A semi-autonomic solution for developing Machine Learning-based applications using Fairness metrics

Thales Eduardo Nazatto
tenazatto@gmail.com
University of Campinas
Campinas, São Paulo, Brazil

Cecília Mary Fischer Rubira
cmrubira@ic.unicamp.br
University of Campinas
Campinas, São Paulo, Brazil

Leonardo Montecchi
leonardo.montecchi@ntnu.no
Norwegian University of Science and Technology
Trondheim, Trøndelag, Norway

ABSTRACT

Machine Learning (ML) is increasingly used with big data to make faster and more assertive decisions, but initial metrics to evaluate algorithms have proven limited in measuring biases that reflect society in an unexpected way. To address this, new Fairness algorithms and metrics have been created to balance discriminated groups, but it increases the complexity for Data Scientists to develop the best models. This article focuses on Machine Learning solutions from a Software Engineering point of view, presenting a Workflow that uses the MAPE-K architecture to determine which algorithm has the best balance, without having to test many techniques. Several case studies were carried out to verify the feasibility of MAPE-K in solving problems, and if the solution can be easily extended to allow the use of more algorithms. This can help ML Specialists to do Software Engineering studies on applications with responsible use of data.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Computer systems organization** → **Self-organizing autonomic computing**; *Pipeline computing*; Maintainability and maintenance; • **Software and its engineering** → *Software product lines*; Software evolution.

KEYWORDS

Automated Machine Learning, AI Ethics, Social Responsibility in AI, Autonomic Loop, Fairness metrics, MAPE-K

ACM Reference Format:
Thales Eduardo Nazatto, Cecília Mary Fischer Rubira, and Leonardo Montecchi. 2023. A semi-autonomic solution for developing Machine Learning-based applications using Fairness metrics. In *Proceedings of 12th Latin-American Symposium on Dependable and Secure Computing (LADC 2023).* ACM, New York, NY, USA, 9 pages. https://doi.org/11.1111/1111111.1111111

1 INTRODUCTION

Artificial Intelligence (AI) and Machine Learning (ML) techniques have been used for a long time in the field of Computer Science. Areas such as robotics and games are great examples, given their need to automate behaviors that would be considered trivial for a human being. However, in recent years there has been a growth in the use of these technologies in traditional applications, mainly due to the large amount of data processed daily by companies and the amount of processing available at low costs. Different profiles can be drawn from these data and using AI solutions generate more assertive decision making in order to improve the user experience and correct problems. However, a common side effect in these solutions is the exposure of biases that, although seen as unintentional by developers due to the possibility of being an *outlier* in the trained model, reflect open prejudices of today's society. Biased data entries result in an algorithm that performs discriminations in its classification [9], and since the metrics used to measure the quality of final model are generally based on accuracy, precision and recall, social contexts and discrimination are not easily perceived by such metrics.

Due to this problem, it is possible to establish different metrics to determine how well the model is prepared for sensitive data [7], a term that is known as *Fairness*. With the evolution of academic research on this concept, new methods were defined to reduce the biases present in datasets, such as *Reweighing* [14], *Adversarial Debiasing* [24] and *Reject Option Classification* [15]. Consequently, there is an improvement in these metrics, but it may disfavor metrics that are already widely used as a guarantee of an effective model developed with Machine Learning techniques.

For this particular context, a method for developing Machine Learning applications was developed, which is handled by Data Scientists in a semi-autonomous way, focused on two main objectives:

- Make the creation of fair and reliable models by automating the choice of algorithms easier, due to increased complexity in choosing algorithms to be used and their execution in the correct steps of the process.
- Define a balance between metrics for evaluating good models with metrics for evaluating fair models.

This structure is divided into 5 modules: Data Engineering, ML Workflow, Autonomic Manager, Backend and Frontend. For Data Engineering, modifications were made to the German Credit Dataset [2] and Lendingclub Dataset [1] datasets to be suitable for the *workflow* implementation. For the workflow development, the *Pipes-and-Filters* architecture was used. For the Autonomic Manager, the MAPE-K [3] architecture was used to analyze a knowledge base and provide the best configuration background following predetermined rules. For the Frontend and the Backend, its creation was thought

LADC 2023, October 16-20 2023, La Paz, Bolívia

ACM ISBN 111-1-1111-1111-1…$0.00
https://doi.org/11.1111/1111111.1111111

of a web application template[1] for evaluation and testing in further studies.

2 BACKGROUND

2.1 Autonomic Computing

In 2001, Paul Horn introduced the concept of Autonomic Computing as a possible solution to the growing complexity of systems at the time, where it was predicted that they would become too large and complex for even the most qualified professionals to configure and maintain. This concept qualifies computing systems that can self-manage in relation to the high-level goals given by administrators and is derived from biology, given the great variety and hierarchy of autonomous systems existant in nature and society [18].

To Autonomic Computing works as intended, an Autonomic Element [4] is implemented, a software component that manages parts of the system based on a MAPE-K *loop* (*Monitor, Analyze, Plan, Execute, and Knowledge*). MAPE-K is a concept that constitutes a control *loop*, used to monitor and control one or more Managed Elements. A Managed Element can be hardware such as a printer, software such as a database, another Autonomous Element, or specific functions such as load balancing.

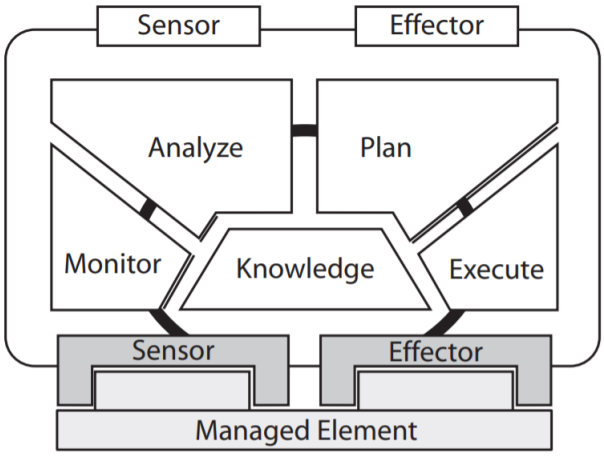

Figure 1: MAPE-K architecture diagram [4].

As illustrated in Figure 1, a MAPE-K control *loop* is split as follows:

- **Monitor:** Responsible for monitoring managed resources and collecting, aggregating and filtering data. Monitoring is done through one Sensor or more.
- **Analyze:** Analyzes the data reported by the monitor part. The analysis aims to understand what the current state of the system is and if there are measures to be taken.
- **Plan:** An action plan is prepared in the basis of the analysis' results. The plan is a series of measures that will move the system from its current state to a desired state.
- **Execute:** The plan is executed and controlled. One Effector or more perform the planned actions on the resource.
- **Knowledge:** The knowledge base is central and accessible by all parts of the *loop*. Separated from collected and analyzed data, it contains additional knowledge such as architectural models, goal models, policies and change plans.

2.2 Machine Learning

Machine Learning can be defined as "the practice of using algorithms to collect data, learn from it, and then make a determination or prediction about something in the world. Instead of doing code manually to complete a particular task, the machine is 'trained' using large amount of data and algorithms that give it the ability to learn how to perform tasks.

The method for training a machine are composed by the following steps:

- Data collection
- Data cleaning and refinement
- Data preparation and split into training and test sets
- Algorithm training and learning using the training set
- Metric evaluation using the test set

A Machine Learning algorithm can be classified into three basic categories:

- **Supervised learning**: Training is performed using labeled examples, such as an input in which the desired output is known. Through methods such as classification, regression, and *gradient boosting*, supervised learning uses patterns to predict the values of labels in additional unlabeled data. Supervised learning is commonly employed in applications where historical data predict likely future events.
- **Unsupervised Learning**: Used on data that does not have historical labels. The "right answer" is not reported to the system, the algorithm must figure out what is being shown. The goal is to explore the data and find some structure within it. Popular techniques include self-organizing maps, proximity mapping, *k-means*, and singular value decomposition. These algorithms are also used to segment text topics, recommend items, and identify outliers in data.
- **Reinforcement learning**: The algorithm discovers through trial-and-error testing which actions yield the greatest rewards. This type of learning has three main components: the agent (the learner or decision maker), the environment (everything the agent interacts with) and actions (what the agent can do). The objective is for the agent to choose actions that maximize the expected reward in a given period of time. The agent will reach the goal much faster if it follows a good policy, so the focus of reinforcement learning is to discover the best policy.

2.3 *Fairness* in Machine Learning

It is possible to describe the concept of *Fairness* in the context of supervised learning, where a model f can predict a set of results y from a set of *features* x, avoiding unfair discrimination against a protected attribute a. It is allowed, but not required, for a to be a component of x [7]. In other words, a fair ML model is one where the correlation of its result is low in relation to input data considered

[1]Repository containing the codes for this project: https://anonymous.4open.science/r/FAIR-2EFD

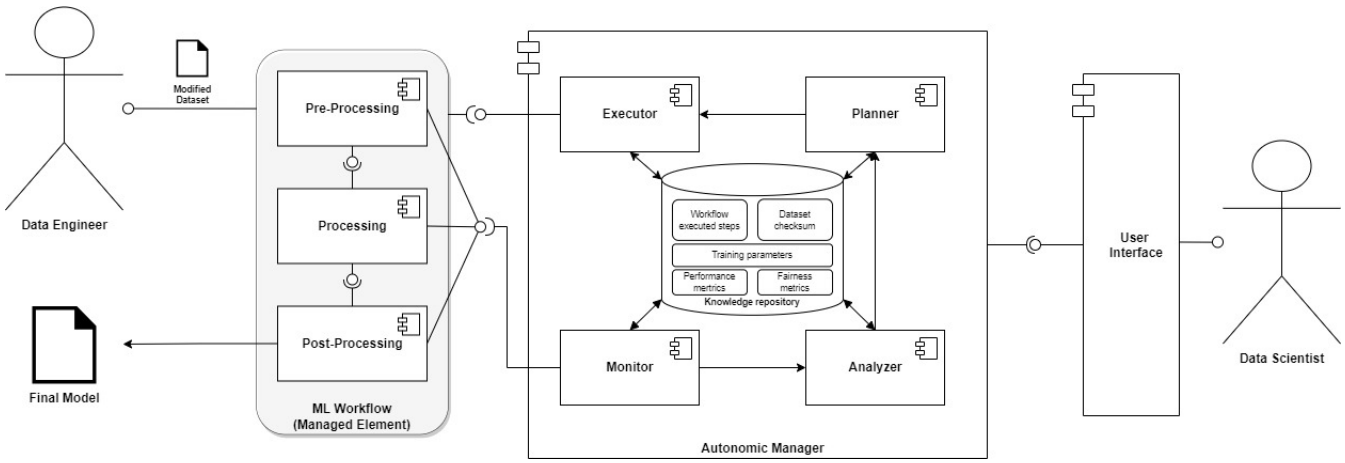

Figure 2: Architecture components of the proposed solution

sensitive to discrimination. Machine Learning algorithms are not able to differentiate social contexts, where a more efficient result according to the available data can amplify social inequalities and make decisions unfairly [19]. These sensitive data, such as race, sex, age and height, are considered protected attributes, which need to be classified and processed before the execution of a Machine Learning algorithm, which will determine how the algorithm will behave and, consequently, will affect the metrics [20]. The groups of data from these protected attributes are considered protected groups, which can be divided into two groups: the privileged group, which has advantages in the problem's context, and the non-privileged group, which has disadvantages in the problem's context and, therefore, subject to discrimination.

Fairness metrics differ from the metrics used for model evaluation, which have the purpose of checking whether a model has reliable predictions or not. While the more traditional metrics evaluate the performance of the model, data as a whole and general results, Fairness metrics evaluate whether the general results are also reflected in specific groups, to verify that there is no disparity or discrimination in the proposed results. Examples of metrics used for this are Statistical Parity Difference, or discrimination [23], Equal Opportunity Difference [8], Average Odds Difference [8], Disparate Impact [8] and Theil Index [22]. Almost all of these metrics needs a value close to 0 to be considered fair. The only exception is Disparate Impact which needs a value close to 1 due to the fact this metric is the only one which is calculated as a ratio.

To balance these metrics in a ML model, new algorithms were created to make the learning process more prepared to deal with biased data, which can be classified into three categories:

- **Pre-processing:** These algorithms try to remove biased data by transforming it. Some examples are *Disparate Impact Remover* [12], *Learning Fair Representations* [23], *Reweighing* [14] and *Optimized Preprocessing* [10]
- **Processing:** These algorithms try to mitigate discrimination during the training process. Some examples are *Adversarial Debiasing* [24], *Exponentiated Gradient Reduction* [5], *Grid Search Reduction* [6], *Meta Fair Classifier* [11], *Prejudice Remover* [16] and *Rich Subgroup Fairness* [17]
- **Post-processing:** These algorithms change the predictions made by an already trained model to make it fair. Some examples are *Equalized Odds* [13], *Calibrated Equalized Odds* [21] and *Reject Option Classification* [15].

3 THE PROPOSED SOLUTION

3.1 Architecture and Implementation

The solution architecture is divided into 5 modules:

- **Data Engineering:** Module with the objective of executing data transformation and cleaning processes.
- **ML Workflow:** Module that executes a Pipeline of an automated ML application, with stages of data preparation (Pre-processing), training (Processing) and result evaluation (Post-processing) of the final model generation.
- **Autonomic Manager:** Module containing a MAPE-K *loop* that controls the Workflow as a Managed Element to automate all its steps, in order to avoid manual executions depending on the algorithm and/or dataset used on the stages.
- **User Interface (Frontend):** Module which goal is to provide a simpler and more intuitive user experience for the Data Scientist in order to configure and start the Autonomic Manager.
- **Backend:** Module created with the objective of connecting the Frontend to the Autonomic Manager.

The integration between these modules is illustrated in Figure 2. The Data Engineering module is used by the Data Engineer, who would return a dataset with the necessary treatments for execution within the supervised learning workflow. The Frontend module would be used by the Data Scientist, who would determine the settings needed to choose the configuration background depending on the problem's context.

The process that takes place, from the User Interface interaction, passing through the execution of workflow and finishing at obtaining the results, can be visualized in the Sequence Diagram present in Figure 3. In this process, while the Data Engineer handles the dataset with transformation and cleaning processes presented in the Data

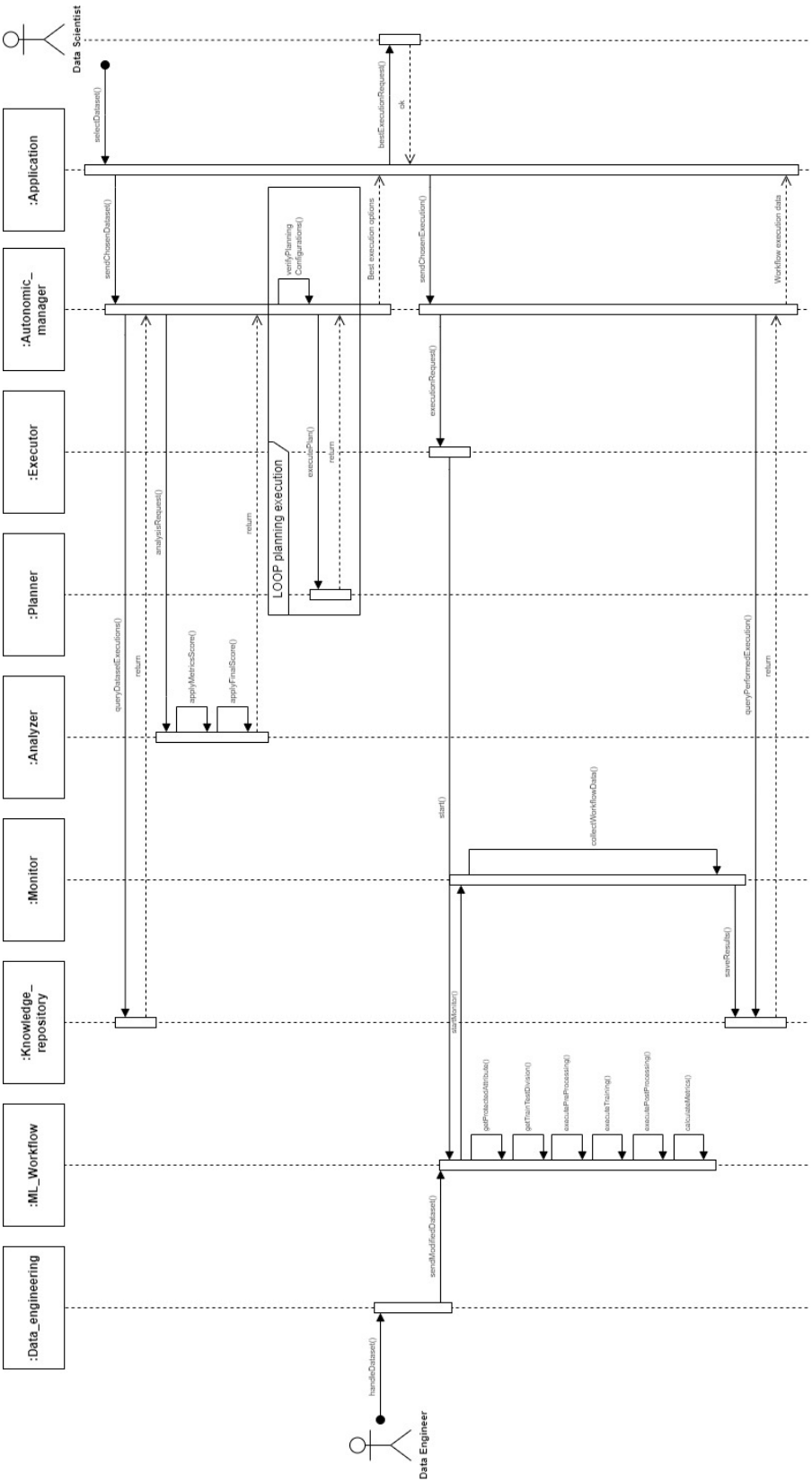

Figure 3: System's sequence diagram

Engineering module and send the modified datasets to integrate and use into ML Workflow, the Data Scientist, via the application's User Interface, choose the dataset and demand an Autonomic Manager execution. The Autonomic Manager queries all the executions made by the ML Workflow that used this dataset and are saved into the Knowledge Repository, do the analysis making some calculations that will be detailed in the next sections, and do a loop verifying the configuration of some implemented execution plans and executing them, returning some best execution options and wait the Data Scientist evaluation to choose one option to request a new ML Workflow execution. After the request, the ML Workflow starts the Monitor to collect data during its execution, get the chosen dataset and protected attribute, split into training and test sets, execute the chosen algorithms in the Pre-processing, Processing and Post-processing steps, and calculate the final model metrics. The Monitor save all collected data into the Knowledge Repository, and the Autonomic Manager queries this performed execution data to show to the Data Scientist.

3.2 Monitor and Analysis Component behavior

To guarantee the workflow autonomy, the Monitor component collects the data obtained during a execution (metrics, parameters and executed steps) and organizes them for the Analyzer to make the data analysis process. In this process, a calculation of weights based on specific metrics is performed to consolidate all metrics and simplify planning strategies. Metrics are divided into two groups (Performance metrics and Fairness metrics), and within this group you can place as many metrics as necessary, as long as the context of each group is respected. Each group is assigned different weights, and each metric in that group is also assigned different weights. The reason these weights exist in the calculation is to give a measure of the problem's context, according to a prior analysis by the Data Scientist and Domain Specialist.

First, it is necessary to standardize the Fairness metrics m_{F_i} to m'_{F_i} in an interval from 0 to 1, as presented in Equation 1. This guarantees the result consistency and the correct application of weights, avoiding distortions in the calculation. As Fairness metrics can involve both the ratio between two different values, like *Disparate Impact*, and the difference between two different values, like *Equal Opportunity Difference*, different calculations are required to get that range. As the evaluated Performance metrics have the same range from 0 to 1, there is no need for additional treatment for the metrics m_{P_i}.

Then, each standardized metric is multiplied by its corresponding weights w_{P_i} and w_{F_i}, and a weighted average is taken within the group to assign a score S_P to the Performance metrics group and S_F for the Fairness metrics group, as shown in equation 2. To make the score evaluation easier, the scores are multiplied by a factor $X = 1000$ to make the score range is 0 to 1000 and the value is rounded. After such scores are obtained, the overall score S is calculated by multiplying them by their corresponding weights w_P and w_F and arranging the weighted average as shown in equation 3.

$$m'_{F_i} = \begin{cases} 1 - |m_{F_i}| & \text{if } m_{F_i} \text{ is subtraction-calculated} \\ & \text{and } -1 < m_{F_i} < 1 \\ 0 & \text{if } m_{F_i} \text{ is subtraction-calculated} \\ & \text{and } m_{F_i} >= 1 \text{ or } m_{F_i} <= -1 \\ 1 - |\frac{1}{m_{F_i}} - 1| & \text{if } m_{F_i} \text{ is division-calculated and } m_{F_i} > 1 \\ 1 - |m_{F_i} - 1| & \text{if } m_{F_i} \text{ is division-calculated and } m_{F_i} <= 1 \\ m_{F_i} & \text{in other cases} \end{cases} \tag{1}$$

$$S_F = \lfloor X \times \frac{\sum_{i=1}^{n_F} w_{m'_{F_i}} \times m'_{F_i}}{\sum_{i=1}^{n} w_{m'_{F_i}}} \rceil \tag{2}$$

$$S_P = \lfloor X \times \frac{\sum_{i=1}^{n_P} w_{m_{P_i}} \times m_{P_i}}{\sum_{i=1}^{n} w_{m_{P_i}}} \rceil$$

$$S = \frac{w_F \times S_F + w_P \times S_P}{w_F + w_P} \tag{3}$$

To do these calculations, previous executions in the workflow are needed as a prerequisite, so the necessary metrics could be recorded and then the best combinations are determined through the score.

3.3 Planner Component

To determine the configuration background, some strategies were implemented in the Planner component:

- **Algorithm filtering**: Some algorithms may not perform well on Workflow or their models may have metrics that need a better analysis by the Data Scientist to be considered reliable. To prevent this from happening, this strategy filters reliable algorithm combinations before selecting the ideal models.
- **Result range**: Executions with unreliable metric results can distort the score obtained in the Analyzer. To ease these distortions, a minimum and maximum score range were created to determine scores that can be considered reliable for evaluation.

After evaluating these strategies, the Planner component will return the top 5 scores for the Data Scientist to choose, and will ask the Executor to run the Workflow again after choosing.

4 SOLUTION EVALUATION

Three Case Studies were executed to check our solution feasibility, future maintenance and the Autonomic Manager feasibility in choosing the best options in different contexts. In all of them, the objective was to obtain a credit rating (good or bad), through a series of *features* and using different datasets. In the case of system changes, the lines of code performed in each change were also counted, to define whether such changes are simple to make.

The execution to choose the best options was performed with 3 different weightings in the overall score:

- 50% for Performance metrics and 50% for Fairness metrics, for a balanced setup.
- 75% for Performance metrics and 25% for Fairness metrics, for a configuration that prioritizes performance over fairness.

- 25% for Performance metrics and 75% for Fairness metrics, for a configuration that prioritizes fairness over performance.

All executions were performed using a knowledge base containing about 650 previous executions with several options of datasets, protected attributes and algorithms, containing in your results the metrics Accuracy, Precision, Recall, F1-Score and AUC as Performance metrics and the metrics Statistical Parity Difference, Equal Opportunity Difference, Average Odds Difference, Disparate Impact and Theil Index as Fairness metrics. All metrics have the same weight in their respective grouping.

In the workflow, the algorithms *Disparate Impact Remover*, *Learning Fair Representations*, *Reweighing* and *Optimized Preprocessing* were implemented in the Pre-Processing step, the algorithms Logistic Regression, Gradient Boosting, Random Forest, Support Vector Machines, *Adversarial Debiasing*, *Exponentiated Gradient Reduction*, *Grid Search Reduction*, *Meta Fair Classifier*, *Prejudice Remover* and *Rich Subgroup Fairness* were implemented in the Processing step, and the algorithms *Equalized Odds*, *Calibrated Equalized Odds* and *Reject Option Classification* were implemented in the Post-Processing step. As no studies were found where bias reduction is used in more than one step of the Machine Learning process only one bias reduction algorithm per workflow is performed in a execution. Due to unreliable data, executions with the algorithms *Optimized Preprocessing* and *Reject Option Classification* were filtered, corresponding to approximately 5% of total base size.

4.1 Case Study 1: Autonomic Manager feasibility and usefulness

In this Case Study, the focus was on testing and verifying how the Autonomic Manager behaves, using the German Credit Dataset [2] as the dataset, Age and Nationality as the protected attribute options and a score range between 500 and 950. The results are presented below in the Tables 1, 2 and 3.

In these executions, two observations are surprising. The first is the predominant use of post-processing algorithms, especially in settings that prioritized performance, against the expectation that algorithms with bias reduction increased justice at the expense of performance. The second is the predominance of pre-processing algorithms combined with Support Vector Machines as a training algorithm in configurations that prioritized fairness. In all executions, it was possible to perceive that the most balanced choice between these two groups of metrics with completely different contexts still becomes diffuse in view of the large number of metrics and algorithms used. Furthermore, the difference between the metrics is extremely small and makes the choice even more difficult. In this context, the consolidation of metrics into groups simplifies the visualization of which workflows are more balanced, and the use of weights for each metric and each group could be calibrated with the best desired balance for a given situation. Thus, it can also be concluded that, in a development context, the process simplifies Data Scientist's decision and significantly reduces time to obtain and deploy an optimized model, since it will not require executions in several algorithms since there is a prior knowledge base. In addition, there could be processing and cost savings in case of future problems using the same dataset, since the executions saved by the teams that would use this process make room for other teams to use past data to verify better choices.

Table 1: Top options chosen by Autonomic Manager 50% Performance/50% Fairness

Workflow				Score		
Protected attribute	Pre-processing	Processing	Post-processing	Performance	Fairness	**Overall**
Age	Nothing	Logistic Regression	Equalized Odds	968	860	**914**
Nationality	Nothing	Random Forest	Calibrated Equalized Odds	902	922	**912**
Nationality	Nothing	Gradient Boosting	Calibrated Equalized Odds	870	925	**898**
Age	Nothing	Gradient Boosting	Equalized Odds	927	862	**894**
Age	Reweighing	Gradient Boosting	Nothing	804	931	**868**

Table 2: Top options chosen by Autonomic Manager 75% Performance/25% Fairness

Workflow				Score		
Protected attribute	Pre-processing	Processing	Post-processing	Performance	Fairness	**Overall**
Age	Nothing	Logistic Regression	Equalized Odds	968	860	**941**
Age	Nothing	Gradient Boosting	Equalized Odds	927	862	**910**
Nationality	Nothing	Random Forest	Calibrated Equalized Odds	902	922	**907**
Nationality	Nothing	Gradient Boosting	Calibrated Equalized Odds	870	925	**883**
Age	Nothing	Random Forest	Equalized Odds	898	799	**874**

Table 3: Top options chosen by Autonomic Manager 25% Performance/75% Fairness

Workflow				Score		
Protected attribute	Pre-processing	Processing	Post-processing	Performance	Fairness	**Overall**
Age	Disparate Impact Remover	Support Vector Machines	Nothing	747	989	**928**
Nationality	Disparate Impact Remover	Support Vector Machines	Nothing	747	989	**928**
Age	Nothing	Adversarial Debiasing	Nothing	742	979	**920**
Nationality	Reweighing	Support Vector Machines	Nothing	755	972	**918**
Nationality	Learning Fair Representations	Support Vector Machines	Nothing	755	972	**918**

4.2 Case Study 2: Workflow evolution by adding a new dataset

In this Case Study, a system evolution was performed by adding a new dataset whose content is closer to a real situation. In addition to reinforcing the Autonomic Manager's versatility in different contexts, more focus was given on system maintenance, discussing whether the steps and architectures chosen are viable to evolve and maintain the Workflow without major deterioration of your original development ideas. This time, Lendingclub Dataset [1] was used as the dataset, Income as the protected attribute, and a score range between 500 and 980. The results are presented below in Tables 4, 5 and 6.

Table 4: Top options chosen by Autonomic Manager 50% Performance/50% Fairness

Workflow				Score		
Protected attribute	Pre-processing	Processing	Post-processing	Performance	Fairness	**Overall**
Income	Learning Fair Representations	Random Forest	Nothing	991	968	**979**
Income	Nothing	Gradient Boosting	Equalized Odds	988	969	**978**
Income	Reweighing	Random Forest	Nothing	991	963	**977**
Income	Learning Fair Representations	Logistic Regression	Nothing	981	973	**977**
Income	Reweighing	Gradient Boosting	Nothing	987	964	**976**

Table 5: Top options chosen by Autonomic Manager 75% Performance/25% Fairness

Workflow				Score		
Protected attribute	Pre-processing	Processing	Post-processing	Performance	Fairness	**Overall**
Income	Nothing	Logistic Regression	Equalized Odds	985	965	**980**
Income	Learning Fair Representations	Gradient Boosting	Nothing	987	960	**980**
Income	Learning Fair Representations	Logistic Regression	Nothing	981	973	**979**
Income	Nothing	Grid Search Reduction	Nothing	989	950	**979**
Income	Reweighing	Logistic Regression	Nothing	981	965	**977**

Table 6: Top options chosen by Autonomic Manager 25% Performance/75% Fairness

Workflow				Score		
Protected attribute	Pre-processing	Processing	Post-processing	Performance	Fairness	**Overall**
Income	Learning Fair Representations	Logistic Regression	Nothing	981	973	**975**
Income	Nothing	Gradient Boosting	Equalized Odds	988	969	**974**
Income	Learning Fair Representations	Random Forest	Nothing	991	968	**973**
Income	Nothing	Exponentiated Gradient Reduction	Nothing	986	966	**971**
Income	Reweighing	Gradient Boosting	Nothing	987	964	**970**

In these executions, the algorithms with better performances changed completely, with predominance of the union between *Learning Fair Representations* for bias reduction and Logistic Regression for training. It is also noted that algorithms with post-processing bias reduction and training algorithms such as *Support Vector Machines* were not as efficient as in the previous Case Study. The main conclusion is that the Autonomic Manager can help in different problem contexts and data, and help in a decision in a more efficient and agile way. However, the data and metadata obtained did not help to understand why these changes happen.

To process the Lendingclub Dataset, modifications were necessary to do the system evolution and add this dataset as an option in Workflow. These were counted according to their *commits* made in the repository and displayed in the Table 7.

Table 7: Number of modifications performed when adding a new dataset to Workflow

Module	Changed lines	Total lines	Changed files	Total files	% changed lines	% changed files
Data Engineering	122	277	2	3	**44,04%**	**66,67%**
ML Workflow	76	1982	5	38	**3,84%**	**13,16%**
Autonomic Manager	0	457	0	10	**0,00%**	**0,00%**
Frontend	13	2905	2	14	**0,45%**	**14,29%**
Backend	4	432	1	7	**0,93%**	**14,29%**
TOTAL	**215**	**6053**	**10**	**72**	**3,55%**	**13,89%**

Adding the dataset did not require modifications to the Autonomic Manager, even though the results were completely different from the previous Case Study. This reinforces the autonomy proposed in MAPE-K *loop*, allowing different choices based on the metadata present in the Workflow without making changes. The Autonomic Manager will only require modifications if the saved Workflow metadata is modified, or if you modify any configuration intrinsic to the Manager itself, which is decoupled from the Workflow.

Frontend elements required very little modification and could be resumed to simple additions to bring the new dataset option available to the Data Scientist. The biggest changes were made to the ML Workflow and mainly to Dataset Transformations. In ML Workflow, most modifications (34 lines, or 44.74% of them) were made in data pre-processing so that training algorithms are applied correctly. Although structuring based on the *Pipes-and-Filters* architecture requires the creation of additional classes for the workflow (2 for pipes and 1 for filters), it is in the data processing and transformation steps that Data Engineers and Scientists will spend most of the time and make more modifications.

Although using the *Pipes-and-Filters* architecture requires writing a few more lines, it allows you to encapsulate the algorithms and separate concerns in a simple way, making it possible to have a good code design. This makes the system maintenance and evolution relatively simple in the Workflow and in the Frontend, as long as the developer knows which files the modifications will be made. Therefore, creating documentation is extremely important for a new developer to understand the system as a whole and not add lines unnecessarily.

4.3 Case Study 3: Workflow evolution with developer with no prior knowledge adding a new training algorithm

In this Case Study, another system evolution was performed by adding a new classification algorithm. System maintenance is discussed again, this time focusing on other developers' decisions. For this, it was verified if the chosen architectures are versatile and simple enough so that new developers can understand the system context and easily make changes, again using Lendingclub Dataset [1] as the dataset, Income as protected attribute and a score range between 500 and 980.

During a development session with another developer, lasting between 1 and 2 hours, a new classification algorithm was added to the workflow structure by him using the documentation elaborated during the previous Case Study. During the session, some items such as documentation errors and bugs were noticed and corrected, through developer's observation and feedback. Due to these problems, some troubleshooting was done in a small part of the session to avoid time wasting and allow the developer to focus on developing the workflow. After the session, the developer filled out a questionnaire indicating positive impressions from the session, a implementation without big challenges, and a profile with some experience in data and software engineering careers. Although the beginning objective was that the developer would only be guided by the documentation, he himself considered the adaptation to such an experiment quick, regardless of the help provided. This suggests that the initial decisions taken in the implementation were correct and facilitated the evolutions development.

The classification algorithm chosen for development was Naive Bayes, which is also widely used as a classifier. After development, the results are present below in Tables 8, 9 and 10.

Table 8: Top options chosen by Autonomic Manager 50% Performance/50% Fairness

Workflow				Score		
Protected attribute	Pre-processing	Processing	Post-processing	Performance	Fairness	**Overall**
Income	Learning Fair Representations	Random Forest	Nothing	991	968	**979**
Income	Nothing	Gradient Boosting	Equalized Odds	988	969	**978**
Income	Reweighing	Random Forest	Nothing	991	963	**977**
Income	Learning Fair Representations	Logistic Regression	Nothing	981	973	**977**
Income	Reweighing	Gradient Boosting	Nothing	987	964	**976**

Table 9: Top options chosen by Autonomic Manager 75% Performance/25% Fairness

Workflow				Score		
Protected attribute	Pre-processing	Processing	Post-processing	Performance	Fairness	**Overall**
Income	Nothing	Logistic Regression	Equalized Odds	985	965	**980**
Income	Learning Fair Representations	Gradient Boosting	Nothing	987	960	**980**
Income	Learning Fair Representations	Logistic Regression	Nothing	981	973	**979**
Income	Nothing	Grid Search Reduction	Nothing	989	950	**979**
Income	Reweighing	Logistic Regression	Nothing	981	965	**977**

Table 10: Top options chosen by Autonomic Manager 25% Performance/75% Fairness

Workflow				Score		
Protected attribute	Pre-processing	Processing	Post-processing	Performance	Fairness	**Overall**
Income	Nothing	Naive Bayes	Calibrated Equalized Odds	991	976	**980**
Income	Learning Fair Representations	Logistic Regression	Nothing	981	973	**975**
Income	Nothing	Gradient Boosting	Equalized Odds	988	969	**974**
Income	Learning Fair Representations	Random Forest	Nothing	991	968	**973**
Income	Nothing	Exponentiated Gradient Reduction	Nothing	986	966	**971**

For the Lendingclub Dataset context, the addition of Naive Bayes to the workflow showed its value, presenting a higher score than expected for configurations that privilege justice and was only not highlighted because of the established maximum threshold value for the Analyzer. However, as already seen in previous case studies, it may not work in other contexts and the established data and metadata do not define explanations of why Naive Bayes had a positive behavior for this dataset.

The evolutions made in the Workflow to add Naive Bayes were counted according to their *commits* made in the repository and displayed in the Table 7.

Table 11: Number of modifications performed when adding a new algorithm to Workflow

Module	Changed lines	Total lines	Changed files	Total files	% changed lines	% changed files
Data Engineering	0	277	0	3	**0,00%**	**0,00%**
ML Workflow	60	2042	5	39	**2,94%**	**12,82%**
Autonomic Manager	1	458	1	10	**0,22%**	**10,00%**
Frontend	71	2948	3	14	**2,41%**	**21,43%**
Backend	1	433	1	7	**0,23%**	**14,29%**
TOTAL	**132**	**6157**	**9**	**72**	**2,14%**	**12,50%**

This time, the only module without the need for modifications was Data Engineering, as they had already been implemented previously. Also, the only necessary modification to the Autonomic Manager and Backend was the addition of Naive Bayes as part of a parameterization, which could be transferred to an external configuration file and not require future modifications via code.

Overall, fewer changes were required than the evolution proposed in the previous Case Study. The only part that required more modifications was in the Frontend, largely because of a single component present in the Autonomic Manager Planning Settings screen. Apart from this exception, adding an algorithm is a simpler evolution to make, and together with the documentation created, a developer with relative experience can perform this task without major difficulties.

5 CONCLUSIONS AND FUTURE WORK

The reviewed literature on *Fairness* checks only for binary classification problems, and for evolutions and new methods it is likely that workflow refactorings will occur. With the *Pipes-and-Filters* architecture, it was possible to encapsulate all the procedures present in a workflow of a Machine Learning application in cohesive steps and change them in case there is a need to test with another algorithm, protected attribute or set of data, and make it easier for other developers to understand.

With the MAPE-K architecture, it was possible to create a flow so that the data obtained in the workflow could be analyzed to facilitate decision making. Although autonomy is viable, human supervision is necessary due to problems still faced by the *Fairness* theme, where the problem's social context is extremely important when evaluating whether the model is considered good or not. The use of weights for the metrics and different strategies in the analysis and planning phases of the Autonomic Manager guarantee the performance/fairness balance and help define the context for an evaluation, but it still depends on a Data Scientist and/or a Domain specialist. Domain understand what are the analyzed problem needs and if the results are acceptable to publish an optimized and fair model.

Given these observations, it can be said that these architectures have adapted very well in the main objectives implementation. As future works, it is possible to work with some possibilities. In the Autonomic Manager, the Analyzer can perform a deeper analysis by increasing the number of indicators and considering groups in addition to Fairness metrics and Performance metrics. In the workflow, introduce techniques to improve results such as *Data Augmentation* and *K-Fold Cross-Validation*. It is also possible to change the focus to solving MLOps problems, using the system to determine a better deployment in case of worsening in the metrics of a model already used by customers.

REFERENCES

[1] [n. d.]. Lendingclub Dataset. https://www.kaggle.com/datasets/wordsforthewise/lending-club

[2] [n. d.]. Statlog (German Credit Data) Data Set. https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

[3] 2005. *An Architectural Blueprint for Autonomic Computing.* Technical Report. IBM. https://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf

[4] Nadeem Abbas, Jesper Andersson, and Welf Löwe. 2010. Autonomic Software Product Lines. *ACM International Conference Proceeding Series*, 324–331.

[5] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. 2018. A reductions approach to fair classification. In *International Conference on Machine Learning*. 60–69.

[6] Alekh Agarwal, Miroslav Dudík, and Zhiwei Steven Wu. 2019. Fair regression: Quantitative definitions and reduction-based algorithms. In *International Conference on Machine Learning*. 120–129.

[7] Tom Begley, Tobias Schwedes, Christopher Frye, and Ilya Feige. 2021. Explainability for fair machine learning.

[8] Sumon Biswas and Hridesh Rajan. 2020. Do the Machine Learning Models on a Crowd Sourced Platform Exhibit Bias? An Empirical Study on Model Fairness. (2020), 642–653.

[9] Joy Buolamwini and Timnit Gebru. 2018. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In *Conference on Fairness, Accountability and Transparency (Proceedings of Machine Learning Research, Vol. 81)*. 77–91.

[10] Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. 2017. Optimized Pre-Processing for Discrimination Prevention. In *Advances in Neural Information Processing Systems*. https://proceedings.neurips.cc/paper/2017/file/9a49a25d845a483fae4be7e341368e36-Paper.pdf

[11] L Elisa Celis, Lingxiao Huang, Vijay Keswani, and Nisheeth K Vishnoi. 2019. Classification with fairness constraints: A meta-algorithm with provable guarantees. In *Proceedings of the conference on fairness, accountability, and transparency*. 319–328.

[12] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and Removing Disparate Impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 259–268. https://doi.org/10.1145/2783258.2783311

[13] Moritz Hardt, Eric Price, Eric Price, and Nati Srebro. 2016. Equality of Opportunity in Supervised Learning. In *Advances in Neural Information Processing Systems*. https://proceedings.neurips.cc/paper/2016/file/9d2682367c3935defcb1f9e247a97c0d-Paper.pdf

[14] Faisal Kamiran and Toon Calders. 2011. Data Pre-Processing Techniques for Classification without Discrimination. *Knowledge and Information Systems* (2011). https://www.researchgate.net/publication/228975972_Data_Pre-Processing_Techniques_for_Classification_without_Discrimination

[15] Faisal Kamiran, Asim Karim, and Xiangliang Zhang. 2012. Decision Theory for Discrimination-Aware Classification. In *2012 IEEE 12th International Conference on Data Mining*. 924–929.

[16] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. 2012. Fairness-Aware Classifier with Prejudice Remover Regularizer. 35–50. https://www.researchgate.net/publication/262176212_Fairness-Aware_Classifier_with_Prejudice_Remover_Regularizer

[17] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. 2018. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *International Conference on Machine Learning*. 2564–2572.

[18] Jeffrey Kephart and D.M. Chess. 2003. The Vision Of Autonomic Computing. (2003), 41 – 50.

[19] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A Survey on Bias and Fairness in Machine Learning. *Comput. Surveys* (2021), 1–35.

[20] Carlos Mougan, José Manuel Álvarez, Gourab K. Patro, Salvatore Ruggieri, and Steffen Staab. 2022. Fairness implications of encoding protected categorical attributes. (2022).

[21] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. 2017. On Fairness and Calibration. In *Advances in Neural Information Processing Systems*. https://proceedings.neurips.cc/paper/2017/file/b8b9c74ac526fffbeb2d39ab038d1cd7-Paper.pdf

[22] Till Speicher, Hoda Heidari, Nina Grgic-Hlaca, Krishna P. Gummadi, Adish Singla, Adrian Weller, and Muhammad Bilal Zafar. 2018. A Unified Approach to Quantifying Algorithmic Unfairness. (2018). http://dx.doi.org/10.1145/3219819.3220046

[23] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. 2013. Learning Fair Representations. In *Proceedings of the 30th International Conference on Machine Learning*. 325–333. https://proceedings.mlr.press/v28/zemel13.html

[24] Brian Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating Unwanted Biases with Adversarial Learning. 335–340. https://www.researchgate.net/publication/330299272_Mitigating_Unwanted_Biases_with_Adversarial_Learning