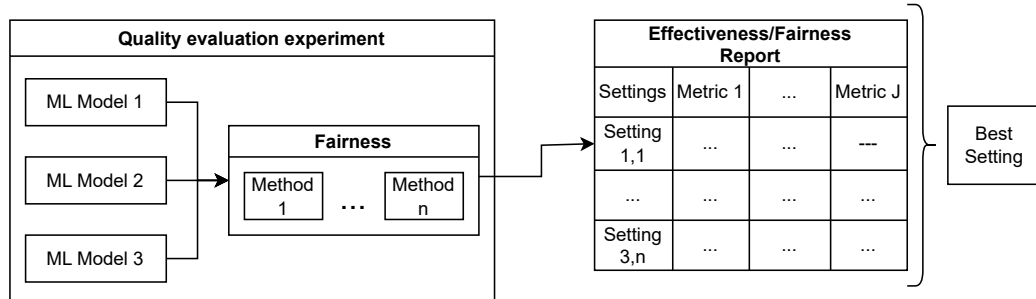


Figure 8: Quality evaluation process example

evaluation process is done in MANILA. In this example, the data scientist has selected three ML algorithms and wants to assure Fairness. She has chosen n methods to enhance Fairness and has selected j metrics for Fairness and effectiveness. Then, the testing process applies the n fairness methods to each ML algorithm and computes the j fairness metrics. Finally, the

process returns a report synthesizing the obtained results for Fairness and Effectiveness along with the best Fairness and Effectiveness setting.

```

1  from utils import cross_val
2  from methods import FairnessMethods
3  ...
4
5  ml_methods = {
6      'logreg': LogisticRegression(),
7      'gradient': GradientBoostingClassifier(),
8  }
9  fairness_methods = {
10     'no_method': FairnessMethods.NO_ONE,
11     'preprocessing': [
12         FairnessMethods.DEMV,
13     ],
14     'inprocessing': [
15         FairnessMethods.EG,
16         FairnessMethods.GRID,
17     ],
18     'postprocessing': []
19 }
20 base_metrics = {
21     'stat_par': [],
22     'eq_odds': [],
23     'zero_one_loss': [],
24     'disp_imp': [],
25     'acc': [],
26     'hmean': [],
27 }
28 for m in ml_methods.keys():
29     model = Pipeline([
30         ('scaler', StandardScaler()),
31         ('classifier', ml_methods[m])
32     ])
33     for f in fairness_methods.keys():
34         model = deepcopy(model)
35         data = data.copy()
36         if f == 'preprocessing':
37             for method in fairness_methods[f]:
38                 model_fair, ris_metrics = cross_val(..., model=model,
39                                                         metrics=base_metrics, preprocessor=method)
40                 ...
41             elif f == 'inprocessing':
42                 for method in fairness_methods[f]:
43                     model_fair, ris_metrics = cross_val(..., model=model,
44                                                         metrics=base_metrics, inprocessor=method)
45                     ...
46             elif f == 'postprocessing':
47                 for method in fairness_methods[f]:
48                     model_fair, ris_metrics = cross_val(..., model=model,
49                                                         metrics=base_metrics, postprocessor=method)
50                     ...

```

Listing 1: Portion of the generated experiment code

To have a more concrete visualization of how the experimental evaluation is conducted, Listing 1 reports a small portion of the generated Python code. In the code, there are three dictionaries containing the list of ML methods, fairness methods, and metrics to use (namely `ml_methods`, `fairness_methods`, and `base_metrics`, respectively). Next, there is a *for* loop exploring the list of ML methods and, for each method, the function creates a pipeline including a preprocessing method (in this example, the `StandardScaler` preprocessing approach). Finally, a nested *for* loop explores the list of fairness methods, and, for each of them, a function performing the evaluation is called according to the fact that the fairness method is a *preprocessing*, *inprocessing*, or *postprocessing* one. It is worth noting how the depicted generated experiment may vary based on the features selected. However, most sections of the experiment are general and are not related to the selected features.

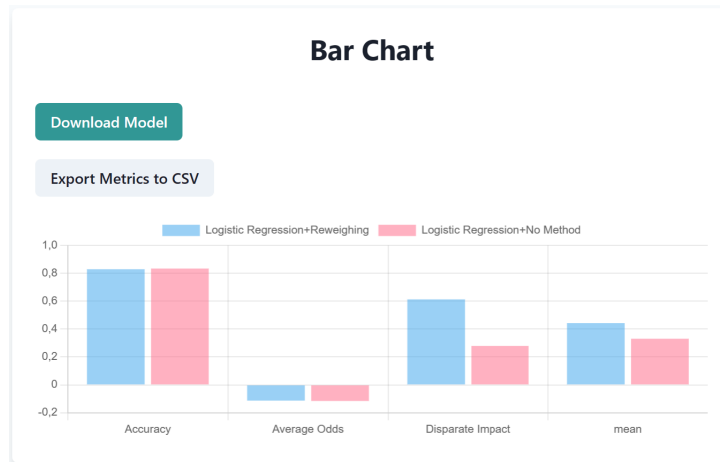
If the data scientist chooses to download the code, the generated experiment can be invoked directly through the Python interpreter using the command given in listing 2. Otherwise, it can be called through a REST API or any other interface such as a desktop application or a Scientific Workflow Management System like *KNIME* [62, 63]. This generality of our experimental workflow makes it very flexible and suitable for many use cases.

```
$ python main.py -d <DATASET PATH>
```

Listing 2: Experiment invocation

After the execution, the code returns the quality report in CSV format and the best ML model trained with the full input dataset and ready to be deployed. The ML model returned by the experiment is saved as a *pickle* file [64]. We have chosen this format since it is a standard format to store serialized objects in Python and can be easily imported into other scripts.

Instead, if the data scientist executes the experiment on the server, the results are shown on a dedicated page. Figure 9 shows the different elements of the results page. In this particular example, we used MANILA to evaluate the fairness and effectiveness of a *Logistic Regression* model alone and a *Logistic Regression* model with the application of the *Reweighting* preprocessing method[46]. We used the *Accuracy* metric to evaluate the effectiveness and the *Average Odds*[58] and *Disparate Impact*[47] metrics to evaluate the fairness of the two settings. Finally, we adopted the *Statistical Mean* as the aggregation function. The result page shows the metrics both in a bar chart and in a tabular way. Figure 9a shows the bar chart along with two buttons to download the fully trained best ML model in *pickle* format and the



(a) Metrics bar chart

Raw results

FAIRNESS METHOD	MACHINE LEARNING MODEL	ACCURACY	AVERAGE ODDS	DISPARATE IMPACT	MEAN
Reweighing	Logistic Regression	0.83	-0.11	0.61	0.44
No Method	Logistic Regression	0.83	-0.12	0.28	0.33

(b) Raw results

How to read the metrics

- Classification Metrics**
 These metrics are used to evaluate the performance of a classification and have an optimal value equal to **one**.
 The only exception is **Zero One Loss** which has an optimal value equal to **zero**.
- Regression Metrics**
 These metrics measure the error of the predictions of the model and have an optimal value equal to **zero**.
- Fairness Metrics**
 These metrics measure the fairness of the predictions of the model and have an optimal value equal to **zero**.
 The only exception is **Disparate Impact** which has an optimal value equal to **one**.

(c) Metrics description

Figure 9: MANILA result page

computed metrics as a CSV file, respectively. From the bar chart, it can be seen how the Logistic Regression plus Reweighing combination (the blue bar in figure 9a) performs better because it achieves a higher value of Disparate Impact (the closer this metric is to one, the more fair is the model) and a higher value of Statistical Mean. The same results are shown in a tabular format as displayed in figure 9b. In this case, the best combination is highlighted in green in the table. Finally, figure 9c shows a short description of how to read and interpret the different metrics.

5. MANILA Empirical Evaluation

In this section, we describe the evaluation we performed on MANILA. To this aim, we formulate the following research questions (RQ):

RQ₁ *Can MANILA effectively assist in conducting real-world fairness evaluations?*

RQ₂ *To what extent the results returned by MANILA are in line with baselines?*

RQ₃ *How much is MANILA perceived as useful and usable?*

RQ₄ *What are the main points of improvement of MANILA for its usability?*

To tackle RQ₁ and RQ₂, we evaluate MANILA both in terms of *expressiveness* (RQ₁) and *correctness* (RQ₂) by reproducing a set of real-world fairness evaluations taken from the literature. In particular, the *expressiveness* is assessed by proving that MANILA is actually able to replicate the selected fairness evaluations. In contrast, *correctness* is assessed by showing that the results obtained with the experiments generated by MANILA are comparable to the original ones.

To answer RQ₃ and RQ₄, we conduct a user evaluation of MANILA by involving students in Computer Science and Applied Data Science master’s degrees. The user evaluation is performed as a controlled experiment where we asked the participants to perform a simple two-step experiment focused on assessing the fairness and effectiveness of different ML and fairness method combinations. The first step required to make the experiment using Python while the second using MANILA. After the experiment, we asked the participants to answer a questionnaire. The answers to this survey highlighted

both the strengths and issues of our tool. The issues will be considered in the future to develop a new improved release of MANILA.

In Section 5.1 we depict the expressiveness and correctness evaluation performed to answer the RQ₁ and RQ₂. Then we describe and discuss the user evaluation conducted to answer RQ₃ and RQ₄ in Section 5.2.

5.1. Expressiveness and Correctness Evaluation of MANILA

To answer RQ₁ and RQ₂, we replicate with MANILA the experimental evaluation performed on our previous paper [52]. In that paper, we evaluated the fairness and effectiveness of several ML settings using different datasets. In this evaluation, we replicate three specific experiments (i.e.,

Table 2: Replicated experiments

Experiment	ML Settings	Datasets	Metrics
1	LogReg LogReg + EG LogReg + Grid LogReg + DEMV	CMC Crime Drug Law Park Wine	SP EO DI ZO Loss Acc H-Mean
2	Gradient Gradient + EG Gradient + Grid Gradient + DEMV	CMC Law	SP EO DI ZO Loss Acc H-Mean
3	SVM SVM + EG SVM + Grid SVM + DEMV	CMC Law	SP EO DI ZO Loss Acc H-Mean

the ones shown in Tables 16, 18 and 19 of [52]), which are synthesized in table 2. We have chosen to replicate these experiments among all the ones conducted in [52] because they provide the highest combination of ML methods, fairness methods and metrics. In particular, the first replicated experiment is the evaluation of four different ML settings (i.e., *Logistic Regression*

(*LogReg*) alone, *Logistic Regression* plus *Exponentiated Gradient (EG)* [48], *Logistic Regression* plus *Grid Search (GRID)* [48], and *Logistic Regression* plus *DEMV* [52]) on six different datasets (i.e., *Contraceptive Method Choice (CMC)* [65], *Communities and Crime (Crime)* [66], *Drug Usage (Drug)* [67], *Law School Admission (Law)* [14], *Parkinson Telemonitoring (Park)* [68], and *Wine Quality (Wine)* [69]).

The other two replicated experiments involve the same fairness methods as the first one but employ two different ML classifiers (i.e., *Gradient Boosting (Gradient)* and *Support Vector Machines (SVM)*, respectively). In this case, the evaluations are applied only to the *CMC* and *Law* datasets. In all experiments, we consider two sensitive variables for each dataset. The metrics involved are: *Statistical Parity (SP)* [57], *Equalized Odds (EO)* [58], *Zero One Loss (ZO Loss)* [70], *Disparate Impact (DI)* [47], *Accuracy (Acc)* [65], and *Harmonic Mean (H-Mean)* [71] as aggregation function.

In particular, the *expressiveness* is evaluated by assessing if MANILA is able to reproduce the described experiments correctly. The *correctness* is instead evaluated by assessing if the results of the experiments generated by MANILA are close to the ones reported in [52]. In particular, the results of the generated experiments should be within the standard deviation range of the results reported in [52], and there should not be a statistically significant difference between the results.

5.1.1. Expressiveness Evaluation

Concerning the *expressiveness* of MANILA, we were able to correctly reproduce all the experiments reported in table 2⁴. In particular, following the steps described in Sections ?? and ??, we first created 10 configuration files (one for each experiment and dataset) from the graphical interface of MANILA. Next, we generated the corresponding implementation code by invoking the generator module. Finally, we ran the experiments to obtain the results. Being a low-code platform, MANILA does not require to write any line of code to implement the given experiments. In contrast, the original experiments required almost 200 lines of code, as seen in the repository linked in the original paper⁵.

⁴The full replication package of the experiment is available at the following link <https://bit.ly/3ZR7f00>

⁵The original code of the reproduced experiment is available here <https://bit.ly/3t1WGvo>

Answer to RQ₁: MANILA presents a high level of *expressiveness* that allows it to implement real-world fairness evaluations involving real-world datasets and to replicate previous experiments.

5.1.2. Correctness Evaluation

Concerning the *correctness* of the generated experiments, table 3 reports, for each metric of each experiment, the *p-values* of the ANOVA statistical test performed between the results of the original experiments and the ones obtained by executing the code generated by MANILA. The ANOVA test is a statistical test to verify if the values of two groups are statistically different. In particular, if the *p-value* < 0.05 , then it means that the values of the groups are statistically different, otherwise their difference can be considered not statistically significant [72].

Table 3: *p-values* of the ANOVA tests for each experiment

	SP	EO	ZO Loss	DI	Acc	H-Mean
Exp 1	0.913527	0.192048	0.399797	0.528007	0.267832	0.925962
Exp 2	0.712197	0.620129	0.94997	0.804867	0.364792	0.710014
Exp 3	0.918328	0.524131	0.084226	0.90656	0.882236	0.929372

Concerning our evaluation, it can be seen from the results in table 3 that all the *p-values* are > 0.05 , meaning that all the metrics obtained by running the code generated by MANILA are not statistically different from the original ones. In addition, figure 10 reports a comparison of the aggregated h-means of the three experiments⁶. As can be noticed, on average, the results of the three experiments generated by MANILA are very close to the original ones and are within the standard deviation range.

Answer to RQ₂: The *correctness* of the code generated by MANILA allows to correctly replicate baseline fairness evaluations by obtaining results that are statistically equal to the original ones.

5.2. User Evaluation of MANILA

For the user evaluation of MANILA, we followed the guidelines of Runeson *et al.* [73]. The purpose of the evaluation was to assess the *usefulness*,

⁶We visualize only the h-mean because, being an aggregated value, it can be considered as a synthesis of the other selected metrics

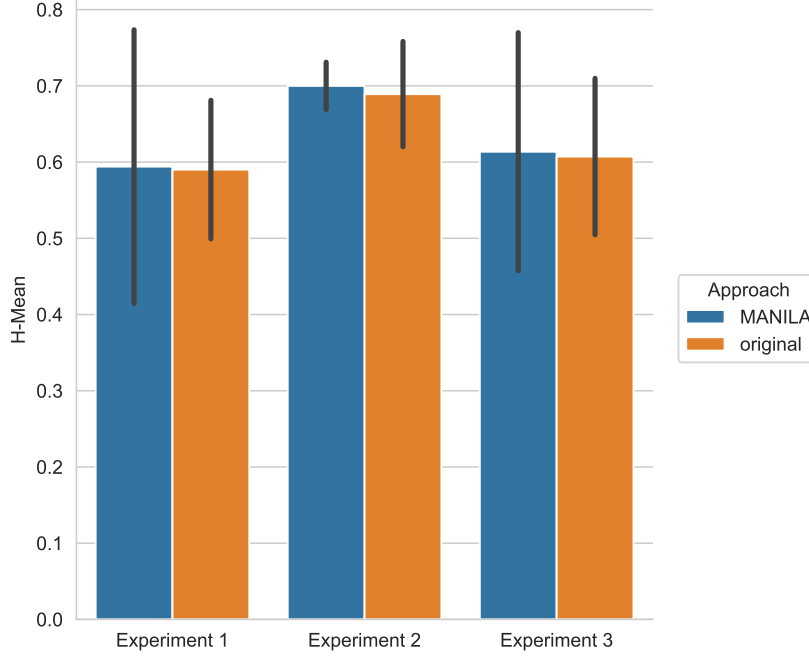


Figure 10: Aggregated H-Means of the original experiments and MANILA’s ones

usability and point of improvement of MANILA.

In the following, we first report on the process considered to execute the user evaluation, and then we discuss the obtained results.

5.2.1. User Evaluation Description

The evaluation involved 15 students of Computer Science and Applied Data Science master’s degrees in a half-day laboratory. First, we introduced the students to the topic of Bias and Fairness in ML and presented the main architecture of MANILA. Next, we asked them to perform a short fairness evaluation experiment. The experiment consisted in evaluating the fairness and effectiveness of three different ML settings (namely, *Logistic Regression* alone, *Logistic Regression* plus *Reweighting* fairness method[46], and *Logistic Regression* plus *DEMV* fairness method[52]) on the *Adult Income* dataset[74]. The metrics used to evaluate the settings were *Disparate Impact*[47], *Accuracy*[75] and, as aggregation function, *Harmonic Mean*[71].

First, the students had to perform the experiment in Python through a

Google Colab notebook⁷. The notebook was pre-filled with some instructions, and they had to complete the missing code. They had 50 minutes to complete this task. Next, the students had to perform the same experiment, but this time using MANILA. In particular, we provided the students with a document describing all the steps to set up MANILA locally and perform the given evaluation⁸. Like the previous task, the students had 50 minutes to perform it. Finally, the students were asked to fill a survey. The survey was structured into three main sections:

- **General information:** this section contains general information about the involved evaluator like age, gender, degree, area of expertise (i.e., Software Engineering, Data Analytics, Machine Learning and AI, or Software Development), level of knowledge of ML, level of knowledge of the classification task in ML, and level of knowledge of Python. The answers to these questions have been used to characterise the population;
- **Usefulness:** this section aims at assessing the level of usefulness of MANILA and includes questions about the level of knowledge of the Bias and Fairness topic in ML, the level of relevance of the Bias and Fairness problem for the student, the extent to which the student has completed the evaluation in Python, the extent to which the student has completed the evaluation in MANILA, the number of errors (in average) the student had during the evaluation in Python, and the number of errors (in average) the student had during the evaluation in MANILA;
- **Usability:** this section aims at assessing the level of usability of MANILA and includes questions about the level of easiness of the evaluation in Python, the level of easiness of the evaluation in MANILA, and a final question about which evaluation was overall the easiest. This section contains also two open questions on how to improve the overall usability and the result's presentation in MANILA. The first set of questions are used to assess the usability of MANILA, while the last two open questions are used to discuss the improvements of MANILA.

⁷The notebook is available here: <https://bit.ly/3QXqiUT>

⁸The document is available here: <https://bit.ly/3QSeLGg>

5.2.2. Evaluation Results

In the following, we present the results of our user evaluation, discussing each section of the survey⁹ and answering the RQ₃ and RQ₄. We organize such discussion into parts presenting the population characterisation, usefulness and usability of MANILA (used to answer the RQ₃), and possible improvements suggested by the respondents (used to answer RQ₄).

Population characterisation. As already mentioned in Section 5.2.1, the population of our study was composed of first and second-year computer science master’s degree students. We chose this sample because we wanted to evaluate MANILA from the perspective of people already familiar with the basic concepts of ML and data science but not fully experts in the field of fairness.

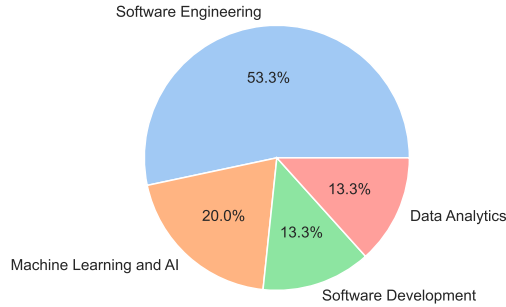


Figure 11: Area of expertise

As shown in figure 11, 53.3% of the population specify Software Engineering as their main area of expertise. At the same time, 20.0% declare Machine Learning and AI as their main area of expertise. At the same time, as reported in figure 12, only 6.67% of the population specified a high level of knowledge of ML, while 53.3% has a medium-high knowledge of ML and AI. The exact distribution of responses of figure 12 also holds for the question about the level of knowledge of the classification task in ML. Finally, as reported in figure 13, 46.6% of the population reported a medium-high knowledge of Python, while 26.67% reported a high level of expertise. Finally, as shown in figure 14a, although most of the respondents reported a

⁹The full results are available here <https://bit.ly/46p8juL>

medium-high level of knowledge of ML learning (figure 12), 66.6% of the population have low knowledge of the problem of Bias and Fairness in ML.

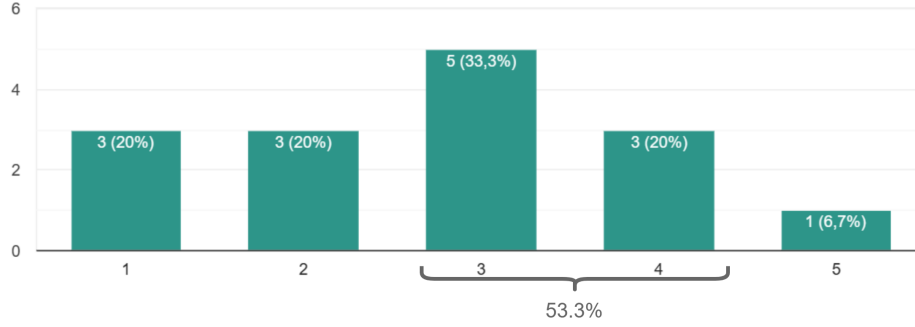


Figure 12: How well do you know ML

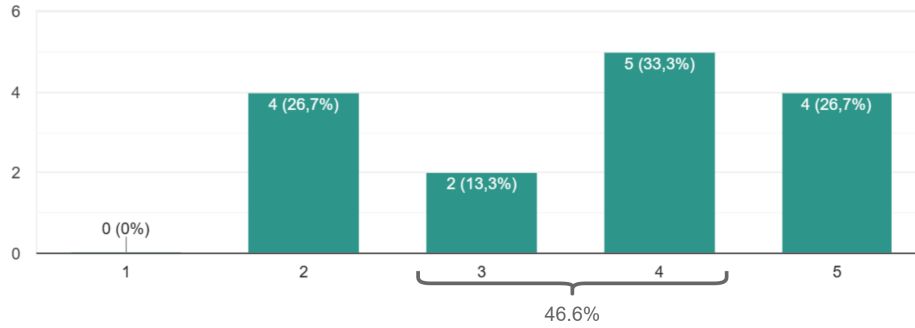
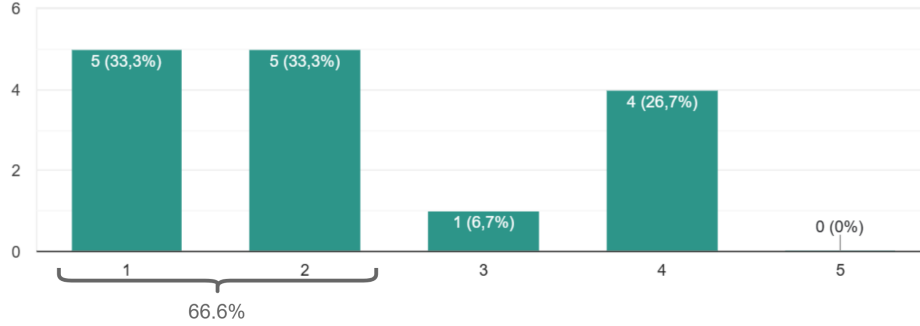


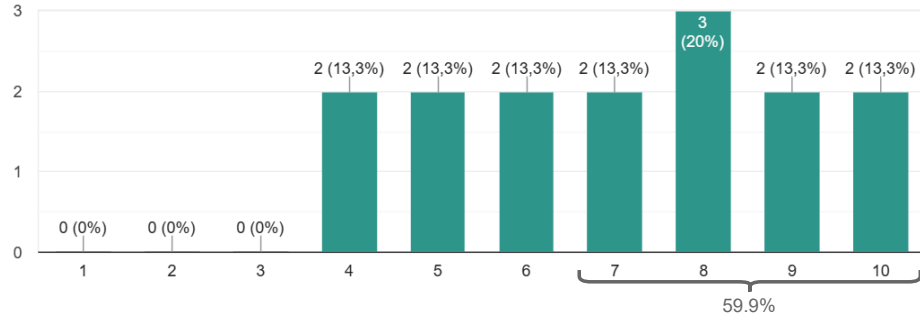
Figure 13: How well do you know Python

Conclusions: The considered sample of evaluators is quite diverse and well fits the user profiles we aim to target with MANILA, which can be expert data scientists and programmers, but also not experts of fairness.

Usefulness. Concerning the usefulness of MANILA, although, as reported in figure 14a, most of the population didn't know about the problem of bias and fairness in ML before the survey (with 66.6% of the people having low knowledge of the topic), the respondents believe that this problem is relevant, with 59.9% of the people giving a medium-high relevance to this topic as reported in figure 14b. Hence, addressing this topic has been seen as relevant by the respondents.



(a) How do you know the problem of Bias and Fairness in ML?

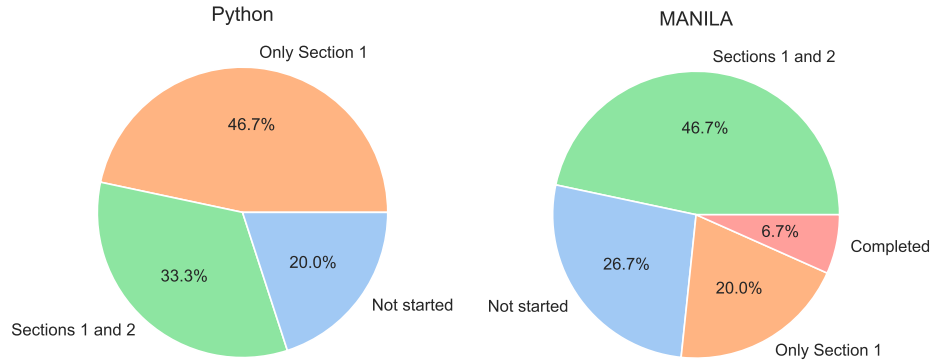


(b) How relevant do you think the problem of Bias and Fairness in Machine Learning is?

Figure 14: Importance of Bias and Fairness

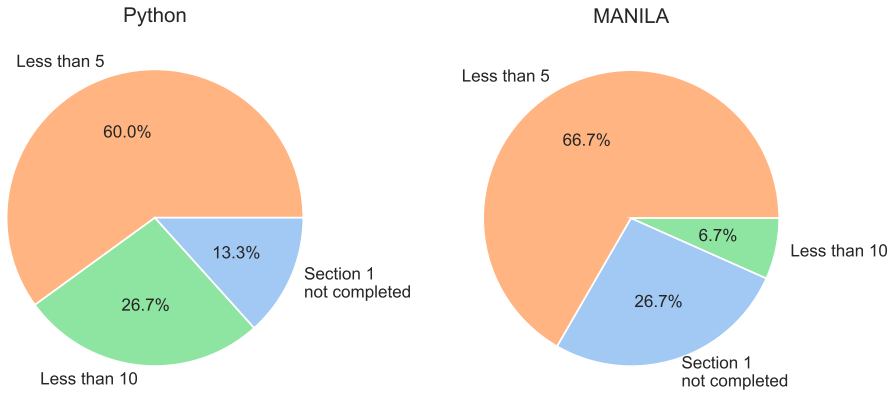
Figure 15 reports the comparison between the level of completion of the evaluation conducted in Python (figure 15a) and in the assessment conducted in MANILA (figure 15b). It is interesting to note how 46.7% of the population has completed at least sections 1 and 2 of the fairness evaluation in MANILA against 33.3% in case of Python. In addition, we note how 6.7% of the population has fully completed the evaluation in MANILA (as shown in figure 15b) while no one has completed the evaluation in Python (as shown in figure 15a). Finally, we also note how a higher portion of the population has not started the evaluation in MANILA (26.7% against 20% as reported in figures 15b and 15a, respectively), probably due to setup difficulties (as discussed later).

Figure 16 reports the comparison between the average number of execution errors obtained in the two evaluations. First of all, we note how the percentage of people in figure 16a reporting that have not completed section



- (a) To what extent have you completed the fairness evaluation in Python in the given time?
 (b) To what extent have you completed the fairness evaluation in MANILA in the given time?

Figure 15: Completion comparison



- (a) How many execution errors (in average) did you have during the fairness evaluation in Python?
 (b) How many execution errors (in average) did you have during the fairness evaluation in MANILA?

Figure 16: Errors comparison

1 of the experiment is different from the percentage of people in figure 15a reporting that have not started the experiment. This difference is explained by the fact that some people have started section 1 of the experiment but then got errors and have not completed it (as can be seen in the full list of responses). Instead, people who started the evaluation in MANILA have at

least completed section 1 of the evaluation (since the percentage of people reporting in figure 16b that have not started section 1 is the same percentage of people reporting in figure 15b that has not begun the evaluation). Moreover, we notice how MANILA leads to fewer execution errors than Python, with 66.7% having less than five errors and 6.7% reporting less than ten errors in MANILA, compared to 60% of people having less than five errors and 26.7% having less than ten errors in Python as shown in figures 16b and 16a, respectively.

Conclusions: From these responses, we can conclude how the respondents see a tool like MANILA as useful because it addresses a relevant topic. In addition, we note how 53.4% of the respondents completed at least sections 1 and 2 of the evaluation in MANILA, in contrast with the 33.3% in Python. Moreover, we also notice how MANILA leads to fewer execution errors than writing the experiment in Python. These results highlight how MANILA is a valuable tool to perform fairness evaluations.

Usability. Concerning usability, figure 17 reports the comparison between the easiness of the two evaluations. As can be seen, on average, the evaluation in MANILA is seen as easier than the evaluation in Python, with half of the population reporting a level of easiness medium-high for MANILA as reported in figure 17b. In contrast, half of the people said that the fairness evaluation in Python was medium-hard, as reported in figure 17a.

Finally, figure 18 reports how 60% of the respondents reported MANILA as the most accessible evaluation.

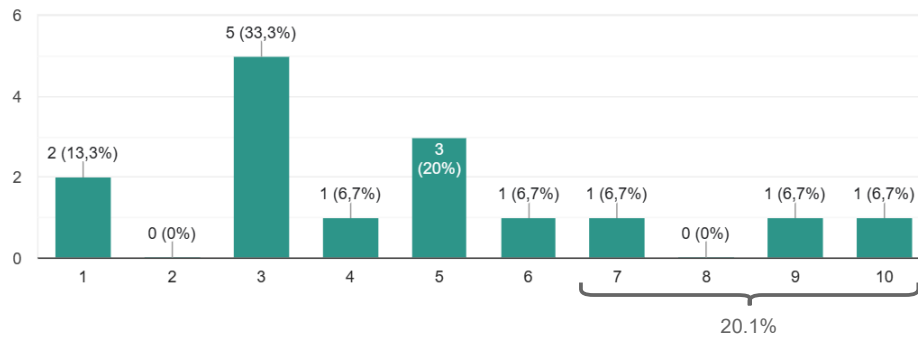
Conclusions: We can conclude that MANILA has been seen as a better solution for making fairness evaluations with respect to writing the evaluation in Python since it has been seen both as *useful* and *usable*.

Answer to RQ₃: MANILA has been seen, by a sample of evaluators with a medium-high knowledge of ML but no knowledge of fairness, as a *useful* and *usable* tool to perform fairness evaluations. In fact, it addresses a relevant topic and, compared to writing the fairness evaluation in Python, it is easier to use and leads to fewer execution errors.

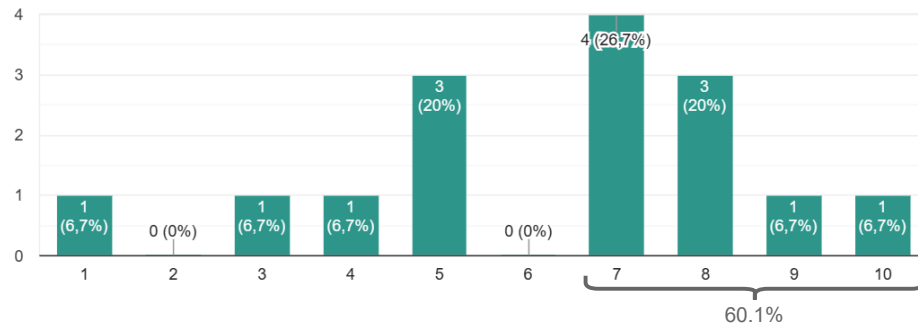
Improvements. Quote 1 shows the most meaningful responses to open questions that help us draw conclusions on how to improve MANILA.

From them, we can derive two main issues:

1. There are too many packages and libraries to download (answers 2, 4, and 5);



(a) How easy was the fairness evaluation in Python?



(b) How easy was the fairness evaluation in MANILA?

Figure 17: Easiness comparison

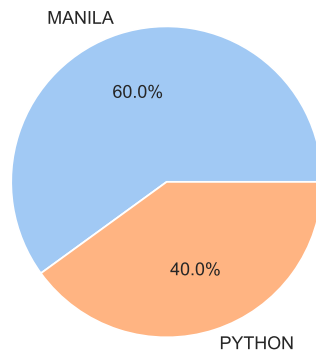


Figure 18: Which evaluation was overall the easiest?

1. *Automatize the installation of the execution of the flow avoiding typing all the Python and conda creation stuff*
2. *Less dependencies*
3. *Faster installation; begin installation before*
4. *If the file was given instead of creating it*
5. *The extraction of the files, they are too heavy*

Quote 1: Excerpt of the answers on how to improve MANILA

2. The overall setup process is too complex to perform (answers 1 and 3).

Following these answers, we believe that a web-based implementation of MANILA should overcome the highlighted limitations. In fact, differently from a desktop tool, a web application will completely remove all the download and setup processes and will also be a better solution in terms of performance since all the computational resources required to run the evaluation experiments will be moved to the server hosting the web app.

Answer to RQ₄ The respondents found difficult to manage all the dependencies and to overall setup MANILA. A web version of the tool should overcome these issues by removing all the download and setup steps and by moving all the computational resources required to run the experiments to the server.

6. Threats to Validity

This section discusses possible threats that can hamper the results of the performed evaluation.

Internal validity concerns factors that can influence the results of our evaluation. First, there could be other weaknesses in MANILA not highlighted by our user evaluation. In this respect, we argue that most of the participants described all the same shortcomings highlighted in this work (i.e., the difficulty in the download and installation of MANILA), meaning that this was an actual limitation of our approach. In any case, we are planning to conduct

a more extensive user evaluation after addressing the highlighted shortcomings of MANILA to find, and in case, improve other limitations. Secondly, although we referred to the most adopted fairness library (i.e., AIF360) for selecting the fairness methods and metrics to include in the tool, there could be other methods or metrics currently not included in MANILA. However, we have shown how MANILA has a level of *expressiveness* able to model different real-world use cases. In addition, MANILA can be easily extended to include other methods or metrics by adding a new entry in the feature model and its actual implementation in the code generation template.

External validity threats concern the generalizability of our approach. In this respect, there could be some real-world use cases that can not be implemented in MANILA. In particular, at this stage of work, we are not considering *individual* fairness definitions, but only *group* fairness definitions [10]. However, we have re-implemented a large set of use cases involving different ML methods, fairness methods, metrics and datasets to show how MANILA is able to manage different *group* fairness use cases, and, in our future works, we will extend MANILA to include also *individual* fairness definitions. Finally, the general quality evaluation workflow described in Section 3 has been derived from our previous experience in the quality-based development of ML Systems, but may not hold for all possible quality attributes. However, we have shown how this workflow is general enough to model fairness evaluation experiments as SPL and can be easily extended to cover additional quality attributes.

7. Conclusion and Future Work

In this paper, we have presented a novel low-code application, MANILA, to perform fairness and effectiveness evaluations. First, we have performed an extensive review of the literature presenting both papers that motivated our work and papers proposing related approaches. Concerning the latter, we have also highlighted the main differences between other approaches and ours. Next, we have presented a general workflow for the quality-based development of ML systems. Following the general workflow, we presented the architecture of MANILA describing its main components. We first presented the Extended Feature Model (ExtFM), which is the formalism at the basis of the tool. Then, we presented the code generation module and we described how the experimental evaluation is performed. Next, we presented a user evaluation of MANILA, conducted to highlight its main points of improve-

ment. From the evaluation, we have seen how the installation and set-up process of MANILA is too complex for the users. By developing a web-based version of MANILA, we should overcome these limitations. Finally, we performed an evaluation of the *expressiveness* and *correctness* of MANILA, by re-implementing a set of real-world fairness evaluations taken from the literature.

In future, we first plan to implement MANILA as a low-code web application, to overcome the highlighted limitations. Following, we plan to perform a user evaluation of the web-based version of MANILA by also involving data scientists working in industries, and researchers studying ML development and quality assessment (i.e., expert users). Next, we plan to extend MANILA by including metrics and methods related to individual fairness definitions and to other relevant quality attributes (e.g., explainability or privacy) and by implementing in the framework a trade-off analysis that combines the different quality attribute evaluations when required by means of Pareto-front functions. Finally, since MANILA supports the configuration of an experiment by running all possible combinations of the selected features, a limit of the proposed approach can be its complexity and the time needed to obtain the results. Such limitation is mitigated by the feature selection step, which demands the user to choose which features to include in the experiment. As future work, to enlarge the usage of MANILA, we will better study such aspects and provide guidelines to the users on how to mitigate such potential limitations.

Acknowledgements

This work is partially supported by the European Union - NextGenerationEU - National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) - Project: “SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics” - Prot. IR0000013 - Avviso n. 3264 del 28/12/2021

References

- [1] G. d’Aloisio, A. Di Marco, G. Stilo, Democratizing quality-based machine learning development through extended feature models, in: International Conference on Fundamental Approaches to Software Engineering, Springer Nature Switzerland Cham, 2023, pp. 88–110.

- [2] M. Rönkkö, J. Heikkinen, V. Kotovirta, V. Chandrasekar, Automated preprocessing of environmental data, *Future Generation Computer Systems* 45 (2015) 13–24. URL: <https://www.sciencedirect.com/science/article/pii/S0167739X14002040>. doi:<https://doi.org/10.1016/j.future.2014.10.011>.
- [3] P. M. Goncalves Jr., R. S. M. Barros, Automating data preprocessing with dmpml and kddml, in: 2011 10th IEEE/ACIS International Conference on Computer and Information Science, 2011, pp. 97–103. doi:10.1109/ICIS.2011.23.
- [4] X. He, K. Zhao, X. Chu, Auttml: A survey of the state-of-the-art, *Knowledge-Based Systems* 212 (2021) 106622. URL: <https://www.sciencedirect.com/science/article/pii/S0950705120307516>. doi:<https://doi.org/10.1016/j.knosys.2020.106622>.
- [5] H. Muccini, K. Vaidhyanathan, Software Architecture for ML-based Systems: What Exists and What Lies Ahead, in: 2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN), 2021, pp. 121–128. doi:10.1109/WAIN52551.2021.00026.
- [6] S. Martínez-Fernández, J. Bogner, X. Franch, M. Oriol, J. Siebert, A. Trendowicz, A. M. Vollmer, S. Wagner, Software Engineering for AI-Based Systems: A Survey, *ACM Transactions on Software Engineering and Methodology* 31 (2022) 37e:1–37e:59. URL: <https://doi.org/10.1145/3487043>. doi:10.1145/3487043.
- [7] J. Bosch, H. H. Olsson, I. Crnkovic, Engineering AI Systems: A Research Agenda, 2021. URL: <https://www.igi-global.com/chapter/engineering-ai-systems/www.igi-global.com/chapter/engineering-ai-systems/266130>. doi:10.4018/978-1-7998-5101-1.ch001, iISBN: 9781799851011 Pages: 1-19 Publisher: IGI Global.
- [8] U. Nations, THE 17 GOALS | Sustainable Development, 2022. URL: <https://sdgs.un.org/goals>.
- [9] H. B. Braiek, F. Khomh, On testing machine learning programs, *Journal of Systems and Software* 164 (2020) 110542. URL: <https://www>.

sciencedirect.com/science/article/pii/S0164121220300248.
doi:<https://doi.org/10.1016/j.jss.2020.110542>.

- [10] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, A. Galstyan, A Survey on Bias and Fairness in Machine Learning, *ACM Computing Surveys* 54 (2021) 1–35. URL: <https://dl.acm.org/doi/10.1145/3457607>. doi:10.1145/3457607.
- [11] European Commission, Regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts, 2021. URL: https://eur-lex.europa.eu/resource.html?uri=cellar:e0649735-a372-11eb-9585-01aa75ed71a1.0001.02/DOC_1&format=PDF.
- [12] A. Sahay, A. Indamutsa, D. Di Ruscio, A. Pierantonio, Supporting the understanding and comparison of low-code development platforms, in: 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE, 2020, pp. 171–178.
- [13] J. Angwin, J. Larson, S. Mattu, L. Kirchner, Machine bias, *ProPublica*, May 23 (2016) 139–159.
- [14] K. A. Austin, C. M. Christopher, D. Dickerson, Will I Pass the Bar Exam: Predicting Student Success Using LSAT Scores and Law School Performance, *Hofstra l. rev.* 45 (2016) 753. Publisher: HeinOnline.
- [15] G. Giray, A software engineering perspective on engineering machine learning systems: State of the art and challenges, *Journal of Systems and Software* (2021) 111031.
- [16] E. de Souza Nascimento, I. Ahmed, E. Oliveira, M. P. Palheta, I. Steinmacher, T. Conte, Understanding development process of machine learning systems: Challenges and solutions, in: 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), IEEE, 2019, pp. 1–6.
- [17] H. Villamizar, T. Escovedo, M. Kalinowski, Requirements engineering for machine learning: A systematic mapping study, in: SEAA, 2021, pp. 29–36.

- [18] F. Kumeno, Software engineering challenges for machine learning applications: A literature review, *Intelligent Decision Technologies* 13 (2019) 463–476.
- [19] J. M. Zhang, M. Harman, L. Ma, Y. Liu, Machine learning testing: Survey, landscapes and horizons, *IEEE Transactions on Software Engineering* (2020).
- [20] ISO, ISO/IEC 25010:2011, Technical Report, 2011. URL: <https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/03/57/35733.html>.
- [21] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, T. Zimmermann, Software Engineering for Machine Learning: A Case Study, in: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), IEEE, Montreal, QC, Canada, 2019, pp. 291–300. URL: <https://ieeexplore.ieee.org/document/8804457/>. doi:10.1109/ICSE-SEIP.2019.00042.
- [22] S. Studer, T. B. Bui, C. Drescher, A. Hanuschkin, L. Winkler, S. Peters, K.-R. Müller, Towards crisp-ml (q): a machine learning process model with quality assurance methodology, *Machine Learning and Knowledge Extraction* 3 (2021) 392–413.
- [23] F. Martínez-Plumed, L. Contreras-Ochando, C. Ferri, J. H. Orallo, M. Kull, N. Lachiche, M. J. R. Quintana, P. A. Flach, Crisp-dm twenty years later: From data mining processes to data science trajectories, *IEEE Transactions on Knowledge and Data Engineering* (2019).
- [24] K. Hamada, F. Ishikawa, S. Masuda, T. Myojin, Y. Nishi, H. Ogawa, T. Toku, S. Tokumoto, K. Tsuchiya, Y. Ujita, et al., Guidelines for quality assurance of machine learning-based artificial intelligence., in: SEKE, 2020, pp. 335–341.
- [25] S. Azimi, C. Pahl, A layered quality framework for machine learning-driven data and information models., in: ICEIS (1), 2020, pp. 579–587.
- [26] F. Ishikawa, Concepts in quality assessment for machine learning-from test data to arguments, in: International Conference on Conceptual Modeling, Springer, 2018, pp. 536–544.

- [27] J. Siebert, L. Joeckel, J. Heidrich, A. Trendowicz, K. Nakamichi, K. Ohashi, I. Namba, R. Yamamoto, M. Aoyama, Construction of a quality model for machine learning systems, *Software Quality Journal* (2021) 1–29.
- [28] R. Capilla, J. Bosch, P. Trinidad, A. Ruiz-Cortés, M. Hinchey, An overview of Dynamic Software Product Line architectures and techniques: Observations from research and industry, *Journal of Systems and Software* 91 (2014) 3–23. URL: <https://www.sciencedirect.com/science/article/pii/S0164121214000119>. doi:<https://doi.org/10.1016/j.jss.2013.12.038>.
- [29] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, A. S. Peterson, Feature-oriented domain analysis (FODA) feasibility study, Technical Report, Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 1990.
- [30] D. Benavides, S. Segura, A. Ruiz-Cortés, Automated analysis of feature models 20 years later: A literature review, *Information Systems* 35 (2010) 615–636. URL: <https://www.sciencedirect.com/science/article/pii/S0306437910000025>. doi:10.1016/j.is.2010.01.001.
- [31] C. Di Sipio, J. Di Rocco, D. Di Ruscio, D. P. T. Nguyen, A Low-Code Tool Supporting the Development of Recommender Systems, in: Fifteenth ACM Conference on Recommender Systems, ACM, Amsterdam Netherlands, 2021, pp. 741–744. URL: <https://dl.acm.org/doi/10.1145/3460231.3478885>. doi:10.1145/3460231.3478885.
- [32] Z. Chen, J. M. Zhang, M. Hort, F. Sarro, M. Harman, Fairness Testing: A Comprehensive Survey and Analysis of Trends, 2022. URL: <http://arxiv.org/abs/2207.10223>, arXiv:2207.10223 [cs].
- [33] B. Johnson, Y. Brun, Fairkit-learn: a fairness evaluation and comparison toolkit, in: Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings, ICSE ’22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 70–74. URL: <https://doi.org/10.1145/3510454.3516830>. doi:10.1145/3510454.3516830.

- [34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research* 12 (2011) 2825–2830.
- [35] M. S. A. Lee, J. Singh, The Landscape and Gaps in Open Source Fairness Toolkits, in: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 1–13. URL: <https://doi.org/10.1145/3411764.3445261>. doi:10.1145/3411764.3445261.
- [36] A. Yohannis, D. Kolovos, Towards model-based bias mitigation in machine learning, in: *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*, MODELS '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 143–153. URL: <https://doi.org/10.1145/3550355.3552401>. doi:10.1145/3550355.3552401.
- [37] S. Apel, D. Batory, C. Kästner, G. Saake, *Feature-oriented software product lines*, Springer, 2016.
- [38] J. A. Galindo, D. Benavides, P. Trinidad, A.-M. Gutiérrez-Fernández, A. Ruiz-Cortés, Automated analysis of feature models: Quo vadis?, *Computing* 101 (2019) 387–433. URL: <http://link.springer.com/10.1007/s00607-018-0646-1>. doi:10.1007/s00607-018-0646-1.
- [39] P. Refaeilzadeh, L. Tang, H. Liu, *Cross-Validation*, Springer New York, New York, NY, 2016, pp. 1–7. doi:10.1007/978-1-4899-7993-3_565-2.
- [40] V. Grossi, B. Rapisarda, F. Giannotti, D. Pedreschi, Data science at SoBigData: the European research infrastructure for social mining and big data analytics, *International Journal of Data Science and Analytics* 6 (2018) 205–216. URL: <https://doi.org/10.1007/s41060-018-0126-x>. doi:10.1007/s41060-018-0126-x.
- [41] M. Assante, L. Candela, D. Castelli, R. Cirillo, G. Coro, L. Frosini, L. Lelii, F. Mangiacrapa, P. Pagano, G. Panichi, et al., *Enacting open*

- science by d4science, *Future Generation Computer Systems* 101 (2019) 555–563.
- [42] T. Thüm, C. Kästner, F. Benduhn, J. Meinicke, G. Saake, T. Leich, Featureide: An extensible framework for feature-oriented software development, *Science of Computer Programming* 79 (2014) 70–85.
 - [43] S. Patro, K. K. Sahu, Normalization: A preprocessing stage, *arXiv preprint arXiv:1503.06462* (2015).
 - [44] M. Kearns, S. Neel, A. Roth, Z. S. Wu, An empirical study of rich subgroup fairness for machine learning, in: *Proceedings of the conference on fairness, accountability, and transparency*, 2019, pp. 100–109.
 - [45] L. E. Celis, L. Huang, V. Keswani, N. K. Vishnoi, Classification with fairness constraints: A meta-algorithm with provable guarantees, in: *Proceedings of the conference on fairness, accountability, and transparency*, 2019, pp. 319–328.
 - [46] F. Kamiran, T. Calders, Data preprocessing techniques for classification without discrimination, *Knowledge and Information Systems* 33 (2012) 1–33. URL: <http://link.springer.com/10.1007/s10115-011-0463-8>. doi:10.1007/s10115-011-0463-8.
 - [47] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, S. Venkatasubramanian, Certifying and Removing Disparate Impact, in: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, Sydney NSW Australia, 2015, pp. 259–268. URL: <https://dl.acm.org/doi/10.1145/2783258.2783311>. doi:10.1145/2783258.2783311.
 - [48] A. Agarwal, A. Beygelzimer, M. Dudik, J. Langford, H. Wallach, A Reductions Approach to Fair Classification, in: *Proceedings of the 35th International Conference on Machine Learning*, PMLR, 2018, pp. 60–69. URL: <https://proceedings.mlr.press/v80/agarwal18a.html>, iSSN: 2640-3498.
 - [49] P. Putzel, S. Lee, Blackbox Post-Processing for Multiclass Fairness, *arXiv:2201.04461 [cs]* (2022). URL: <http://arxiv.org/abs/2201.04461>, arXiv: 2201.04461.

- [50] R. K. Bellamy, K. Dey, M. Hind, S. C. Hoffman, S. Houde, K. Kannan, P. Lohia, J. Martino, S. Mehta, A. Mojsilović, et al., Ai fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias, IBM Journal of Research and Development 63 (2019) 4–1.
- [51] S. Bird, M. Dudík, R. Edgar, B. Horn, R. Lutz, V. Milan, M. Sameki, H. Wallach, K. Walker, Fairlearn: A toolkit for assessing and improving fairness in AI, Technical Report MSR-TR-2020-32, Microsoft, 2020. URL: <https://www.microsoft.com/en-us/research/publication/fairlearn-a-toolkit-for-assessing-and-improving-fairness-in-ai/>.
- [52] G. d’Aloisio, G. Stilo, A. Di Marco, A. D’Angelo, Enhancing fairness in classification tasks with multiple variables: A data-and model-agnostic approach, in: International Workshop on Algorithmic Bias in Search and Recommendation, Springer, 2022, pp. 117–129.
- [53] P. Refaeilzadeh, L. Tang, H. Liu, Cross-validation., Encyclopedia of database systems 5 (2009) 532–538.
- [54] P.-E. Danielsson, Euclidean distance mapping, Computer Graphics and image processing 14 (1980) 227–248.
- [55] A. Singh, A. Yadav, A. Rana, K-means with three different distance metrics, International Journal of Computer Applications 67 (2013).
- [56] G. J. McLachlan, Mahalanobis distance, Resonance 4 (1999) 20–26.
- [57] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, R. Zemel, Fairness through awareness, Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (2012) 214–226. URL: <https://doi.org/10.1145/2090236.2090255>. doi:10.1145/2090236.2090255.
- [58] M. Hardt, E. Price, E. Price, N. Srebro, Equality of Opportunity in Supervised Learning, in: Advances in Neural Information Processing Systems, volume 29, Curran Associates, Inc., 2016. URL: <https://proceedings.neurips.cc/paper/2016/hash/9d2682367c3935defcb1f9e247a97c0d-Abstract.html>.
- [59] R. Berk, H. Heidari, S. Jabbari, M. Kearns, A. Roth, Fairness in criminal justice risk assessments: The state of the art, <https://doi.org/10.1177/0049124118782533> 50 (2018) 3–44. URL:

<https://journals.sagepub.com/doi/10.1177/0049124118782533>.
doi:10.1177/0049124118782533, publisher: SAGE PublicationsSage
CA: Los Angeles, CA.

- [60] E. Lepeschkin, B. Surawicz, Characteristics of true-positive and false-positive results of electrocardiographs master two-step exercise tests, *New England Journal of Medicine* 258 (1958) 511–520.
- [61] PalletsProject, Jinja website, 2023. URL: <https://jinja.palletsprojects.com/>.
- [62] J. Liu, E. Pacitti, P. Valduriez, M. Mattoso, A survey of data-intensive scientific workflow management, *Journal of Grid Computing* 13 (2015) 457–493.
- [63] M. R. Berthold, N. Cebon, F. Dill, T. R. Gabriel, T. Kötter, T. Meinel, P. Ohl, K. Thiel, B. Wiswedel, Knime - the konstanz information miner: Version 2.0 and beyond, *SIGKDD Explor. Newsl.* 11 (2009) 26–31. URL: <https://doi-org.univaq.cla.s.cineca.it/10.1145/1656274.1656280>. doi:10.1145/1656274.1656280.
- [64] Pickle documentation, 2023. URL: <https://docs.python.org/3/library/pickle.html>.
- [65] T.-S. Lim, W.-Y. Loh, Y.-S. Shih, A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms, *Machine learning* 40 (2000) 203–228. Publisher: Springer.
- [66] M. Redmond, A. Baveja, A data-driven software tool for enabling cooperative information sharing among police departments, *European Journal of Operational Research* 141 (2002) 660–678. Publisher: Elsevier.
- [67] E. Fehrman, A. K. Muhammad, E. M. Mirkes, V. Egan, A. N. Gorban, The Five Factor Model of Personality and Evaluation of Drug Consumption Risk, in: F. Palumbo, A. Montanari, M. Vichi (Eds.), *Data Science, Studies in Classification, Data Analysis, and Knowledge Organization*, Springer International Publishing, Cham, 2017, pp. 231–242. doi:10.1007/978-3-319-55723-6_18.
- [68] A. Tsanas, M. Little, P. McSharry, L. Ramig, Accurate telemonitoring of Parkinson’s disease progression by non-invasive speech tests, *Nature*

- Precedings (2009) 1–1. URL: <https://www.nature.com/articles/npre.2009.3920.1>. doi:10.1038/npre.2009.3920.1, publisher: Nature Publishing Group.
- [69] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis, Modeling wine preferences by data mining from physicochemical properties, *Decision support systems* 47 (2009) 547–553. Publisher: Elsevier.
 - [70] P. Domingos, M. Pazzani, On the Optimality of the Simple Bayesian Classifier under Zero-One Loss, *Machine Learning* 29 (1997) 103–130. URL: <https://doi.org/10.1023/A:1007413511361>. doi:10.1023/A:1007413511361.
 - [71] W. F. Ferger, The nature and use of the harmonic mean, *Journal of the American Statistical Association* 26 (1931) 36 – 40. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-83055185035&doi=10.1080%2f01621459.1931.10503148&partnerID=40&md5=66d14fb6f6ec05b27941150ae41da25c>. doi:10.1080/01621459.1931.10503148, cited by: 35.
 - [72] L. St, S. Wold, et al., Analysis of variance (anova), *Chemometrics and intelligent laboratory systems* 6 (1989) 259–272.
 - [73] P. Runeson, M. Host, A. Rainer, B. Regnell, *Case study research in software engineering: Guidelines and examples*, John Wiley & Sons, 2012.
 - [74] C. A. Ratanamahatana, D. Gunopulos, Scaling up the naive Bayesian classifier: Using decision trees for feature selection (2002). Publisher: Citeseer.
 - [75] G. Rosenfield, K. Fitzpatrick-Lins, A coefficient of agreement as a measure of thematic classification accuracy., *Photogrammetric Engineering and Remote Sensing* 52 (1986) 223–227. URL: <http://pubs.er.usgs.gov/publication/70014667>.