ViCE: Visual Counterfactual Explanations for Machine Learning Models

Oscar Gomez[*]
Steffen Holter[*]
{oscar.gomez,steffen.holter}@nyu.edu
New York University Abu Dhabi

Jun Yuan
Enrico Bertini
{junyuan,enrico.bertini}@nyu.edu
NYU Tandon School of Engineering

Abstract
The continued improvements in the predictive accuracy of machine learning models have allowed for their widespread practical application. Yet, many decisions made with seemingly accurate models still require verification by domain experts. In addition, end-users of a model also want to understand the reasons behind specific decisions. Thus, the need for interpretability is increasingly paramount. In this paper we present an interactive visual analytics tool, *ViCE*, that generates counterfactual explanations to contextualize and evaluate model decisions. Each sample is assessed to identify the minimal set of changes needed to flip the model's output. These explanations aim to provide end-users with personalized actionable insights with which to understand, and possibly contest or improve, automated decisions. The results are effectively displayed in a visual interface where counterfactual explanations are highlighted and interactive methods are provided for users to explore the data and model. The functionality of the tool is demonstrated by its application to a home equity line of credit dataset.

CCS Concepts
• **Human-centered computing → Visualization systems and tools**; • **Computing methodologies →** *Machine learning*.

Keywords
Machine learning, interpretability, explainability, counterfactual explanations, data visualization

ACM Reference Format:
Oscar Gomez, Steffen Holter, Jun Yuan, and Enrico Bertini. 2020. *ViCE*: Visual Counterfactual Explanations for Machine Learning Models. In *25th International Conference on Intelligent User Interfaces (IUI '20), March 17–20, 2020, Cagliari, Italy.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3377325.3377536

1 Introduction
The accessibility of high performing machine learning models has resulted in their integration into various applications pertaining to complex and high-risk data. Even in industries such as financial services and health care where the assimilation of predictive models has been slower due to the associated risks, machine learning is now seeing rapid adoption. However, simple accuracy measures that are used to describe models often fail to describe deeper flaws such as hidden biases and false generalizations. In high-risk situations such as cancer diagnosis or fiscal lending such oversight cannot be accommodated and can result in detrimental consequences.

In this paper we present *ViCE*[1], a novel design for an explainable machine learning visual analytics tool and its evaluation using a case study. The tool is built to describe a machine learning model by breaking down individual predictions. This caters directly to our end-user, who we envision as being the client-facing person trying to better understand predictions made by the model. This could include doctors inferring why a patient is predicted as high risk for diabetes or admissions officers looking into why a particular candidate was rejected. Although *ViCE* could also be useful for model developers, that is not the tool's primary purpose. The analysis is driven by the introduction of counterfactual explanations. Our tool relies on a new algorithm for calculating counterfactuals that is not limited to binary variables and is intended for use with tabular numerical data. Furthermore, we have created the first visual interface that is able to display these explanations effectively and coherently. *ViCE* is supplemented with functionality that contextualizes the targeted sample with regards to the rest of the dataset. The combination of these features guarantees that the resulting interface does not only clarify the model's decision but can also be used to pinpoint bias and undesired behaviour.

For the targeted end-user each explanation provides actionable suggestions that can help adjust the model's prediction. In other words, the tool establishes what changes are required to alter their current state. For example, it could be used by a loan-officer looking to get a previously rejected application approved.

2 Background and Related Works
The problem of machine learning model interpretability and explanation has been recognized by many researchers and practitioners. Previous works [1–4, 6, 10, 15, 17] provide an overview of methods, opportunities and challenges in this area.

To interpret a machine learning model, methods vary according to the category of models. Machine learning models are often categorized into two classes: *white-box* and *black-box*. White-box models are those intrinsically interpretable models, where the logic of making a decision is transparent and intelligible (e.g. *decision trees*, *linear regression*, etc.) [15]; while black-box models tend to have complex structures and are hard to understand (e.g., *neural*

[*]Both authors contributed equally to this research.

[1]https://github.com/5teffen/ViCE

networks, *ensemble models*, etc.). In this paper, we introduce an explanation algorithm that is model independent, that is, the method works with any model without having access to its internal logic.

Generally, model explanations can be categorized as *local* or *global*. Local explanations try to explain how a decision is made for a specific instance, while global explanation methods refer to showing the overall logical structure of a model. Some approaches such as LIME [18], and SHAP [11], focus on generating a weight for each feature as its contribution to the final decision. Others provide explanations through a counterfactual, where the explanation consists of the minimal set of changes to the feature values that allows the prediction for the instance to change to a different outcome. For example, finding the smallest feature perturbation that would change the prediction of a loan application from rejected to approved. Wachter *et al.* [22] provide a general framework for counterfactual generation using stochastic optimization, while Ustun *et al.* [21] present an approach specific to linear classifiers.

As for the presentation of model explanations, visualization has been increasingly used to support the understanding, debugging, verification, and refinement of machine learning models.

As a black-box explanation tool, *ViCE* does not rely on the internal logic of the model, but is designed to let users explore the relationships between inputs (instances) and outputs (predictions). Some existing visual analytic systems follow the black-box approach. For example, Manifold [24] enables the comparison of data distribution at different levels of granularity; RuleMatrix [14] visualizes extracted rules for a given model; iForest [25] and Ensemble Matrix [19] attempt to explain ensemble models.

Similarly to our work, the What-If Tool (WIT) [23] tries to answer a what-if question. WIT shows how model predictions change after the inference of data, while our tool, visualizes how to inference from data in order to change a prediction into other classes.

Likewise, Rivelo [20] and the Workflow for Visual Diagnostics proposed by Krause *et al.* [8] also provide a solution to a counterfactual question of how to change data to achieve a target class for black-box models. However, their solution adapts an algorithm [12] originally designed for text documents and works only on binary inputs. Our work extends this algorithm to situations pertaining to continuous numerical data.

3 ViCE

The main goal of the proposed tool is to support understanding of individual predictions through counterfactual explanations and to provide an intuitive visual representation for them. More precisely, our objective is to show, for a given instance, what is the minimal set of changes that is required to change the prediction. In our case, we focus on numeric features, therefore the tool has to provide two pieces of information: (1) which features need to change and (2) the extent to which they have to change.

ViCE was designed through an iterative design process. We analyzed published work to compile a list of questions end-users may want to answer when using counterfactual explanations and designed several solutions. The final result is the tool we present in the paper. The following list summarizes the desired functionality that we deemed essential to support our goal.

Q1 **Data distribution** - How do the values of the instance compare to those across the rest of the dataset?
Example: If a student has a GRE score of 320, how does it compare to the scores of their peers?

Q2 **Relevant features** - Which features have the most considerable effect on the model's prediction?
Example: Identifying what variables in a patient's blood work are significant contributors to a negative diagnosis.

Q3 **Possible changes** - Are there changes that could alter the model's current prediction?
Example: If an applicant was rejected for a loan, what changes in their profile would be required for the application to be accepted?

Q4 **Actionable changes** - Is it possible to change only a subset of *actionable* features to change the model's prediction?
Example: If a graduate school applicant knows certain features cannot be changed such as Gender or Age, is it possible to generate an alternative explanation without altering these features?

In the following two sections we first describe the algorithm developed in detail and then describe the visual solution designed for communicating information about the counterfactual explanations.

3.1 Counterfactual Algorithm

The counterfactual algorithm aims to find the minimal set of changes needed to change the model's output. We implement a simple heuristic algorithm to find changes that are at the same time interpretable (minimal set of features) and feasible (minimal amount of change); characteristics that are crucial for user-friendly explanations [13].

In order to extend the algorithm proposed in [12], the entire dataset is discretized by fitting a Gaussian on each of the features and splitting the values into n bins such that the middle n-2 capture four standard deviations from the mean, and the extreme bins capture data points beyond this. The algorithm greedily moves feature values across the bins until the predicted class is changed, or until the pre-defined constraints (no more than w features are changed in a single explanation and no feature value is moved across more than l bins) are reached.

The algorithm starts with the original feature values of the instance to explain, and it is given an arbitrary set of unlocked features which can be acted upon. In each iteration, it independently moves the value in each of the unlocked features to the bins above and below the current one and chooses the one eliciting the largest change in the model's output (in the direction of the opposite category). It then takes the maximum change across all the unlocked features and uses this as the input for the next iteration. This greedy procedure continues until the modified instance crosses the model's decision boundary or until the constraints can no longer be satisfied.

3.2 Visual Interface

In Fig. 1, we present the explanation for an instance in a diabetes dataset [7]. For demonstration purposes, a support vector machine is used. The individual explanation view shows a detailed summary regarding the model's decision for a single data point while also giving context to the values relative to the rest of the dataset. The percentage bar (Fig. 1①) is used to indicate the exact prediction made by the model, thereby quantifying the strength of a prediction beyond the binary result. In our solution, any model prediction value greater than 50% is classified as *positive* and shown in green while all other decisions are classified as *negative* and shown in red. The correctness of the prediction is also presented (Fig. 1②). It is

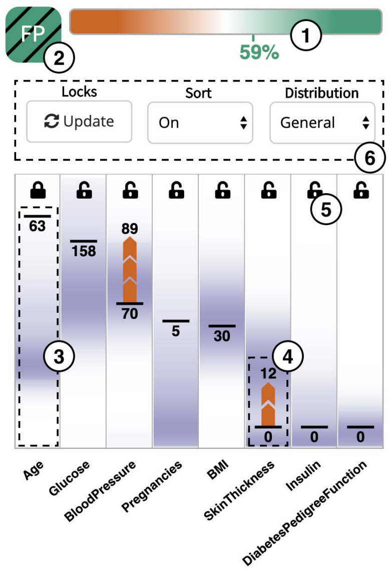

Figure 1: Demonstrating a single local explanation using a diabetes dataset. ① model's predicted probability, ② classification correctness of the model's prediction, ③ frequency density distribution and the feature value for the given instance, ④ counterfactual explanation, ⑤ locking functionality, ⑥ lock, sort, and distribution toggles.

important to note that knowledge of the ground truth is not a requirement as this information is unavailable in many real use-cases. However, when available, knowing whether the model's decision is correct helps categorize the sample point as either an example of the model's desired operation or of its potential shortcomings.

The main part of the interface separates the data by features and displays their numerical values. These values are positioned relative to the distribution for that feature across the entire dataset (Fig. 1③). Each attribute column is also supplemented with a density distribution visualisation. Based on the opacity of the purple background, the frequency of occurrence at that position can be analysed. For example, in this explanation the patient's age and glucose levels are clearly above the average. This information might suggest that these factors are contributing to the false positive prediction. By default, the tool displays the density distribution based on all the data points, however, the user has the option to map the densities based on points with *positive* or *negative* target values (*Distribution* selection in Fig. 1⑥). In other words, it is possible to see how the sample under consideration compares with known *positive* or *negative* predictions. This effectively helps contextualize the values of the sample and highlight the features with singular values.

The local view will also display counterfactual explanations if the conditions set by the algorithm are fulfilled (Fig. 1④). Arrow shaped polygons are used to exhibit a single increment in the bins used to discretize the tabular data. The current value and the suggested new level are both shown numerically for clearer reading and detail. The color of these symbols is based on the binary decision made by the model. For a *positive* prediction red arrows are used to show what changes would result in the decision becoming *negative*, while green arrows are used for *negative* instances as indicators for a *positive* change. In this example, if two features had greater values then the patient would no longer be considered at risk for diabetes. Thus, according to the model, in order for this patient to become healthy they would need to slightly increase their blood pressure and skin thickness levels. This clearly exemplifies how the end-user benefits from having information that extends beyond a binary classifier.

To guarantee versatility, a locking function is available to remove certain attributes from consideration (Fig. 1⑤). This can be useful if a user has certain features they deem unable to change or modify. In this case the age feature can be treated as unfeasible to change and can be locked using the icon. In most cases, the counterfactuals are elicited in examples with prediction percentages nearer to the cutoff threshold of 50%. This is due to the fact that samples in which the model predicts a very high or low percentage usually cannot be flipped by implementing a few changes and would require larger modifications than those allowed by the algorithm.

The tool also has a sorting option (*Sort* in Fig. 1⑥). Toggling the sort button orders the features based on their standardized values. In this way users can quickly identify singular feature values that are considerably above or below the average for a feature. The sort functionality can be very effective in comparing monotonic features and highlighting key attributes.

Notice that the lack of a counterfactual explanation does not mean that no information can be derived from the visualization. Comparing the data values to the density distributions depicted by the shaded area can help identify anomalies and derive hypotheses on why and how the model produces a given prediction. It is also worth noticing that the visualization interface can accommodate any other counterfactual generation methods [9, 16, 21, 22].

3.3 Implementation

ViCE is built as a Flask web application with the back-end running on Python. The visualisations are created using D3 and JavaScript. With versatility in mind the tool accepts any binary classification dataset in a CSV format. A default SVM model is trained with scikit-learn, however, the program also accommodates custom input models as long as probability prediction methods are provided. The data is processed in real time to accommodate customized end-user inputs. For our implementation we split feature values across $n = 10$ bins and set $w = l = 5$ for the algorithm constraints.

4 Case Study

To demonstrate how *ViCE* can help with ML explanation, we showcase its use with the Home Equity Line of Credit (HELOC) dataset. The design goals set out in Section 3 are used to evaluate the performance of the tool and are referenced directly in the use case.

The HELOC dataset was released as part of the FICO xML challenge [5]. It is comprised of applications made by real homeowners in attempts to get a credit line from the bank. The target is to predict the binary variable *Risk Performance* where *bad* indicates that a consumer was at least 90 days past due once and *good* that they never were. Some initial testing revealed that the *External Risk Estimate* feature had a very strong correlation with the target class. Since this feature is not directly actionable it is initialized as locked. However, the user retains the ability to unlock it if desired.

To simulate the end-user experience an arbitrary client was picked. The chosen instance seen in Fig. 2 shows a negative model

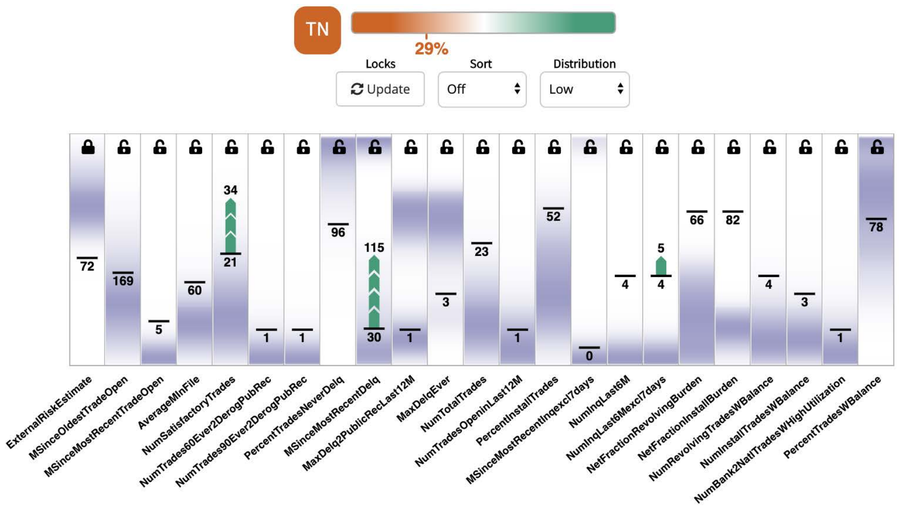
Locks: Update
Sort: Off
Distribution: Low

Figure 2: Use case for a sample from the HELOC dataset

prediction with 29% probability and the TN label on the upper left indicates that the model prediction matches the ground truth. Setting the data point into context using the density distributions reveals that there is considerable variation from the dataset averages in a number of the features **(Q1)**. Toggling the sort functionality helps identify the most singular features to be *Net Fraction Revolving Burden*, *Percentage Trades Never Delinquent* and *Months Since Most Recent Delinquency*. The *External Risk Estimate* score is also considerably lower than the average. Since this sample elicits a number of uncommon values there is a certain degree of subjectivity involved with identifying the features that should be of particular interest. Redrawing the distribution for *negative* samples shows the frequency density distribution for other known *negative* points. Using this view it is possible to confirm that the features identified above are significantly different, even in the context of other poorly performing samples **(Q2)**. Furthermore, changing between the general, positive, and negative density frequency distribution views gives an indication of the monotonicity of the features.

To understand what changes would be required by the user to receive a positive prediction we can examine the counterfactuals. This sample cannot be considere an edge case, however, since the percentage prediction is not too low at 29% and there still exist combinations of changes that would result in the model flipping the decision **(Q3)**. Yet, as expected, these changes are significant. The tool suggests that a sizable increase in both the *Number of Satisfying Trades* and *Months Since Most Recent Delinquency* and a small rise in the *Number of Inquiries in the Last 6 Months excluding last 7 days* would be sufficient. Intuitively, all of these changes are manageable, but if the user was in a rush to get their credit line approved the time based features might not be feasible **(Q4)**. Locking these attributes and reloading the explanation generates a new explanation with changes in *Average Months in File*. Since this is also time dependent it was subsequently locked as well. With these limitations imposed by the user, the algorithm is no longer able to identify a way in which the decision can be changed within the pre-defined constraints. Therefore, it is apparent that the model weights features with time variables highly in its decision making for this instance.

For further exploration, unlocking the *External Risk Estimate* variable instantly demonstrates the strength of its correlation with the model decision. The explanation now suggests large changes in *External Risk Estimate* as the optimal way of flipping the decision.

5 Limitations

This work is the first step in our goal to provide full end-user oriented model explanations. The tool currently has certain limitations. For example, the algorithm cannot effectively handle categorical features. Possible solutions might involve presetting a search path or performing a brute force analysis of features that are known to be categorical. In addition, the tool does not extend to multi-class classification or other contexts such as image classification. The visualization itself can realistically display a maximum of around 30 features. However, larger datasets can be accommodated by utilizing the sorting feature and only displaying the top *k* features.

6 Conclusions and Future Work

In this paper, we presented *ViCE* – a novel way for the end-user to gain insight into model predictions through counterfactual explanations. For each sample the minimal set of changes needed to alter the decision was shown. Interacting with the interface allows customizing the explanation according to the user's requirements. A use case was chronicled by applying the tool on a loan dataset. To the best of our knowledge this tool is the first in visualising counterfactuals for non-binary data. While already providing increased model interpretability, the modular black-box based nature of the tool allows for a seamless integration of improvements such as including different methods to generate counterfactuals, or providing users with a set of alternatives to the displayed counterfactual explanation.

Future work will aim to introduce increased interactivity for the UI. This would include adding an option to view the impact of custom changes inputted by the user. To improve the visualization, additional explanation methods can be integrated. For example, customizing the sorting functionality to order the features according to their local importance magnitudes could provide a way to corroborate the insights gained from the counterfactuals. Finally, extending the tool to a global scale through the aggregation of instance explanations could further increase its usefulness for model developers.

Acknowledgements

This work was partially supported by the DARPA D3M program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA. We thank the nexquare team, especially Haseeb Ahmed and Sarah Ameen, for providing valuable feedback and sharing their unique perspective, having led multiple data science engagements in the education domain. We also thank the anonymous reviewers for their valuable feedback.

References

[1] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). *IEEE Access* 6 (2018), 52138–52160.

[2] Or Biran and Courtenay Cotton. 2017. Explanation and justification in machine learning: A survey. In *IJCAI-17 workshop on explainable AI (XAI)*, Vol. 8.

[3] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. 2019. Machine Learning Interpretability: A Survey on Methods and Metrics. *Electronics* 8, 8 (2019), 832.

[4] Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608* (2017).

[5] FICO. 2018. Explainable Machine Learning Challenge. https://community.fico.com/s/explainable-machine-learning-challenge?tabset-3158a=2.

[6] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2019. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)* 51, 5 (2019), 93.

[7] R S Johannes. 1988. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. *Johns Hopkins APL Technical Digest* 10 (1988), 262–266.

[8] Josua Krause, Aritra Dasgupta, Jordan Swartz, Yindalon Aphinyanaphongs, and Enrico Bertini. 2017. A workflow for visual diagnostics of binary classifiers using instance-level explanations. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE, 162–172.

[9] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. 2017. Inverse Classification for Comparison-based Interpretability in Machine Learning. *arXiv preprint arXiv:1712.08443* (2017).

[10] Zachary C Lipton. 2016. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490* (2016).

[11] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*. 4765–4774.

[12] David Martens and Foster Provost. 2013. Explaining data-driven document classifications. (2013).

[13] Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence* 267 (2019), 1–38.

[14] Yao Ming, Huamin Qu, and Enrico Bertini. 2019. RuleMatrix: Visualizing and Understanding Classifiers with Rules. *IEEE transactions on visualization and computer graphics* 25, 1 (2019), 342–352.

[15] Christoph Molnar. 2019. *Interpretable Machine Learning*. https://christophm.github.io/interpretable-ml-book/.

[16] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT* ’20)*. Association for Computing Machinery, New York, NY, USA, 607–617. https://doi.org/10.1145/3351095.3372850

[17] Forough Poursabzi-Sangdeh, Daniel G Goldstein, Jake M Hofman, Jennifer Wortman Vaughan, and Hanna Wallach. 2018. Manipulating and measuring model interpretability. *arXiv preprint arXiv:1802.07810* (2018).

[18] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 1135–1144.

[19] Justin Talbot, Bongshin Lee, Ashish Kapoor, and Desney S Tan. 2009. EnsembleMatrix: interactive visualization to support machine learning with multiple classifiers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1283–1292.

[20] Paolo Tamagnini, Josua Krause, Aritra Dasgupta, and Enrico Bertini. 2017. Interpreting black-box classifiers using instance-level visual explanations. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*. ACM, 6.

[21] Berk Ustun, Alexander Spangher, and Yang Liu. 2019. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. ACM, 10–19.

[22] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GPDR. *Harv. JL & Tech.* 31 (2017), 841.

[23] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. 2019. The What-If Tool: Interactive Probing of Machine Learning Models. *IEEE transactions on visualization and computer graphics* (2019).

[24] Jiawei Zhang, Yang Wang, Piero Molino, Lezhi Li, and David S Ebert. 2018. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 364–373.

[25] Xun Zhao, Yanhong Wu, Dik Lun Lee, and Weiwei Cui. 2019. iForest: Interpreting Random Forests via Visual Analytics. *IEEE transactions on visualization and computer graphics* 25, 1 (2019), 407–416.