A Methodology for Non-Functional Property Evaluation of Machine Learning Models

Marco Anisetti
Università degli Studi di Milano
Milano, Italy
marco.anisetti@unimi.it

Claudio A. Ardagna
Università degli Studi di Milano
Milano, Italy
claudio.ardagna@unimi.it

Ernesto Damiani
C2PS – Khalifa University
Abu Dhabi, UAE
ernesto.damiani@kustar.ac.ae

Paolo G. Panero
SORIS S.p.A
Torino, Italy
paolo.panero@soris.torino.it

ABSTRACT

The pervasive diffusion of Machine Learning (ML) in many critical domains and application scenarios has revolutionized implementation and working of modern IT systems. The behavior of modern systems often depends on the behavior of ML models, which are treated as black boxes, thus making automated decisions based on inference unpredictable. In this context, there is an increasing need of verifying the non-functional properties of ML models, such as, fairness and privacy, to the aim of providing certified ML-based applications and services. In this paper, we propose a methodology based on Multi-Armed Bandit for evaluating non-functional properties of ML models. Our methodology adopts Thompson sampling, Monte Carlo Simulation, and Value Remaining. An experimental evaluation in a real-world scenario is presented to prove the applicability of our approach in evaluating the fairness of different ML models.

CCS CONCEPTS

• **Computing methodologies** → Machine learning; • **Security and privacy** → **Formal methods and theory of security**; • **Software and its engineering** → **Extra-functional properties**.

KEYWORDS

Non-functional properties, Machine Learning Assurance, Multi-armed bandit

ACM Reference Format:
Marco Anisetti, Claudio A. Ardagna, Ernesto Damiani, and Paolo G. Panero. 2020. A Methodology for Non-Functional Property Evaluation of Machine Learning Models. In *12th International Conference on Management of Digital EcoSystems (MEDES '20), November 2–4, 2020, Virtual Event, United Arab Emirates.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3415958.3433101

MEDES '20, November 2–4, 2020, Virtual Event, United Arab Emirates

ACM ISBN 978-1-4503-8115-4/20/11…$15.00
https://doi.org/10.1145/3415958.3433101

1 INTRODUCTION

Today, traditional software systems based on deterministic algorithms are increasingly substituted by systems where ML models reason on data to calculate a solution to individual instances of a problem [17]. ML models are at the core of these systems and are deeply changing their design and development, as well as their monitoring and verification. The verification of non-functional properties of software systems is a major challenge by itself, which accompanied all steps of software system evolution from traditional software based-systems to service-based systems, cloud, and Internet of Things [4]. Assurance techniques [7] have been developed to address it focusing on the complexity and heterogeneity of systems, requirements on costs, and resource limitations. This challenge has been even exacerbated by the development of systems based on ML models. According to a study by Yoshioka and Ishikawa [46] involving several industry experts, assessment of non-functional requirements is the most complex stage of a ML-based software system development. ML models are in fact less transparent and difficult to monitor, thus impeding the adoption of traditional assurance approaches [7].

This paper presents a methodology based on Multi-Armed Bandit (MAB) for the evaluation of non-functional properties of ML models, which represent the cornerstone for a future certification of ML-based software systems. Specifically we consider a scenario where multiple ML models are available and can be selectively compared in terms of their non-functional properties. Our solution can be initially adopted at development time to select the best model to be integrated in the system (*static evaluation*) and then used to monitor the model behavior at run time (*dynamic evaluation*). In the latter case, ML models evolve over time (e.g., partial re-training/tuning) and therefore their performance in terms of non-functional property support is monitored possibly triggering substitution of the deployed model. In this paper we depart from the assumption of having the complete training process under control and consider the worst-case scenario in which pre-trained models need to be verified (i.e., tested and monitored). We note that our assumption does not affect the generality of our approach that can be applied in scenarios with full control of the training process, complementing the precision-recall evaluation of ML models with a non-functional property score.

The contribution of this paper is threefold. Fist, we propose a taxonomy of non-functional properties of ML models. Second, we propose a MAB-based methodology evaluating and comparing non-functional properties of different ML models. Such models can vary depending on the selected ML algorithm or training process. Third, we experimentally evaluate our solution in a real-world scenario focusing on the property fairness.

The paper is organized as follows. Section 2 presents our taxonomy of ML non-functional properties. Section 3 describes our reference scenario focusing on property fairness. Section 4 presents our MAB-based methodology. Section 5 presents the implementation of our methodology. Section 6 shows our experimental results. Section 7 presents the related work and Section 8 draws our conclusions.

2 NON-FUNCTIONAL PROPERTY TAXONOMY

According to [27] a non-functional requirement "*describes not what the software will do but how the software will do it.*" The unambiguous specification of a security property is an important aspect for the correct evaluation of non-functional requirements. For instance, Glinz et al. [22] presented *11* different definitions of non-functional properties. Anisetti at al. [5, 6] detailed security non-functional properties for web services in the context of a certification framework. When applied to machine learning models, however, classic non-functional properties must be rethought and redefined. ML models are opaque (black-box) compared to traditional source code and it is difficult to predict run-time software behavior at design time.

We then propose a taxonomy on non-functional properties for ML models as follows.

- **Transparency**: the property targeted by explainable AI solutions. Many definitions of this property have been proposed; most have to do with the feasibility of ex-post interpretation, that is, a human being able to step through the procedure that the model applied upon receiving an input and reach the same decision. We specialize it into two sub-properties:
  - **Explainability**: the ability of explaining the model (what has been learned), on one side, and individual decisions taken by the model, on the other side. Currently the ability to explain decisions depends on the learning algorithm used to train the model [45].
  - **Interpretability**: it expresses a cause and the resulting effect can be observed. It permits to predict consequences on models when changes to input data are observed [21].
- **Fairness**: defined as the absence of discrimination, that is, prejudice against an individual or a group based on the value of specific features [26].
- **Legality**: measuring the compliance to laws and regulation. The pervasiveness of Artificial Intelligence in fact has pushed public institutions to legislate on many non-functional aspects of AI, including privacy, quality, fundamental human rights, fairness, civil and criminal liability [23].
- **Maintainability**: ML models do not shirk the need to be adapted to new needs or to the changing of the environment. Maintainability (also known as *retrainability*) expresses the ability of going through complete or partial model retraining [38].
- **Modularity**: measure the ability to successfully manage large or highly complex systems. Its applicability depends on the specific algorithm used for model training [35].
- **Performance**: the performance of a computation. Usually evaluated by metrics that depend on the specific problem solved by the model. Many metrics are available such as accuracy, precision/recall, Area Under the ROC curve, to name but a few. Often performance and accuracy are used interchangeably.
- **Privacy**: it expresses the notion that ML models do not store information about their training set, nor such information can be inferred from the models' output. Often, this refers to the compliance to a specific national or international regulation, such as the General Data Protection Regulation (GDPR).
- **Reliability**: the ability of the software to operate without failures and to maintain a certain level of performance when used under normal conditions and for a certain period of time [32]. Bosnic et al. [12] restricted its meaning for ML models as the quality of the single prediction, moving more closely to a concept of accuracy of prediction.
- **Safety**: "*the expectation that a system does not, under defined conditions, lead to a state in which human life, health, property, or the environment is endangered*" [27]. It is a concept closely related to the application domain.
- **Scalability**: ML models follow the usual rule that as the context grows, the system must be able to adapt and guarantee a sufficient level of performance. Property scalability expresses the capability of handling the increase in the data volume without affecting the model's inference time.
- **Security**: it expresses ML robustness against attacks. ML techniques include new assets to be protected from attackers, such as data, models, hyperparameters, weights and coefficients used by models, and proprietary algorithms. ML can be attacked with traditional techniques or ad hoc techniques (e.g., adversarial machine learning techniques).
- **Testability**: the very nature of ML models makes testing extremely arduous since, unlike traditional software, output is not known in advance but it is represented by a prediction. Moreover depending on the algorithms used during the training, the different generated models can provide different outputs starting from the same input data. It is not possible to test a model for which there is no predicted data available. In this case only a functional expert can evaluate what is predicted by a model based on the same input data and repeating the cognitive process.
- **Usability**: it expresses the effort required by users to learn the software, prepare input data and interpret the results.

3 REFERENCE SCENARIO: PROPERTY FAIRNESS

We consider a reference scenario where different pre-trained ML models are compared on the basis of a given non-functional property. More in detail we consider *i)* a static evaluation scenario where ML models are compared at development time to select the one

best supporting the non-functional property and *ii)* a dynamic evaluation scenario where the models are partially updated at run time and compared with the original model possibly triggering model substitution. The first scenario complements traditional approaches where ML models are compared in terms of precision and recall with a comparison based on non-functional property evaluation. The second scenario considers ML models that are updated (e.g., via partial re-training/tuning) based on the operation conditions, and aims to avoid the possibility of employing a sub-optimal model whose performance degrades over time.

For conciseness, but with no loss of generality, we focus on a specific functional property, namely, fairness. Modelling the concept of fairness is extremely complex, since it is affected by ethical, socio-economic, cultural and religious factors [19, 42]. More than 20 different definitions of fairness are known in literature [44]; among the most relevant ones we emphasize the following:

- *fairness through unawareness*, achieved without using explicitly protected attributes in the predictive process. Currently this definition is considered as naive;
- *group fairness*, stating that the predictor must predict a certain result for all individuals regardless of their belonging to a group, making sure that the probability with which a certain result is assigned is almost equal between groups [15];
- *individual fairness*, it defines a fair predictor as a predictor that produces similar results for similar individuals [18]. From an ethical point of view, every individual is worthy of protection and, moreover, in case of particularly sensitive application domains, the individual discriminated can appeal to the appropriate Authority against those who used the model to make the decision;
- *equality of opportunity*, it requires no discrimination between the groups of those receiving the advantageous forecast [25].

Many concrete projects have been developed trying to implement evaluation of machine learning models fairness [2, 3, 9, 11, 20]. These projects, however, produce reports that show several different metrics and require an analysis of the results by a human being with specific skills, in charge of evaluating each case on the basis of the application context. The proposal in this paper requires no human intervention. To demonstrate it, we adopt a simple notion of fairness based on the identification of a set of protected attributes [19].

4 THE MAB-BASED METHODOLOGY

We propose a methodology based on *Multi-Armed Bandit* for comparatively evaluating the non-functional properties of ML models. Our methodology permits to compare different models to find the one with the best support for a specific non-functional property.

Multi-Armed Bandit represents a class of problems where a sequential experiment has the purpose of maximizing the reward earned by performing a single action chosen in a set of different actions, every action (arm) with a different odds of winning/losing. Typically there are K arms each of which associated with an unknown value v_a representing arm's reward. The goal of the experiment is to choose the arm with the best reward to pile up the greatest reward amount across the sequence of experiments. Rewards rely on a probability distribution $f_a(y|\theta)$ being y the observed reward and θ a set of unknown parameters that must be learned through experimentation. $v_a(\theta)$ is a function of θ such that if θ is known, the optimum arm is known as well. This class of problems is based, from a mathematical point of view, on Bayesian inference. The success probability is unknown before the execution of each experiment and is modelled by a probability distribution called *Beta distribution*, which is a distribution defined by 2 parameters (α and β) on the [0,1] interval. This distribution is used within Bayesian statistics to rule probability p of a Bernoulli process. Beta distribution is a special case of the Dirichlet distribution, is related to the Gamma distribution, and has the following probability density function:

$$f(x; a, b) = \frac{1}{B(\alpha, \beta)} x^{(\alpha-1)}(1-x)^{\beta-1} \quad (1)$$

where the normalisation function B is the Euler beta function $B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt$.

While the experiment is executed, arms are pulled and their Beta distributions are updated according to arms' results. In our methodology the selected arms refer to the ML models that are evaluated against the requested non-functional property. After pulling an arm, it detects and collects the result, and updates the arm's Beta distribution. Then it checks if the experiment can terminate by evaluating whether the winning arm has been identified. If the experiment cannot terminate, it chooses the next arm to be pulled and continues.

4.1 Solving the MAB

Many solutions to the MAB problem have been proposed in literature. A greedy solution was proposed by Sutton et al. [41], but it turned out to be suboptimal. In this paper we use Thompson Sampling [14]. Let us consider y_t denoting a set of data observed at time t; we define

$$w_{at} = \Pr(a \text{ is optimal} \mid y_t)$$
$$= \int l(a = \arg\max v_a(\theta) p(\theta|y_t) d\theta$$

where $p(\theta|y_t)$ is the Bayesian posterior probability distribution of θ considering the data observed at time t. Thompson sampling assigns observation at time $t+1$ to arm a with probability w_{at}, which maximizes the reward. Once an arm has been chosen by Thompson Sampling, its actual performance must be evaluated by applying a score function to it; the algorithm can then determine whether the choice made by Thompson Sampling was good or not. As the experiment goes on, arms with best performance are rewarded by updating their Beta distribution in order to recall their good performance. This causes arms that do not provide good performance to be "punished" and ending up being "pulled" fewer times. Probability w_{at} can be calculated with a Monte Carlo simulation [34] that permits to define an early experiment termination before having consumed all the runs. Early termination is based on each arm's probability to be the winner of the experiment and the notion of "value remaining in the experiment" [39], according to which the experiment ends when 95% of the samples of a Monte Carlo simulation have a residual value less than 1% of the value of the optimal arm. This is assumed to minimize the "regret" (the missed reward) for the early termination of the experiment. Formally, let us denote θ_0 as the true value of θ and a^*=arg max$_a$ $v_a(\theta_0)$ as the

very optimal arm; early termination of experiment regret at time t is given by $v_{a^*}(\theta_0) - v_{a_t^*}(\theta_0)$, which stands for the value of the difference between the very optimal arm and the seemingly optimal arm at time t. Basically, regret is not directly observable, but may be computed by means of a posterior probability distribution. Indeed, being $v_*(\theta^{(g)})$=max$_a$ $v_a(\theta^{(g)})$ with $\theta^{(g)}$ drawn from $p(\theta|y_t)$, we have

$$r^{(g)} = v_*(\theta^{(g)}) - v_{a_t^*}(\theta_0)$$

deriving from a regret posterior probability distribution. Note that $v_*(\theta^{(g)})$ represents the maximum available value within Monte Carlo draw set g and $v_{a_t^*}(\theta_0)$ represents the value (still taken in g) for the arm considered best among Monte Carlo extractions. Their difference is usually 0, but can be positive in some cases. The regret's statistical distribution can be summarized by high quantile, usually the 95$^{\text{th}}$ percentile, to show the potential value remaining (PVR) of the experiment. It represents the value of the reward, for each run, that would be lost if the experiment ends at time t. This value can be expressed in the unit of measure used in the experiment. In real scenarios, with the need to find an arm that has a very high reward, a criterion normally used is to end the experiment when the PVR is lower than a certain significant threshold. In a MAB type experiment, it is possible that more than one arm show the same performance. In this case, thanks to PVR, the experiment can end with two groups of arms: the first containing the arms lower than the threshold, the second with arms that can be almost equivalent to each other. From this second group, any arm can be chosen to conclude the experiment. In addition, it is possible to use w_{at} value to choose the final arm among potential winners. Lastly, regret may be calculated (and this is our choice) as the percentage deviation from the arm currently identified as optimal, so that draws from the posterior probability are given by

$$p^{(g)} = \frac{v_*(\theta^{(g)}) - v_{a_t^*}(\theta^{(g)})}{v_{a_t^*}(\theta^{(g)})} \quad (2)$$

which represents a percentage value independent from the adopted unit of measure.

Defining a suitable score function is crucial for our methodology, since it is fundamental to determine a winner arm with the Thompson sampling approach. The score function can be used to enforce non-functional properties and its definition depends on the property itself and the scenario (e.g., the available data and the type of prediction). For instance, the score function for property fairness in Section 3 is the variance of the prediction varying the set of protected attributes. We leave the mapping of non-functional properties in Section 2 to appropriate score functions for our future work. For instance, the Shapley Additive Explanation function in [30] can be adapted to our scenario to produce a score function that is computationally efficient.

5 MAB IMPLEMENTATION

We present the implementation of our methodology in Section 4.1 following the pseudo-code in Figure 1.

The models to be evaluated and the data set used during processing are first loaded and trigger the execution of the methodology. MAB algorithm (function **k_arm_bandit**) receives as input *i)* the models (*arms*), *ii)* the data needed to test the models, *iii)* parameter *par_alpha*, used by value remaining in experiment to terminate when the (1 - *par_alpha*)th percentile of residual value is less than 1% of MAB wins of an arm, and *iv)* the minimum number of iterations to be done (*burn_in*) to prevent the experiment to end without executing enough draws. MAB algorithm starts setting the threshold value of a given fairness-related score function, which is used to decide whether an arm is a winner or a loser. Then, for every row in the data set:

- it uses function **thompson_sampling** to choose an arm to experiment. This is done by drawing a sample from each arm's beta distribution in equation (1) and returning the one with the maximum value;
- once the arm is chosen, function **fair_prediction** makes a prediction with the inline data and the chosen model. The prediction is carried out in an ad hoc function that, starting from the single input data corresponding to an individual, generates many different data lines covering all the possible combinations of those protected attributes that are relevant for the given score function. Once these data lines (fake rows) have been generated, they are given as input to the model that makes the predictions. Then the score function of this set of predictions is calculated and returned as output;
- if the score value returned as output exceeds the threshold set at the beginning of the experiment (function **k_arm_bandit**), the model has failed, that is, it has made non-fair predictions. Otherwise, when the variance is below the threshold, the prediction is considered fair. Depending on the result, the positive or negative outcome of the prediction is tracked for the arm by increasing arm's alpha or beta values, respectively;
- function **monte_carlo_simulation** is executed to simulate probabilities of arms being winners. This is done by creating a two-dimensional matrix, having dimensions represented by the number of arms and a fixed number which in our case is 100, filled with samples extracted from arms' beta distributions in equation (1). Then, it counts the frequency of each arm being winner, and returns the matrix and the probability of each arm being the winner;
- function **k_arm_bandit** records current estimates of each arm's wins and checks whether the experiment can terminate by verifying whether there have been enough iterations against (*burn_in*) and by calling function **should_terminate**. The latter checks if value remaining in the experiment is less than 1% of the number of wins of the winning arm, by calculating the regret applying equation (2) against the (1 âĂŞ *par_alpha*)th percentile.

6 EXPERIMENTAL EVALUATION

We experimentally evaluated our MAB methodology in a real-world scenario.

6.1 Experimental Setup

Let us consider the problem of estimating the bail that an individual awaiting trial can pay to be set free using a ML approach. In such a context, fairness of prediction is of paramount importance, even more important than the precision of the bail prediction. Our

```
INPUT
arms[]: models to evaluate
data[]: data set for experiment

OUTPUT
winner_idx: index of winner arm
est_models[]: percentage of estimated
              wins for each arm
draws[]: number of draws for each arm

MAIN
Initialize par_alpha: terminate experiment
          when the (1- par_alpha)th
          percentile of the remaining value is
          less than 1% of the winner's wins
Initialize burn_in: minimum number of
          iterations
(winner_idx, est_models[], draws[]) =
          k_arm_bandit(arms[],data[],
          par_alpha,burn_in)

K_ARM_BANDIT
Initialize VARIANCE: threshold value upon
          which prediction is fair
for each item∈data[]
   arm= thompson_sampling(arms[])
   var= fair_prediction(arm,item)
   if (var< VARIANCE)
     record arm's success by increasing
     arm's alpha parameter
   else
     record arm's failure by increasing
     arm's beta parameter
   (mc[], p_winner[]) =
     monte_carlo_simulation(arms[])
   est_models[] = record current estimates of each
     arms[]' wins as alpha / (alpha + beta)
   if (item.index > burn_in) AND
   should_terminate(p_winner[], est_models[],
   mc[], par_alpha)
     break
draws[] = count draws for each arms[]
return(winner_idx, est_models[], draws[])

THOMPSON_SAMPLING
for each arm∈arms[]
   sample_p[] = draw sample from arm's
   beta distribution
idx= arg max(sample_p[])
return(idx)

FAIR_PREDICTION
rows[] = item
rows[] += generate test data for all protected
          groups against item
predicted_vals[] = arm.predict(rows[])
return(variance(predicted_vals[]))

MONTE_CARLO_SIMULATION
Initialize draws= 100
alphas[], betas[] = arms[]'s alphas and betas
mc[] = matrix [arms.dimension, draws)
          of samples from beta distributions of
          alphas and betas
counts[] = count frequency of each arm
          being winner
for each count∈counts[]
   p_winner[] = count / draws to approximate
   probability distribution
return(mc[],p_winner[] )

SHOULD_TERMINATE
winner_idx= arg max(p_winner[])
values_remaining = (max(mc) - mc(winner_idx))
   / mc(winner_idx)
percentile = compute values_remaining'
   ( 1 - par_alpha)th percentile
if (percentile< (0.01 * est_models[winner_idx]))
  return(TRUE)
else
  return(FALSE)
```

Figure 1: Pseudocode of our MAB-Based Methodology.

experimental evaluation used the Connecticut State Department of Correction data set[1] that provides a daily updated list of people detained in the Department's facilities awaiting for a trial. This data set anonymously discloses data of individual people detained in the correctional facilities every day starting from July 1st, 2016. It contains attributes such as last admission date, race, gender, age, type of offence and facility description, in more than 4 millions data points (at the download date). We partitioned the data set into training (80% of the available data) and testing (20% of the available data) sets. We used the training set to train three supervised machine learning models suitable for the bail prediction (i.e., classification and regression trees, K-nearest neighbours and naive Bayes classifier) and then considered such models as our pre-trained models to be compared for fairness. We then partitioned the test set into 4 different experiments defined as follows: *i)* exp_1: 114.135 rows (a portion of the test set), *ii)* exp_2: 986.977 rows (the entire test set), *iii)* exp_3: 1.321 rows (a subset of experiment 1), exp_4: 50 rows (very limited test set). In the following, we consider race and gender as protected attributes for our notion of fairness (Section 3), and define the score function as the variance of the predicted bails. Our solution has been implemented in Python and tested on a laptop with Intel Core i7 processor at 2.60 GHz and with 16.0 GB of RAM memory running a Microsoft Windows 10 OS. All the scripts used to implement our methodology and to carry out this experimental evaluation are available at https://bit.ly/3jCZ3KZ.

[1]Available at https://data.ct.gov/Public-Safety/Accused-Pre-Trial-Inmates-in-Correctional-Faciliti/b674-jy6w and downloaded February 21st, 2020 in the form of a CSV text file

6.2 Experimental Results

The score function threshold was set to 200, a very low threshold that has been chosen to better pick bail variations among runs. Table 1 shows the results of the application of the MAB methodology in our 4 experiments. Each cell represents the number of times a model has been chosen by Thompson sampling within our solution.

Table 1: MAB results for each experiments.

Models (arms) / Experiments	exp_1	exp_2	exp_3	exp_4
Classification and Regression Trees	16	9	9	10
K-nearest neighbours	85	61	17	17
Naive Bayes classifier	1300	1331	1295	23
TOTAL DRAWS	1401	1401	1321	50

Figure 2 shows the percentage of *draws* made for each experiment on each model. In all experiments the model trained with Naive Bayes classifier has been the most successful. Experiments 1, 2 and 3 completed much earlier than consuming the whole data set thanks to the application of the *value remaining in experiment*, as the model winning prevailed over others. We can observe that the models most likely to be the ones with the best performance were extracted a greater number of times. This clarify how the Thompson sampling with Monte Carlo simulations using value remaining permitted to identify the "winning" model over the others, choosing it preferentially, although models with lower performance were also tested from time to time.

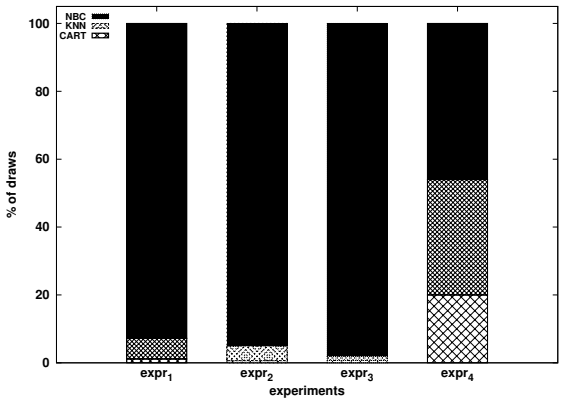

Figure 2: Percentage of draws for each experiment. Naive Bayes Classifier (NBC), Classification and Regression Trees (CART), and K-nearest neighbours (KNN).

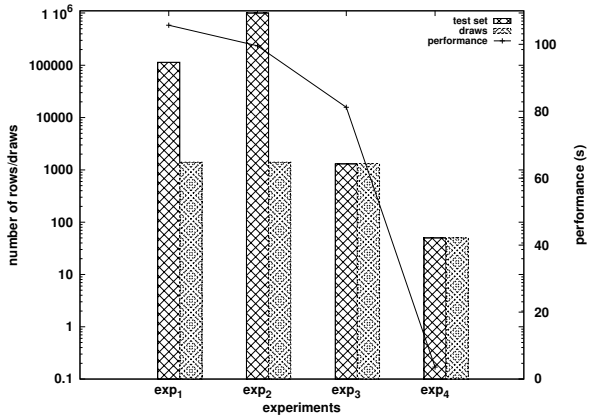

Figure 3: Performance in terms of seconds compared to experiment's dataset dimension(in log scale) and number of draws.

The explore/exploit trade-off has therefore been fully addressed. The result was confirmed by experiment exp_3, where a data set forcibly lower than the number of iterations normally sufficient to stop the experiment was used. In experiments exp_3 and exp_4, instead, the algorithm consumed all the input data since the low cardinality of the input did not allow to terminate the experiment in advance. The experiment exp_4 shows that, given the few available iterations, the MAB does not highlight a preference for a clear choice.

6.3 Performance evaluation

The experiments took the following time to complete: *i)* exp_1: 105.711*s*, *ii)* exp_2: 99.546*s*, *iii)* exp_3: 81.176*s*, *iv)* exp_4: 3.588*s*. We note that the execution time is more affected by data set rows than draws. The reason is that the value remaining in experiment allows experiments to end earlier. So it takes almost the same CPU effort to Thompson Sampling to solve the MAB problem. Figure 3 shows processing time (expressed in seconds) for every experiment, the related test set size and the number of draws. We note that executing the experiment on the same data multiple times lead to partially different results, but the same winning arm.

6.4 Discussion on fairness and score function

Let us consider the sample data shown in Table 2 that shows the details of a series of predictions made during our experiments. The table focuses on unfair predictions made by models. Column *run* refers to the progressive execution, that is, a row has been read from the data set and has been processed. As already described, for every row read from the data set, fake rows are generated to cover all the possible combinations of protected attributes. To better understand the table, predictions must be read by grouping them by column *run*. Column *bail* contains the result of the prediction (the predicted bail for the input data) and column *variance* shows the variance of *bail* among the *run*. Column *arm* shows the model index in charge of the prediction chosen by Thompson Sampling. In the case of run 5, there is a discrimination between females and males, having females lower predicted bail. Run 124 shows a discrimination between black people and other races, having black people lower predicted bail, no matter for gender. Run 481 shows even more discrimination among races with three different predicted bail amounts for indeterminate race, white and asian/black/Hispanic race.

7 RELATED WORK

Continuous monitoring and evaluation of machine learning model properties are hot research topics.The increasing adoption of ML models at the basis of automatic decisions involving humans in fact calls for a new governance of ML models, allowing users to understand where and when (if not how) an ML inference came to be. The research agenda for AI governance [16] aims to define policies regulating the use of AI according to the desired properties of AI systems. System properties can be classified in architectural properties (how the system is structured), functional properties (what the system can do), and non-functional properties (the one in this paper) [17, 24]. In this context, some solutions have been provided to represent architectural properties of AI pipelines via symbolic representation [28, 37]. Unfortunately, as discussed in [17], automatic reasoning on formal representations of ML models is still in its infancy [29], and only solutions to verify robustness with respect to adversarial perturbations are available [10]. Another possible verification approach is based on certification [7]; however, traditional solutions cannot be applied since the new generation of ML-powered applications challenges the assumption of classic verification and testing that the business logic of the application, or at least its design, is stable at verification time.

Many concrete projects have been developed trying to implement evaluation of machine learning models on specific properties, most of them focusing on property fairness. FlipTest [11] is a test methodology that allows to highlight discriminatory trends in classifier algorithm models. It is a black-box technique that uses the concept of optimal transport to compare individuals located in groups of different protected attributes, comparing the behavior of the model in the face of a variation in sensitive attributes. Agarwal et al. [2] describe a similar approach based on *i)* techniques of symbolic execution used for automatic generation of inputs through the

Table 2: Sample data for unfair runs. CART refers to Classification and Regression Tree arm and KNN to K-Nearest Neighbours.

(Run)	Age	Offense Id.	Facility Id.	Detainer Id.	Gender	Race	Bail	Variance	Arm
5	31	92	35	4	Female	Black	10000	60025000000	CART
5	31	92	35	4	Female	Indeterminate	10000	60025000000	CART
5	31	92	35	4	Female	Asian	10000	60025000000	CART
5	31	92	35	4	Female	Hispanic	10000	60025000000	CART
5	31	92	35	4	Female	White	10000	60025000000	CART
5	31	92	35	4	Male	Indeterminate	500000	60025000000	CART
5	31	92	35	4	Male	Asian	500000	60025000000	CART
5	31	92	35	4	Male	Black	500000	60025000000	CART
5	31	92	35	4	Male	Hispanic	500000	60025000000	CART
5	31	92	35	4	Male	White	500000	60025000000	CART
124	30	33	14	7	Male	Hispanic	1700000	400000000	KNN
124	30	33	14	7	Female	Indeterminate	1700000	400000000	KNN
124	30	33	14	7	Female	Asian	1700000	400000000	KNN
124	30	33	14	7	Female	Black	1650000	400000000	KNN
124	30	33	14	7	Female	Hispanic	1700000	400000000	KNN
124	30	33	14	7	Female	White	1700000	400000000	KNN
124	30	33	14	7	Male	Indeterminate	1700000	400000000	KNN
124	30	33	14	7	Male	Asian	1700000	400000000	KNN
124	30	33	14	7	Male	Black	1650000	400000000	KNN
124	30	33	14	7	Male	White	1700000	400000000	KNN
481	33	24	35	4	Female	Indeterminate	20300	41313600	KNN
481	33	24	35	4	Female	Asian	5000	41313600	KNN
481	33	24	35	4	Female	Black	5000	41313600	KNN
481	33	24	35	4	Female	Hispanic	5000	41313600	KNN
481	33	24	35	4	Female	White	15050	41313600	KNN
481	33	24	35	4	Male	Indeterminate	20300	41313600	KNN
481	33	24	35	4	Male	Asian	5000	41313600	KNN
481	33	24	35	4	Male	Black	5000	41313600	KNN
481	33	24	35	4	Male	Hispanic	5000	41313600	KNN
481	33	24	35	4	Male	White	15050	41313600	KNN

systematic exploration of all the execution paths of a software and *ii)* local explainability to provide paths to the symbolic execution which, compared to a black-box model such as an an ML model, could not trace all the paths. AIF360 [9] is an open source tool that allows models to be subjected to an evaluation process that outputs a series of metrics for accuracy and permits to evaluate data sets from the point of view of their fairness. It provides algorithms to be *injected* into the solutions under scrutiny to make them fairer. FairDM [3] is a framework for the evaluation of decision models which permits, among other metrics, to measure metrics of individual and group fairness. Themis [20] is an open source software that proposes to evaluate the fairness of models through automatic testing, by capturing cause/effect relationships between inputs and outputs. It automatically generate test cases and evaluate fairness of outputs based on classic individual and group fairness metrics. These projects mainly produce reports that need to be interpreted by expert users. The proposal in this paper requires no human intervention. Other work has been done in the context of ethics of AI [13]. In particular, research has been conducted to define methodologies to implement non-functional properties into ML models at training time. Two main approaches have been followed training set filtering [1, 33] and bias shaping [31, 36]. These approaches are different from the one in this paper because they try to support properties by design rather than verifying them in operation.

Focusing on the solution in this paper, the idea to use Multi-Armed bandit to combine multiple machine learning models was first studied by Auer et al. in [8] who proposed to explore the model space at run time to find the model variation capable of keeping up the desired property (accuracy). Multi-Armed bandit implementations can also verify model variations to maximize a defined score function representing the desired property [43]. More recently Spiro at al. [40] points out how model enforcement requires checking functional and non-functional properties. Much inline with our approach, Damiani at al. [17] uses BAM in the context of ML model evaluation based on non-functional properties, presenting some preliminary discussion and results on the topic. Among the papers which gave stronger theoretical contribution to our proposal, Kesner et al. [28] lay the foundations of using statistical analysis to human decision-making processes. Our approach puts forward the idea to use it to evaluate machine learning models with regards to software properties. Chapelle and Li [14] used Thompson Sampling as an outperforming method to solve Multi-Armed bandit problem. The use of Monte Carlo simulation to calculate the probability of an arm to be the winner came out following Metropolis in [34] and, for the experiment termination, value remaining in experiment was widely studied by Scott in [39].

8 CONCLUSIONS

We presented a methodology based on Multi-armed Bandit for evaluating non-functional properties of ML models. The methodology used Thompson sampling, Monte Carlo simulations, value remaining in experiment to effectively select the ML model with

a specific non-functional property at development time and monitor the continuous support of the non-functional property at run time, possibly triggering model substitution. While being tested on property fairness, the proposed solution can be applied to any properties and scenarios where alternative ML models can be selected. It can therefore be positioned upstream of a certification authority in charge of issuing digital certificates that guarantee compliance to a desired non-functional requirement. The paper leaves space for future work. First, we will focus on the definition of a score function for each property in the taxonomy, starting from existing score functions in literature [30]. Then, we will investigate an analytic approach to provide sound thresholds to determine arms' wins or losses, in absence of a score function. Finally, we will focus on the definition of a certification scheme based on the approach in this paper, to the aim of providing certified machine learning models.

ACKNOWLEDGMENTS

This work has received funding from CONCORDIA, the Cybersecurity Competence Network supported by the European Union's Horizon 2020 Research and Innovation program under grant agreement No 830927.

REFERENCES

[1] T. Adel, I. Valera, Z. Ghahramani, and A. Weller. One-network adversarial fairness. In *Proc. of 33rd AAAI Conference on Artificial Intelligence*, Honolulu, HI, USA, January–February 2019.
[2] A. Agarwal, P. Lohia, S. Nagar, K. Dey, and D. Saha. Automated Test Generation to Detect Individual Discrimination in AI Models. 2018.
[3] Y. Ahn and Y.-R. Lin. FairSight: Visual Analytics for Fairness in Decision Making. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1–1, 2019.
[4] M. Anisetti, C. Ardagna, E. Damiani, and F. Gaudenzi. A semi-automatic and trustworthy scheme for continuous cloud service certification. *IEEE Transactions on Services Computing*, 2020.
[5] M. Anisetti, C. Ardagna, E. Damiani, and F. Saonara. A test-based security certification scheme for web services. *ACM Transactions on the Web (TWEB)*, 7(2):1–41, May 2013.
[6] M. Anisetti, C. A. Ardagna, and E. Damiani. A low-cost security certification scheme for evolving services. In *2012 IEEE 19th International Conference on Web Services*, pages 122–129. IEEE, 2012.
[7] C. Ardagna, R. Asal, E. Damiani, and Q. Vu. From security to assurance in the cloud: A survey. *ACM Computing Surveys (CSUR)*, 48(1):2:1–2:50, August 2015.
[8] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: the adversarial multi-armed bandit problem. *Annual Symposium on Foundations of Computer Science - Proceedings*, pages 322–331, 1995.
[9] R. K. Bellamy, A. Mojsilovic, S. Nagar, K. N. Ramamurthy, J. Richards, D. Saha, P. Sattigeri, M. Singh, K. R. Varshney, Y. Zhang, K. Dey, M. Hind, S. C. Hoffman, S. Houde, K. Kannan, P. Lohia, J. Martino, and S. Mehta. AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. *IBM Journal of Research and Development*, 63(4-5), 2019.
[10] B. Biggio and F. Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317 – 331, 2018.
[11] E. Black, S. Yeom, and M. Fredrikson. FlipTest: Fairness Auditing via Optimal Transport. 2019.
[12] Z. Bosnic and I. Kononenko. An Overview of Advances in Reliability estimation of Individual Predictions in Machine Learning. Technical report, University of Ljubljana, Ljubljana, 2008.
[13] J. Bryson and A. Winfield. Standardizing ethical design for artificial intelligence and autonomous systems. *Computer*, 50(5):116–119, 2017.
[14] O. Chapelle and L. Li. An empirical evaluation of Thompson sampling. In *Neural Information Processing Systems*, 2011.
[15] S. Corbett-Davies, E. Pierson, A. Feller, S. Goel, and A. Huq. Algorithmic decision making and the cost of fairness. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Part F1296:797–806, 2017.
[16] A. Dafoe. Ai governance: A research agenda. *Governance of AI Program, Future of Humanity Institute*, 2018.
[17] E. Damiani and C. Ardagna. Certified machine-learning models. In *Proc. of the 46th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2020)*, Limassol, Cyprus, January 2020.
[18] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness through awareness. *ITCS 2012 - Innovations in Theoretical Computer Science Conference*, pages 214–226, 2012.
[19] P. Gajane and M. Pechenizkiy. On Formalizing Fairness in Prediction with Machine Learning. 2017.
[20] S. Galhotra, Y. Brun, and A. Meliou. Fairness testing: Testing software for discrimination. *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Part F1301:498–510, 2017.
[21] R. Gall. Machine Learning Explainability vs Interpretability: Two concepts that could help restore trust in AI, 2018.
[22] M. Glinz. On Non-Functional Requirements. *15th IEEE International Requirements Engineering Conference*, page 361, 2007.
[23] Global Legal Research Directorate. Regulation of Artificial Intelligence in Selected Jurisdictions. Technical Report January, The Law Library of Congress, 2019.
[24] R. Guizzardi, F.-L. Li, A. Borgida, and J. Mylopoulos. An ontological interpretation of non-functional requirements. In *Proc. of the 8th International Conference on Formal Ontology in Information Systems (FOIS 2014)*, Rio de Janeiro, Brasil, September 2014.
[25] M. Hardt, E. Price, and N. Srebro. Equality of Opportunity in Supervised Learning. *International Journal of Instruction*, 2016.
[26] L. Hendricks, K. Burns, K. Saenko, T. Darrell, and A. Rohrbach. Women also Snowboard: Overcoming Bias in Captioning Models. In *Computer Vision – ECCV*. Springer, 2018.
[27] ISO/IEC and IEEE. ISO/IEC/IEEE 24765:2010 - Systems and software engineering – Vocabulary. *Iso/Iec Ieee*, 2010:410, 2010.
[28] R. P. Kesner, P. E. Gilbert, and G. V. Wallenstein. Testing neural network models of memory with behavioral experiments. *Current Opinion in Neurobiology*, 10(2):260–265, 2000.
[29] P. Khosravi, Y. Liang, Y. Choi, and G. Van den Broeck. What to expect of classifiers? rea-soning about logistic regression with missing features. In *Proc. of the 28th International Joint Conference on Artificial Intelligence*, Macao, China, June 2019.
[30] S. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 2017.
[31] D. Madras, E. Creager, T. Pitassi, and R. Zemel. Fairness through causal awareness: Learning causal latent-variable models for biased data. In *Proc. of the ACM Conference on Fairness, Accountability, and Transparency*, Atlanta, GA, USA, January 2019.
[32] D. Mairiza, D. Zowghi, and N. Nurmuliani. An investigation into the notion of non-functional requirements. *Proceedings of the ACM Symposium on Applied Computing*, pages 311–317, 2010.
[33] D. McNamara, C. Soon Ong, and R. Williamson. Costs and benefits of fair representation learning. In *Proc. of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, Honolulu, HI, USA, January 2019.
[34] N. Metropolis. The beginning of the Monte Carlo method. *Los Alamos Science*, (Special Issue dedicated to Stanislaw Ulam):125–130, 1987.
[35] C. E. Perez. Modularity, 2018.
[36] E. Raff, J. Sylvester, and S. Mills. Fair forests: Regularized tree induction to minimize model bias. In *Proc. of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, New Orleans, LA, USA, February 2018.
[37] S. Schelter, J.-H. Bose, J. Kirschnick, T. Klein, and S. Seufert. Automatically tracking metadata and provenance of machine learning experiments. In *Proc. of Workshop on ML Systems*, Long Beach, CA, USA, December 2017.
[38] M. Schmitz. Why your Models need Maintenance, 2017.
[39] S. L. Scott. Multi-armed bandit experiments in the online service economy. *Applied Stochastic Models in Business and Industry*, 31(1):37–45, 2015.
[40] M. Spiro. The FTC and AI Governance : A Regulatory Proposal The FTC and AI Governance : A Regulatory Proposal. *Seattle Journal of Technology, Environmental & Innovation Law*, 10(1), 2020.
[41] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, 1998.
[42] C. Tobler. Limits and potential of the concept of indirect discrimination. Technical report, European Network of Legal Experts in Anti-Discrimination, 2008.
[43] J. Vanschoren. Meta-Learning: A Survey, 2018.
[44] S. Verma and J. Rubin. Fairness definitions explained. *Proceedings - International Conference on Software Engineering*, pages 1–7, 2018.
[45] J. Winkler and A. Vogelsang. What does my classifier learn? A visual approach to understanding natural language text classifiers. In *Intl. Conference on Natural Language & Information Systems*, 2017.
[46] F. Yoshioka and I. N. How do engineers perceive difficulties in engineering of machine-learning systems? - Questionnaire survey. In *Joint Intl. Workshop on Conducting Empirical Studies in Industry and Intl. Workshop on Software Engineering Research and Industrial Practice*, 2019.