A Web Application Software for Causal-based Machine Learning Discrimination Estimation

Raluca Panainte, Yassine Turki, Sami Zhioua

Preprint

MMMMMM YYYY, Volume VV, II. *doi:* ———

A Web Application Software for Causal-based Machine Learning Discrimination Estimation

Raluca Panainte* INRIA, Ecole Polytechnique

Yassine Turki INRIA, Ecole Polytechnique

Sami Zhioua INRIA, Ecole Polytechnique

Abstract

Addressing the problem of fairness is crucial to safely use machine learning algorithms to support decisions with a critical impact on people's lives such as job hiring, child maltreatment, disease diagnosis, loan granting, etc. Several notions of fairness have been defined and examined in the past decade, such as statistical parity and equalized odds. The most recent fairness notions, however, are causal-based and reflect the now widely accepted idea that using causality is necessary to appropriately address the problem of fairness. The big impediment to the use of causality to address fairness, however, is the unavailability of the causal model (typically represented as a causal graph). This paper describes a software tool that implements all required steps to estimate discrimination using a causal approach, including, the causal discovery, the adjustment of the causal model, and the estimation of discrimination. The software has a web interface which makes it accessible online without any required setup on the user side.

Keywords: Causality, Machine Learning, Fairness, Discrimination.

1. Introduction

Machine learning is being used to inform decisions with critical consequences on human lives such as job hiring, college admission, loan granting, and criminal risk assessment. Unfortunately, these automated decision systems have been found to consistently discriminate against certain individuals or sub-populations, typically minorities. Because the discrimination is very often unintentional, discovering and addressing it is a challenging task. The most commonly used fairness notions are observational and rely on mere correlation between variables. For example, statistical parity Dwork, Hardt, Pitassi, Reingold, and Zemel (2012) requires that the proportion of positive outcome (e.g. granting loans) is the same for all sub-populations (e.g. male and female groups). Equal opportunity Hardt, Price, and Srebro (2016) requires that the true positive rate (TPR) is the same for all sub-populations. The main problem of

correlation-based fairness notions is that they fail to detect discrimination in presence of statistical anomalies such as Simpson's paradox Simpson (1951) and Berkson's paradox Berkson (1946); Kim and Perl (1983). A famous example of the Simpson's paradox is the gender bias in 1973 Berkley admission Bickel, Hammel, and O'Connell (1975); Loftus, Russell, Kusner, and Silva (2018). In that year, 44% of male applicants were admitted against only 34% of female applicants. While this looks like a bias against female candidates, when the same data has been analyzed by department, acceptance rates were approximately the same.

One way to address this limitation is to consider how data is generated in the first place which leads to causal-based fairness notions. Because this new breed of fairness notions is immune to statistical paradoxes, it is now widely accepted that causality is necessary to appropriately address the problem of fairness Loftus *et al.* (2018). Examples of causal-based fairness notions include total effect Pearl (2009), interventional fairness Salimi, Rodriguez, Howe, and Suciu (2019), counterfactual fairness Kusner, Loftus, Russell, and Silva (2017), counterfactual effects Zhang and Bareinboim (2018), and path-specific counterfactual fairness Chiappa (2019); Wu, Zhang, Wu, and Tong (2019).

2. Related Work

In this section, we will discuss some of the current existing softwares/libraries that are available for public use. Mostly, we will focus on AI Fairness 360 Bellamy, Dey, Hind, Hoffman, Houde, Kannan, Lohia, Martino, Mehta, Mojsilović *et al.* (2019), FAIRVIS Cabrera, Epperson, Hohman, Kahng, Morgenstern, and Chau (2019) and FairKit Johnson and Brun (2022).

AI Fairness 360 (AIF360) Bellamy *et al.* (2019) is an open source Python toolkit for algorithmic fairness developed by IBM Research. It encompasses various sets of fairness metrics (such as Statistical Parity Difference and Equal Opportunity Difference) for datasets and models, as well as explainability tools to understand these metrics. Moreover, the toolkit also includes bias mitigation algorithms that can be used for both datasets and models in order to diminish bias. Essentially, AIF360 provides Classifiers, Pre / In / Post processing Algorithms to examine bias. IBM Research also made available a web application in order to facilitate the use of its software and make it available for a wide audience.

Fairkit-learn Johnson and Brun (2022) is a Python module that combines the Machine Learning models provided by scikit-learn Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, VanderPlas, Passos, Cournapeau, Brucher, Perrot, and Duchesnay (2012) with the datasets, fairness metrics and pre/post-processing algorithms provided by AIF360.

Another fairness tool is FAIRVIS Cabrera *et al.* (2019), developed by Georgia Institute of Technology researchers. FAIRVIS is a visual analytics system that offers subgroup discovery techniques to users in order to examine the fairness of ML models. The software focuses on the generation of subgroups within a dataset in order to highlight the existence of some types of bias. FAIRVIS uses different techniques to create underperforming subgroups and locate comparable subgroups. These strategies are used in a web-based system that closely combines numerous, coordinated perspectives to assist users in discovering fairness concerns in known and undiscovered subgroups. FAIRVIS uses visualizations to support this exploration.

Gcastle Zhang, Zhu, Kalander, Ng, Ye, Chen, and Pan (2021) is a Python toolbox developed by Huawei Technologies. Gcastle allows the user to generate data from real-world datasets, in

order to learn the causal structure from the data and evaluate the learned directed graph. It implements most of the widely used causal discovery algorithms such as PC Kalisch, Mächler, Colombo, Maathuis, and Bühlmann (2012) and Greedy Equivalence Search (GES)Chickering (2002), with the addition of inserting background knowledge and post-processing to remove false discoveries. Similar to gcastle, Tetrad Ramsey, Zhang, Glymour, Romero, Huang, Ebert-Uphoff, Samarasinghe, Barnes, and Glymour (2018) also generates directed causal graphs from real world data and includes background knowledge and post-processing. Tetrad offers a wide flexibility in terms of hyper parameters of the causal discovery algorithms and graph fine tuning. As it is implemented in Java, another version of Tetrad was released in Python : Causal-LearnZheng, Huang, Chen, Ramsey, Gong, Cai, Shimizu, Spirtes, and Zhang (2023).

The graphs generated by causal discovery algorithms can be used as input by Dowhy Sharma and Kiciman (2020) in order to extract meaning to the data and compute fairness metrics from the graphs, such as Average Total Effect, Natural Direct Effect, and Indirect Effect, using different Machine Learning models. Dowhy identifies the causal structure from the graph, estimates the metrics and offers refutation algorithms in order to question the certainty of the estimation.

We developed this web application in order to consolidate what is currently used in the field and merge graph generation with the computation of fairness metrics using Machine Learning Models.

3. Software Architecture

In order to connect the causal discovery algorithms to an interactive platform, we followed the structure of an API, in other words, an Application Programming Interface. APIs are mechanisms that allow two software components to communicate with each other using a set of definitions and protocols Amazon Web Services (2023).

Figure 1 illustrates the structure of the software. For this project, the application refers to the causal discovery algorithms written in Python using related libraries such as Gcastle, Causal-learn, and DoWhy. The client side is representated by a web application written in HTML, CSS, JavaScript, and Bootstrap, an open-source CSS framework used to create a responsive front-end. These two communicate using a server written in Flask, a micro web framework in Python.

All of these elements work together with the help of a virtual environment. This approach not only facilitated an easy installation method of the web application software, but also allowed us to store data efficiently using individual user sessions.

There exists a constant flow of information between the application and the client, hence we decided to opt for sessions in order to store the user's inputs throughout their interaction with the web app. Every piece of information is stored temporarily, and not permanantely on a database. This also applies to the uploaded datasets by the user.

4. Causal Discovery Phase

In order to generate a Causal Graph from a dataset, our software offers the choice of two of the most prominent algorithms of the literature : the constraint based PC algorithm Spirtes, Glymour, and Scheines (2000) and the score based Greedy Equivalence Search algorithm Chick-

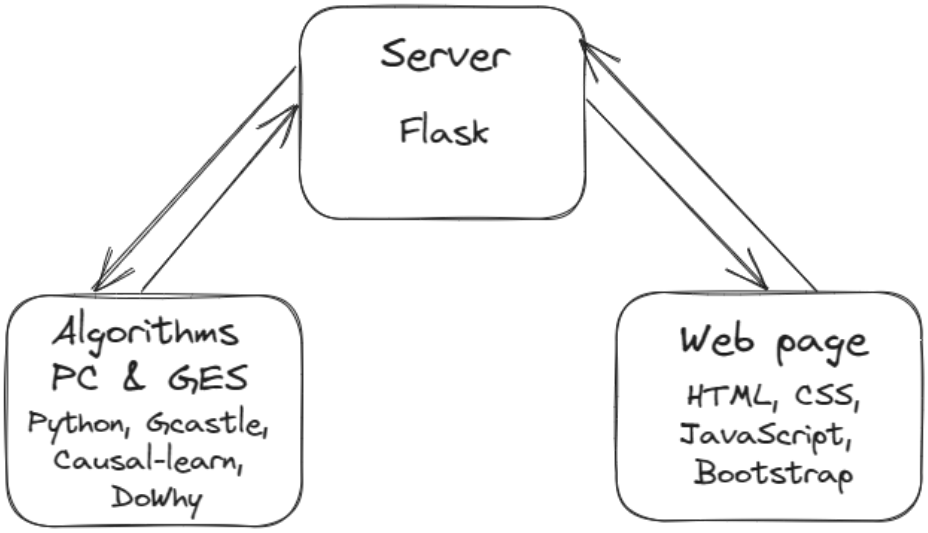

Figure 1: Structure of the software along with the languages and libraries used

ering (2002). For each algorithm, the user is prompted to choose between the **gcastle** implementation or the **causal-learn** implementation. It is worth noting that in the future, the software will feature more algorithms such as FCI and LiNGAM.

4.1. Algorithms used

- **The PC Algorithm:** The PC Algorithm Spirtes *et al.* (2000) is a constraint-based causal discovery method. It is a widely used algorithm that is reliable when the data consists of independent and identically distributed samples, satisfies the faithfulness assumption, the causal Markov condition and no latent latent confounders exist Zheng *et al.* (2023). The PC algorithm consists of forming an undirected complete graph and perform a series of independence tests based on conditioning on variables to finally return a Completed Partially Directed Acyclic Graph (CPDAG), a causal graph consisted of directed and undirected edges between nodes. We choose to represent undirected paths by dashed paths in our software, as the meaning for these paths is ambiguous. The PC algorithm allows a high adaptability to the user, by choosing the independence test that fits best the data, such as the Fisher-Z test Fisher *et al.* (1921) for linear Gaussian data, Chi-squared test and G squared test Tsamardinos, Brown, and Aliferis (2006), both suitable for discrete data. Moreover, **causal-learn** offers to use the Kernel Based Conditional Independence (KCI) Xie, Cai, Huang, Glymour, Hao, and Zhang (2020) for non parametric data and Missing Value PC Tu, Zhang, Ackermann, Mohan, Kjellström, and Zhang (2019) used in the presence of missing values in the data. Whereas both libraries support the stable version of PC Colombo and Maathuis (2013), **gcastle** allows the user to additionally select another variant of the algorithm: Parallel PC Le, Hoang, Li, Liu, Liu, and Hu (2019) . Finally, both libraries support specifying background knowledge in the form of a Tier list of variables for the PC algorithm, which

will make PC follow two rules : A node in a higher Tier cannot have a directed eedge to a node in a lower Tier, and nodes in the same Tier cannot have directed edges between them. For example, if we specify 3 tiers as follows with arbitrary variables denoted by $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$:

- Tier 1 : $\mathbf{A}, \mathbf{B}$
- Tier 2 : $\mathbf{C}$
- Tier 3 : $\mathbf{D}$

We impose on the PC Algorithm that no directed edge can go from D to C or from D to A or B. Moreover, A and B cannot have an edge connecting them. To specify a Tier, the user needs to supply a list of lists to the application. To refer to our previous example, one would specify the Tiers in this manner : [["A", "B"], ["C"], ["D"]]. It is important to note that when specifying background knowledge to the algorithm, it will learn the causal graph based on this knowledge and assumptions, as opposed to the add/delete edge functions that we will discuss a bit later in this paper.

- **The Greedy Equivalence Search Algorithm:** The Greedy Equivalence Search Algorithm (GES) Chickering (2002) is an algorithm designed to generate a causal graph by maximizing a given score function. The two widely used score funtions for GES are **Bayesian Information Criterion (BIC)** Schwarz (1978) score for Linear Gaussian Data and **Bayesian Information Criterion (BDeu)** Buntine (1991) for discrete data. Moreover, **causal-learn** provides an implementation of the General Score Huang, Zhang, Lin, Schölkopf, and Glymour (2018) for non parametric data. There are two steps to generate a causal graph with GES : First, the algorithm executes a Forward Equivalence Search (FES), in which it greedily add edges until a local maximum is reached. Then, it executes a Backward Equivalence Search (BES), in which it greedily deletes edges until a local maximum is reached. It is guaranteed that the algorithm stops at a local maximum, which is the equivalence class returned as the solution Alonso-Barba, delaOssa, Gámez, and Puerta (2013)

4.2. Graph adjustments

Upon generating the causal graph given a dataset and using a given algorithm, one might encounter causal graphs with an edge that should not exists because of an assumption that the algorithm does not have access to. Moreover, in certain scenarios, one can also notice the absence of an essential path, or edge, from two nodes. For example, we know that in a hiring decision process, the education of an applicant has a causal effect on the hiring decision. Therefore, if the algorithm did not capture the edge from the "education" node to the "hired" node, the user can directly modify the causal graph in order to include that edge.

On the Web Application, the user can modify directly the learned causal graph to add or delete an edge. This will not make the algorithm learn the graph again, but simply modify the graph structure in order to provide a more accurate graph to the libraries computing causal inference metrics.

For example, let us look at the graph generated by the PC algorithm (with gcastle) on the Adult dataset (Figure 2) As we can observe, we have a dashed path from "occupation" to

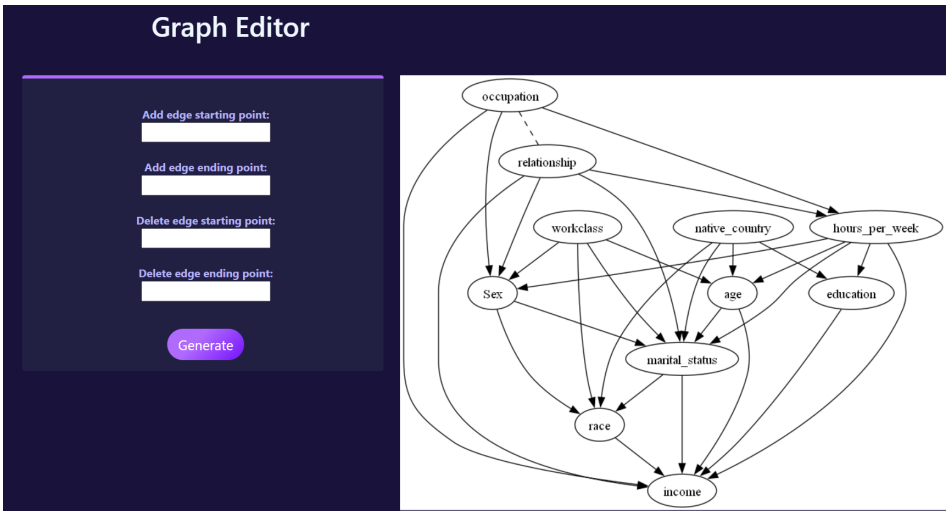

Figure 2: Causal Graph Generated by PC Algoprithm (gcastle) on Adult Dataset

"relationship", which means that the algorithm generated an undirected path between these two nodes. Let us assume that there should not be a path from "relationship" to "occupation". One would simply fill in the names of the two nodes in the delete edge section and generate a new graph, as follows (Figure 3)

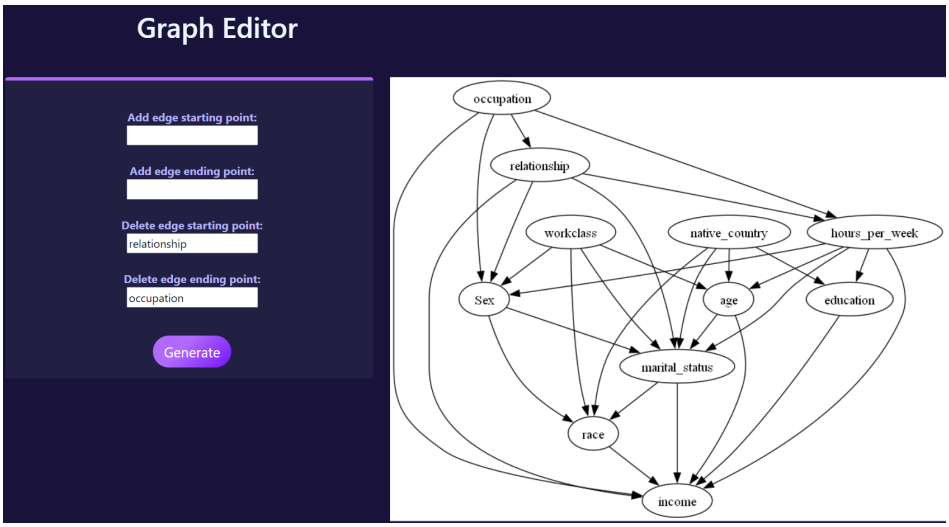

Figure 3: Causal Graph Generated by PC Algorithm (gcastle) on Adult Dataset bafter deleting the path "relationship" -> "occupation"

Now that we have deleted the edge and are satisfied with our causal graph, we can move on to the metrics page and compute causal inference metrics.

5. The causal metrics estimation phase

After generating a Causal Graph and adjusting its edges, the software will compute five widely used metrics :

- **Average Treatment Effect (ATE)**
- **Average Treatment Effect on Treated (ATT)**
- **Average Treatment Effect on Control (ATC)**
- **Natural Direct Effect (NDE)** Note: if you compute the NDE without the presence of a **mediator** between the treatment and the outcome, the software will print a warning to inform that in this situation, the NDE is equal to the ATE.
- **Indirect Effect (IE)** Note: you can only compute the IE if you have a **mediator** between the treatment and the outcome.

For the time being, we only use the **dowhy** library Sharma and Kiciman (2020). Dowhy is an opensource end-to-end library created by Microsoft for causal analysis. Dowhy's API is organized under four steps required for causal analysis :

1. **Model:** create a model based on a causal graph
2. **Identify:** identifying the causal effect from the causal graph using graph based methods.
3. **Estimate:** employ statistical methods for estimating the identified estimand
4. **Refute:** tries to refute the obtained estimate by checking robustness of the estimate.

For the purposes of our software, we only used the first three steps to compute the different metrics. Future work will involve implementing the refutation of the estimate to check for robustness.
Dowhy offers a wide range of estimators to compute the fairness metrics mentionned above (Table 2).

Estimators	**Suitable Data Types**
`Generalized Linear Model`: Compute effect of treatment using a generalized linear model. We chose to use logistic regression.	Outcome needs to be a **binary** variable.
`Instrumental Variable`: Compute effect of treatment using the instrumental variables method.	Used when we observe an instrumental variable in the graph.
`Linear Regression`: Fits a regression model for estimating the outcome using treatment(s) and confounders.	Requires a strong assumption that all relationships from treatment and covariates to the outcome are linear.
`Propensity Score Matching`: Estimate effect of treatment by finding matching treated and control units based on propensity score.	Treatment needs to be a **binary** variable. Requires the presence of a **confounder** between the treatment and the outcome.
`Propensity Score Stratification`: Estimate effect of treatment by stratifying the data into bins with identical common causes.	Treatment needs to be a **binary** variable. Requires the presence of a **confounder** between the treatment and the outcome.
`Propensity Score Weighting`: Estimate effect of treatment by weighing the data by inverse probability of occurrence.	Treatment needs to be a **binary** variable. Requires the presence of a **confounder** between the treatment and the outcome.
`Regression Discontinuity`: Estimates effect by transforming the problem to an instrumental variables problem.	
`Distance Matching`: Simple matching estimator for binary treatments based on a distance metric.	Treatment needs to be a **binary** variable.

Table 1: Dowhy Estimators and Suitable Data Types

6. User's Manual

6.1. Machine Requirements

The software was developed on the Windows operating system, thus we recommend using this OS for an optimal experience.

Before starting the software, make sure the Python version running on the machine is 3.10.11.

Then, install the virtual environment. We use a module named virtualenv, which is a tool to create isolated Python environments. virtualenv creates a folder that contains all the necessary executables to use the packages that our Python project will need.

The web application software works with the help of the following libraries and their respective versions, presented in Table 2. All of these are open-source.

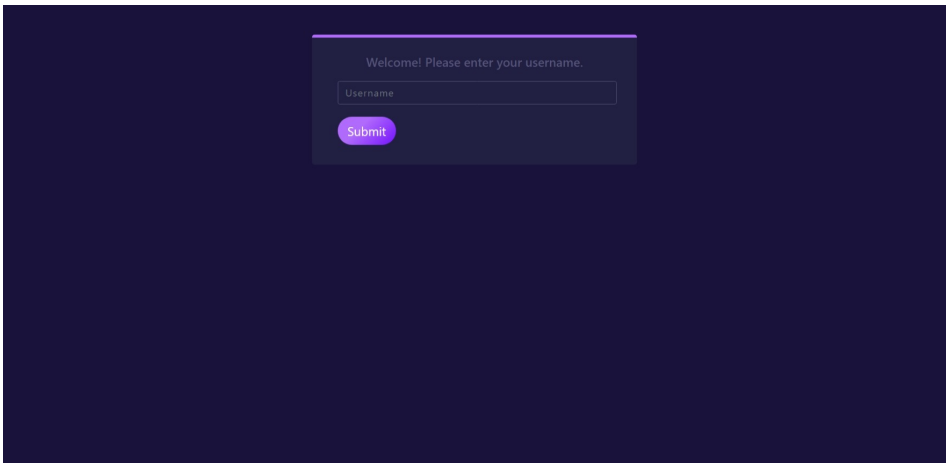

Figure 4: Login Page of the Web Application Software

Library	**Recommended Version**
`Click`	`7.0`
`Flask`	`1.1.2`
`Flask-SQLAlchemy`	`2.4.4`
`gunicorn`	`19.9.0`
`itsdangerous`	`1.1.0`
`Jinja2`	`2.11.3`
`MarkupSafe`	`1.1.1`
`SQLAlchemy`	`1.3.22`
`Werkzeug`	`1.0.1`
`gcastle`	`1.0.3`
`pydot`	`1.4.2`
`ipython`	`8.14.0`
`torch`	`2.0.1`
`dowhy`	`0.9.1`
`Flask-Session`	`0.3.2`
`pygraphviz`	`1.9`

Table 2: Libraries Used in the Development of the Software

6.2. Description of Features

Figure 4 shows the first page of our web application software, in which we ask for a username, in order to activate the individual session.

Once the user inputs their preferred username, they are redirected to the main page of our project. As seen in Figure 3, the page has a form dedicated to the generation of a causal discovery graph. The user can choose to either upload one of their own datasets, or work with one of our preloaded simplified datasets (Adult, Boston, Dutch Census, Compas, Credit Cards, Crime rates, FICO, La Londe, Law School, Police Arrest). A summary about the data contained in each dataset can be accessed by clicking on the Info button. Moreover, the

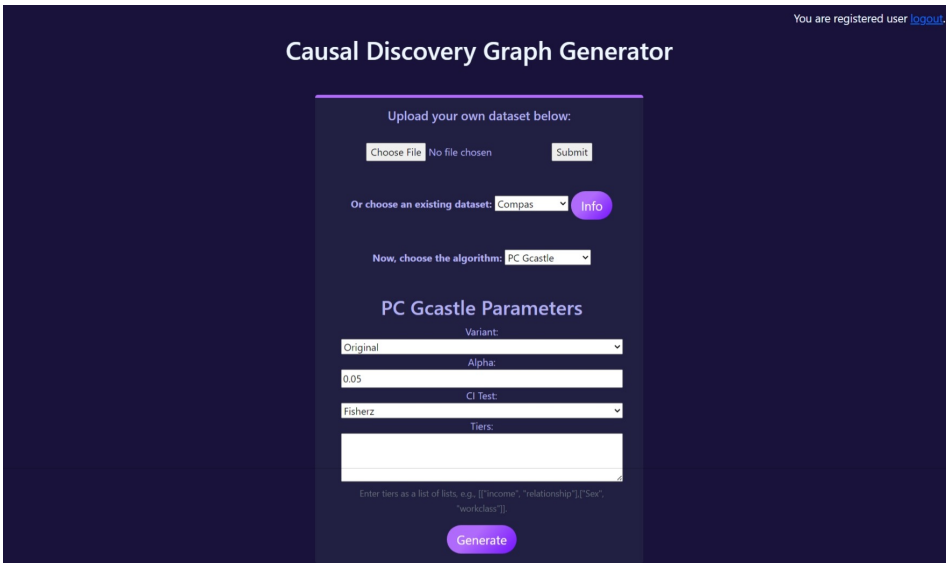

Figure 5: Main page of the Web Application Software

users also have the freedom to pick their preferred causal discovery algorithms, PC or GES, from two supported libraries, Gcastle and Causal-learn. Based on the chosen algorithm, the user has the possibility to set their preferred optional parameters before generating the causal discovery graph. For the sake of the example, we will work with the Compas dataset, the PC algorithm from Gcastle, and the standard optional parameters.

Next, we have the manipulation and estimation stage of the web application. Each page is equipped with a navigation menu at the top, which can be seen in Figure 6. The Info Dataset Page contains the data summary table specific for the used dataset. The Editor Page allows the user to add or remove edges, one at a time, from the initial graph. Finally, the Metrics Page contains the parameters form for the calculation of ATT, ATC, ATE, Direct Effect and Indirect Effect. In the case of incorrect inputs, a pop-up message will alert the user of the possible errors. Otherwise, the metrics will be displayed under the Generate button. For the chosen example, we introduced the following values: Treatment -> age, Outcome -> two year recid, Estimator to compute Direct Effect and Indirect Effect -> linear regression estimator, Estimator to commpute ATT, ATC, ATE -> Linear Regression. In Figure 6, we can see that we received back a part of the metrics, as well as, some information about the errors. The metrics ATT, ATC, ATE were computed with no problem, however no mediator was found between the treatment and the outcome, thus Direct and Indirect Effect could not be calculated. All of this runs with the help of the algorithms provided by the DoWhy library.

7. Challenges and potential improvements

7.1. Challenges

During the development of our project, we have encountered multiple challenges. First, we needed to find a way to connect the causal discovery algorithms to an "easy-to-use" platform. Thus, we opted for using an API and a web application.

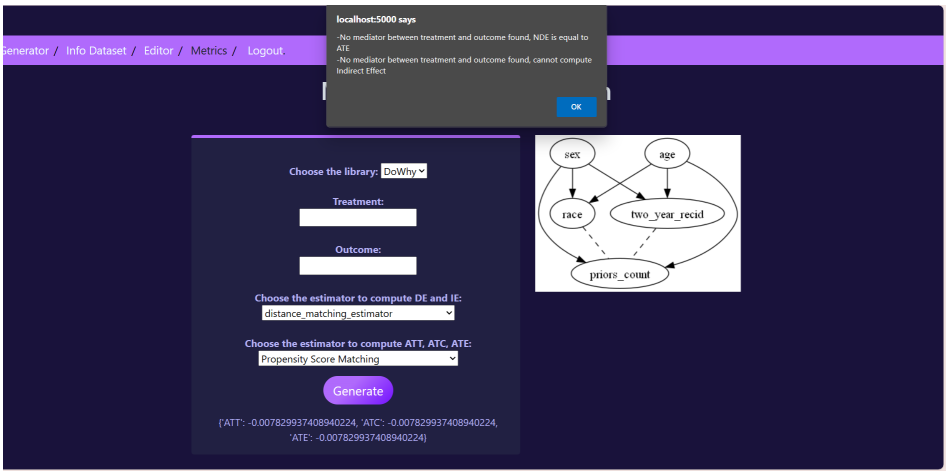

Figure 6: Measuring Discrimination Page

Another challenge we faced was finding the right versions for Python and the libraries in order for all to work together. We need to mention that the library pygraphviz might need special attention when it comes to its installation. We faced problems with this aspect on the Windows operating system.

Next up, we needed to decide upon an efficent way to store all the user's inputs throughout the usage of the software. Hence, we implemented individual sessions to temporary store all the information we needed. Moreover, this approach enables the simultaneous use of the web app between multiple users, which is useful for a future Internet-public implementation.

Lastly, the software might run slower for bigger datasets, extending the waiting time for the generation of the causal discovery graphs.

7.2. Potential Improvements

Our web application software can easily be made public as a website on the Internet, for which user accounts could be implemented, in order to not only keep count of the people using the platform, but to also allow users to save and review previous sessions.

The upload feature can benefit from improvements, when it comes to the storing of uploaded datasets. We consider that the most optimal method for this would be a cloud-based approach.

This software can also be implemented in larger scale Machine Learning projects as a tool to benefit the estimation of potential discriminants.

Acknowledgments

This work was supported by the European Research Council (ERC) project HYPATIA under the European Union's Horizon 2020 research and innovation programme. Grant agreement n. 835294.

References

Alonso-Barba JI, delaOssa L, Gámez JA, Puerta JM (2013). "Scaling up the Greedy Equivalence Search algorithm by constraining the search space of equivalence classes." *International Journal of Approximate Reasoning*, **54**(4), 429–451. ISSN 0888-613X. doi:https://doi.org/10.1016/j.ijar.2012.09.004. Eleventh European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2011), URL https://www.sciencedirect.com/science/article/pii/S0888613X12001636.

Amazon Web Services (2023). "What is API?" https://aws.amazon.com/what-is/api/#:~:text=on%20your%20phone.-. Accessed on: September 25th,.

Bellamy RK, Dey K, Hind M, Hoffman SC, Houde S, Kannan K, Lohia P, Martino J, Mehta S, Mojsilović A, *et al.* (2019). "AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias." *IBM Journal of Research and Development*, **63**(4/5), 4–1.

Berkson J (1946). "Limitations of the application of fourfold table analysis to hospital data." *Biometrics Bulletin*, **2**(3), 47–53.

Bickel PJ, Hammel EA, O'Connell JW (1975). "Sex bias in graduate admissions: Data from Berkeley." *Science*, **187**(4175), 398–404.

Buntine W (1991). "Theory Refinement on Bayesian Networks." In BD D'Ambrosio, P Smets, PP Bonissone (eds.), *Uncertainty Proceedings 1991*, pp. 52–60. Morgan Kaufmann, San Francisco (CA). ISBN 978-1-55860-203-8. doi:https://doi.org/10.1016/B978-1-55860-203-8.50010-3. URL https://www.sciencedirect.com/science/article/pii/B9781558602038500103.

Cabrera ÁA, Epperson W, Hohman F, Kahng M, Morgenstern J, Chau DH (2019). "FairVis: Visual analytics for discovering intersectional bias in machine learning." In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 46–56. IEEE.

Chiappa S (2019). "Path-specific counterfactual fairness." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 7801–7808.

Chickering DM (2002). "Optimal structure identification with greedy search." *Journal of machine learning research*, **3**(Nov), 507–554.

Colombo D, Maathuis MH (2013). "Order-independent constraint-based causal structure learning." 1211.3295.

Dwork C, Hardt M, Pitassi T, Reingold O, Zemel R (2012). "Fairness through awareness." In *Proceedings of the 3rd innovations in theoretical computer science conference*, pp. 214–226.

Fisher RA, *et al.* (1921). "014: On the" Probable Error" of a Coefficient of Correlation Deduced from a Small Sample."

Hardt M, Price E, Srebro N (2016). "Equality of opportunity in supervised learning." In *Advances in neural information processing systems*, pp. 3315–3323. Spain.

Huang B, Zhang K, Lin Y, Schölkopf B, Glymour C (2018). "Generalized Score Functions for Causal Discovery." KDD '18, p. 1551–1560. Association for Computing Machinery, New York, NY, USA. ISBN 9781450355520. doi:10.1145/3219819.3220104. URL https://doi.org/10.1145/3219819.3220104.

Johnson B, Brun Y (2022). "Fairkit-learn: a fairness evaluation and comparison toolkit." In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, pp. 70–74.

Kalisch M, Mächler M, Colombo D, Maathuis MH, Bühlmann P (2012). "Causal Inference Using Graphical Models with the R Package pcalg." *Journal of Statistical Software*, **47**(11), 1–26. doi:10.18637/jss.v047.i11.

Kim H, Perl J (1983). "A computational model for combined causal and diagnostic reasoning in inference systems." In *Proceedings of the 8th International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan-Kaufmann, San Mateo, CA.

Kusner MJ, Loftus J, Russell C, Silva R (2017). "Counterfactual fairness." *Advances in neural information processing systems*, **30**.

Le TD, Hoang T, Li J, Liu L, Liu H, Hu S (2019). "A Fast PC Algorithm for High Dimensional Causal Discovery with Multi-Core PCs." **16**(5), 1483–1495. doi:10.1109/tcbb.2016.2591526. URL https://doi.org/10.1109%2Ftcbb.2016.2591526.

Loftus JR, Russell C, Kusner MJ, Silva R (2018). "Causal reasoning for algorithmic fairness." *arXiv preprint arXiv:1805.05859*.

Pearl J (2009). *Causality*. Cambridge university press.

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, VanderPlas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2012). "Scikit-learn: Machine Learning in Python." *CoRR*, **abs/1201.0490**. 1201.0490, URL http://arxiv.org/abs/1201.0490.

Ramsey JD, Zhang K, Glymour M, Romero RS, Huang B, Ebert-Uphoff I, Samarasinghe S, Barnes EA, Glymour C (2018). "TETRAD—A toolbox for causal discovery." In *8th International Workshop on Climate Informatics*.

Salimi B, Rodriguez L, Howe B, Suciu D (2019). "Interventional fairness: Causal database repair for algorithmic fairness." In *Proceedings of the 2019 International Conference on Management of Data*, pp. 793–810.

Schwarz G (1978). "Estimating the dimension of a model." *The Annals of Statistics*, p. 461–464.

Sharma A, Kiciman E (2020). "DoWhy: An End-to-End Library for Causal Inference." 2011.04216.

Simpson EH (1951). "The interpretation of interaction in contingency tables." *Journal of the Royal Statistical Society: Series B (Methodological)*, **13**(2), 238–241.

Spirtes P, Glymour CN, Scheines R (2000). *Causation, prediction, and search*. MIT press.

Tsamardinos I, Brown L, Aliferis C (2006). "The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm." *Machine Learning*, **65**, 31–78. doi:10.1007/s10994-006-6889-7.

Tu R, Zhang C, Ackermann P, Mohan K, Kjellström H, Zhang K (2019). "Causal Discovery in the Presence of Missing Data." In K Chaudhuri, M Sugiyama (eds.), *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 1762–1770. PMLR. URL https://proceedings.mlr.press/v89/tu19a.html.

Wu Y, Zhang L, Wu X, Tong H (2019). "Pc-fairness: A unified framework for measuring causality-based fairness." In *Advances in Neural Information Processing Systems*, pp. 3404–3414.

Xie F, Cai R, Huang B, Glymour C, Hao Z, Zhang K (2020). "Generalized Independent Noise Condition for Estimating Latent Variable Causal Graphs." 2010.04917.

Zhang J, Bareinboim E (2018). "Fairness in decision-making—the causal explanation formula." In *Proceedings of the... AAAI Conference on Artificial Intelligence*.

Zhang K, Zhu S, Kalander M, Ng I, Ye J, Chen Z, Pan L (2021). "gCastle: A Python Toolbox for Causal Discovery." 2111.15155.

Zheng Y, Huang B, Chen W, Ramsey J, Gong M, Cai R, Shimizu S, Spirtes P, Zhang K (2023). "Causal-learn: Causal Discovery in Python." 2307.16405.

Affiliation:

Raluca Panainte
INRIA, Ecole Polytechnique
Palaiseau, France
E-mail: panainte@polytechnique.edu

Yassine Turki
INRIA, Ecole Polytechnique
Palaiseau, France
E-mail: yassine.turki@polytechnique.edu

Sami Zhioua
INRIA, Ecole Polytechnique
Palaiseau, France
E-mail: zhioua@lix.polytechnique.fr

MMMMMM YYYY, Volume VV, II
doi:-----
Submitted: yyyy-mm-dd
Accepted: yyyy-mm-dd