Counterfactual Examples for Data Augmentation: A Case Study

MGM Mehedi Hasan and Douglas A. Talbert
Department of Computer Science
Tennessee Tech University
Cookeville, USA
mmehediha42@tntech.edu, dtalbert@tntech.edu

Abstract

Counterfactual explanations are gaining in popularity as a way of explaining machine learning models. Counterfactual examples are generally created to help interpret the decision of a model. In this case, if a model makes a certain decision for an instance, the counterfactual examples of that instance reverse the decision of the model. The counterfactual examples can be created by craftily changing particular feature values of the instance. Though counterfactual examples are generated to explain the decision of machine learning models, in this work, we explore another potential application area of counterfactual examples, whether counterfactual examples are useful for data augmentation. We demonstrate the efficacy of this approach on the widely used "Adult-Income" dataset. We consider several scenarios where we do not have enough data and use counterfactual examples to augment the dataset. We compare our approach with Generative Adversarial Networks approach for dataset augmentation. The experimental results show that our proposed approach can be an effective way to augment a dataset.

Introduction

There has been a desire for explanations of how complex computer systems make decisions for quite some time. The need for explanations can be dated back to some of the earliest work on expert systems (Buchanan and Shortliffe 1984). Explanations are critical for machine learning (ML), especially as machine learning-based systems are being used to inform decisions in societally critical domains such as finance, healthcare, education, and criminal justice. However, most explanation methods depend on an approximation of the ML model to create an interpretable explanation. For example, consider a person who applied for a loan and was rejected by the loan approval algorithm of a financial company. Typically, the company may provide an explanation as to why the loan was rejected, for example, due to "poor credit history." However, such an explanation does not necessarily provide the person with sufficient information regarding what they need to do to improve their chances of being approved in the future. Critically, the most important feature may not be enough to flip the decision of the algorithm and, in practice, may not even be changeable such as gender or race.

Wachter et al. (Wachter, Mittelstadt, and Russell 2017) argue that there are three important aims for explanations: (1) to inform and help the person understand why a particular decision was reached, (2) to provide grounds to contest the decision in the case of an undesirable outcome, and (3) to understand what would need to change in order to get a desirable result in the future, based on the current decision making model. A counterfactual explanation of a prediction describes the smallest change to the feature values that changes the prediction to a predefined output, and it can be a good candidate to fulfill the three aims proposed by Wachter et. al. In interpretable machine learning, counterfactual explanations can be used to explain predictions of individual instances. In this paper, we use the terms counterfactuals and counterfactual examples interchangeably.

Counterfactual examples are increasingly seen as enhancing the autonomy of people subject to automated decisions by allowing people to navigate the rules that govern their lives (Barocas, Selbst, and Raghavan 2020). This helps people recognize whether to contest the decision making process and facilitates direct oversight and regulation of algorithms (Wachter, Mittelstadt, and Russell 2017; Selbst and Barocas 2018). Specifically, counterfactual examples provide this information by showing feature-perturbed versions of the same person who would have otherwise received the loan (Mothilal, Sharma, and Tan 2020).

Contribution: Even though there are many advantages to using counterfactuals as a way of explaining model decisions, they can also be considered as a way to augment a very small dataset. We know that machine learning models usually require a large amount of data and often the performance of the model depends on the size of the dataset. If we do not have enough data to train the model, we might not get the desired outcome from the model. There is often a severe lack of adequate data, which deter us from getting a well trained model. In such a case, augmenting a small dataset can be a viable solution. In this work, we explore this application of counterfactuals and verify the feasibility of this approach through several experiments. We use a well known dataset to conduct the experiments in different steps and analyze the results to measure the extent to which counterfactuals can serve to augment the dataset. We compare our data augmentation approach with a Generative Adversarial Networks (GAN) approach (Goodfellow et al. 2014). In this work, we do not consider data augmentation to correct for class imbalance and leave that as a future work. We are particularly interested in dealing with the situation where we have a small dataset.

The rest of this paper is organized as follows: Next, we present the necessary background material to understand the concept. After that, we discuss related work. Then, we describe and run relevant experiments to test our proposed approach, which is followed by the discussion of the experimental results. We then compare our approach with data augmentation using a GAN-based approach. Finally, we draw some conclusions and describe some future research direction.

Background

In interpretable machine learning, counterfactual explanations can be used to explain predictions of individual instances. The counterfactual explanation method is model-agnostic, since it only relies on the model inputs and output, and the interpretation can be expressed as a summary of the differences in feature values. Counterfactuals are human-friendly explanations, because they are contrastive to the current instance and because they are selective, meaning they usually focus on a small number of feature changes.

We can generate counterfactual explanations using a simple, naive approach, searching by trial and error (Molnar 2019). In this approach, we randomly change feature values of the instance of interest and stop when the desired output is predicted. There are, however, better, more practical approaches than trial and error. We can start by defining a loss function that takes as input the instance of interest and the output is a counterfactual or the desired outcome. This loss function measures how far the predicted outcome of the counterfactual is from the predefined outcome and how far the counterfactual is from the instance of interest (Wachter, Mittelstadt, and Russell 2017). There are two ways to optimize the loss function. One way is to optimize the loss directly with an optimization algorithm like Adam (Adaptive Moment Estimation) (Kingma and Ba 2014). Another way is to search around the instance. Wachter et. al (Wachter, Mittelstadt, and Russell 2017) proposed an approach by minimizing the following loss function, which was later refined by Molnar (Molnar 2019):

$$L(x_i, x'_i, y', \lambda) = \lambda \cdot (\hat{f}(x'_i) - y'_i)^2 + d(x_i, x'_i) \quad (1)$$

Here, the term $\lambda \cdot (\hat{f}(x'_i) - y'_i)^2$ represents the quadratic distance between the model prediction ($\hat{f}(x'_i)$) for the counterfactual x'_i for an instance of interest x_i and the desired outcome y'_i, which the user must define in advance. The second term $d(x_i, x'_i)$ is the distance d between the instance of interest x_i to be explained and the desired counterfactual x'_i.

The parameter λ plays an important role here, which balances the distance in prediction *i.e.* $\lambda \cdot (\hat{f}(x'_i) - y'_i)^2$ against the distance in feature values *i.e.* $\hat{f}(x'_i)$. The loss is solved by choosing an appropriate value of λ, and the solution returns a counterfactual x'_i. The value of λ dictates the kind of compromise we want to make in our preference for counterfactuals. For example, if we choose a higher value of λ that means we prefer counterfactuals that are closer to the desired outcome y'_i. On the other hand, if we go for a lower value λ, we prefer counterfactuals x'_i that are very similar to the instance of interest, x_i, in the feature values. A very large value of λ indicates that, the instance with the prediction that comes closest to y'_i will be selected, no matter how far it is away from x_i.

The choice of λ depends on the user, as he/she must decide how to balance the requirement that the prediction for the counterfactual matches the desired outcome with the requirement that the counterfactual is similar to x_i. Wachter et. al suggest instead of selecting a value for λ, we can select a tolerance ϵ. The tolerance indicates how far away the prediction of the counterfactual instance is allowed to be from y'_i. We can write this constraint in the following way:

$$|\hat{f}(x'_i) - y'_i| \leq \epsilon \quad (2)$$

We can use any suitable optimization algorithm to minimize this loss function in Eq. (2). For example, if we have access to the gradients of the machine learning model, we can use gradient-based methods like RMSprop optimizer (Tieleman and Hinton 2012) or Adam.

Related Work

Mothilal et. al extended the work of Wachter et. al (Wachter, Mittelstadt, and Russell 2017) and provided a method to construct a set of counterfactuals with diversity (Mothilal, Sharma, and Tan 2020). Ribeiro et al. (Ribeiro, Singh, and Guestrin 2016) proposed a feature-based approach, LIME, that fits a sparse linear model to approximate non-linear models locally. Guidottiet al. (Guidotti et al. 2018) extended this approach by fitting a decision tree classifier to approximate the non-linear model and then tracing the decision-tree paths to generate explanations. Similarly, Lundberg and Lee (Lundberg and Lee 2017) provided human-comprehensible approximations for linear models and present a unified framework that assigns each feature an importance value for a particular prediction. Russel worked on efficiently finding coherent counterfactuals avoiding the need for brute-force enumeration (Russell 2019). Ustun et. al worked on evaluating a linear classification model in terms of recourse, which behaves similarly to counterfactuals (Ustun, Spangher, and Liu 2019). In this case, the recourse provided a person the ability to change the decision of the model through actionable input variables.

Mahajan et. al addressed the challenge of the feasibility of counterfactual examples by preserving causal relationships among input features (Mahajan, Tan, and Sharma 2019). There has been some peripheral of counterfactual data augmentation in natural language processing for Mitigating Gender Stereotypes in Languages (Zmigrod et al. 2019) and reducing bias (Kaushik, Hovy, and Lipton 2019). The authors use counterfactual examples as a tool to mitigate those issues but not as a tool to augment the dataset.

None of the approaches talk about how counterfactual examples can efficiently augment dataset especially tabular dataset. In this work, we demonstrate that counterfactual examples can be a viable option for data augmentation using different scenarios. We are specially interested to augment a small dataset rather than addressing class imbalance issue.

Case Study

We apply a technique introduced by Mothilal et. al (Mothilal, Sharma, and Tan 2020) to generate the counterfactual examples (CFEs). We generated counterfactual examples using a shallow artificial neural network (ANN) and then used those counterfactual examples in other models. We used different models to experiment with the generated counterfactual examples to avoid potential bias that might arise when the same model that generated the CFEs is again used to test those CFEs. At the same time, we also wanted to make sure that CFEs generated by one model are transferable to another model.

Dataset

In this experiment, we consider the *Adult-Income* data set, which contains demographic, educational, and other information based on the 1994 Census database and is available on the UCI machine learning repository (Kohavi and Becker 1996). The data set is credited to Ronny Kohavi and Barry Becker (Kohavi 1996). It involves using personal details such as education level, hours of work per week, etc. to predict whether an individual will earn more or less than $50,000 per year. The Adult-Income data set is a widely used, standard machine learning data set and has become a de facto data set for counterfactual example experiments (Karimi et al. 2020; Mothilal, Sharma, and Tan 2020; Mahajan, Tan, and Sharma 2019).

We obtained 8 features, namely, hours per week, education level, occupation, work class, race, age, marital status, and sex by applying the preprocessing based on a prior analysis (Zhu 2016). In this case, the ML model's task is to classify whether an individual's income is over $50,000.

Experiments

We trained an artificial neural network (ANN) model using the Adult-Income dataset. We randomly selected 400 instances and generated a maximum of 4 CFEs for each of the instances. In total, we generated 1000 CFEs. We added these CFEs to multiple, differently sized, subsets of the original Adult-Income dataset. We ran our experiments using six different scenarios.

Experiment 1: First, to establish a gold-standard performance against which to compare the other experiments, we ran an experiment using the whole original Adult-Income dataset to train and tested three kinds of models, which were decision trees, Random Forests, and Bagging applied to decision trees. From the dataset, we made an 80:20 train/test split. The results for this and the other five experiments are reported in Table 1.

Experiment 2: Next, to simulate performance when significantly less data is available, we ran an experiment using only 20% of the original Adult-Income dataset to train and test three the same three models. As in Experiment 1, we used an 80:20 train/test split.

Experiment 3: For our first data augmentation experiment, we combined 20% of the original Adult-Income dataset with the generated CFEs to form our data set and evaluated the performance using the three models. Once again, we used an 80:20 train/test split.

Experiment 4: In this experiment, we trained the models using only the generated CFEs, and for the test set, however, we used the original data from the Adult-Income dataset. The amount of test data was equal to the 20% of the CFEs used to train the models in this experiment.

Experiment 5: This experiment required that we first trained a different ANN model from which to generate the CFEs. This time, however, we used a much smaller dataset *i.e.*, only 10% of the Adult-Income dataset. We then combined the resulting CFEs with the 20% of the original Adult-Income dataset to train and test the three models using an 80:20 train/test split

Experiment 6: In our final experiment, we trained the models using only the CFEs that were generated in Experiment 5. Then, as in Experiment 4, we tested the models again using 20% of the original data from the Adult-Income dataset.

Table 1: Accuracy on test set for the three models for all six experimental scenarios.

Scenario	**Decision Tree**	**Rand. Forest**	**Bagging**
Experiment 1	78.1%	79.99%	80.99%
Experiment 2	72.78%	75.27%	75.72%
Experiment 3	77.18%	79.72%	80.18%
Experiment 4	77.91%	80.03%	80.75%
Experiment 5	76.82%	79.01%	80.02%
Experiment 6	76.11%	78.17%	79.88%

Discussion

To test how well CFEs can be used to augment a dataset, we reduced the original dataset by 80%. This truncated dataset became our new baseline dataset, which we wanted to augment. The rationale behind this idea is if, for some experiments, we do not have enough data to adequately train a model then whether CFEs can be used to effectively supplement the given data.

In such a case, we would use that small dataset to train a model and use this trained model to generate CFEs. These CFEs along with the original dataset can then be used to train another model. In the latter case, choosing a different model than the one used to generate the CFEs is preferable. In this way, we can avoid the bias that might potentially be created using the same model for both the purposes. This also paves the way for the generated CFEs to become model-agnostic *i.e.*, we can train any classifier with the generated CFEs.

The experimental results show very encouraging output in this regard. Experiment 1 shows the accuracies of the three models when models trained and tested using the full dataset. However, Experiment 2 shows the test accuracy, when the models are trained on a much smaller dataset. We wanted to observe whether adding the generated CFEs to this small dataset improved the accuracy *i.e.* whether CFEs can effectively augment the dataset. This attempt resulted in the test accuracy seen in Experiment 3. We observed a clear improvement over Experiment 2 in all three models. However, Experiment 4 put the CFEs to a further test, we trained the three models using only the CFEs and tested the accuracy with the 20% of the original dataset. Like Experiment 3, these results also show improved accuracy compared to Experiment 2.

The CFEs used thus far were generated from an ANN model that was trained on the complete Adult-Income dataset, and we have already observed that they are effectively augmenting the reduced dataset. However, we might not always have a dataset as big as the full Adult-Income dataset. In that case, we would need to train a model with that smaller dataset and subsequently generate CFEs. How effective will data augmentation using those CFEs be? To test this scenario, Experiment 5 trained the ANN model with a much smaller dataset, only 10% of the original Adult-Income dataset and then generated CFEs from this trained model. We merge these CFEs with the 20% of the original Adult-Income dataset to train and test the same three models. The results still show improvement compared to Experiment 2 albeit not as much as the results from Experiment 3.

The slight difference in performance between Experiment 3 and Experiment 5 is understandable. In the former case, we generated CFEs from a model that was trained using a complete dataset, and this gives the model a better understanding of the counterfactual world. As a result, the model generated more effective CFEs. However, in the latter case, the CFEs were generated from a model that was trained on a much smaller dataset, which resulted in a more limited understanding of the counterfactual world and thereby resulting in less effective counterfactual examples. The important takeaway from here is that even those less effective counterfactual examples worked pretty well in augmenting the dataset.

Lastly, Experiment 6 trained those three models again but this time only with the CFEs that were generated using the reduced dataset stated above. In this case, we test the accuracy with the reduced original dataset. In this case, the amount of test data was equal to the 20% of the CFEs used to train the models in this experiment. This experiment's results also show improved performance compared to Experiment 2. Again, The slight difference in performance between Experiment 4 and Experiment 6 can be attributed to the same reasoning we gave for Experiment 3 and Experiment 5.

Overall, we observed a clear improvement in terms of performance for all the three models when we use CFEs to augment the small dataset (20% of the original Adult-Income dataset).

Why Counterfactual Examples Work?

Counterfactual examples generation method is model-agnostic, since it only works with the model inputs and output (Molnar 2019). As we have demonstrated in the experiments, we used one model (e.g. ANN) to generate counterfactual examples and different models (e.g. decision tree, RandomForest, and Bagging) to test the utility of the generated examples.

To generate good counterfactual examples, there are certain criteria that need to be fulfilled. One of the foremost requirements is that a counterfactual instance produces the predefined prediction as closely as possible by defining a relevant change in the prediction of an instance (i.e. the alternative reality) (Molnar 2019). However, it is not always possible to match the predefined output exactly i.e. we might not get proper counterfactual example for every instance. To be considered as a good counterfactual example, it should be as similar as possible to the instance regarding feature values. This criterion necessitates an appropriate distance measure between two instances. The counterfactual example should not only be close to the original instance, but should also change as few features as possible (Molnar 2019). To fulfill this criterion, an appropriate distance measure like the Manhattan distance is required. The last requirement is that a counterfactual instance should have feature values that are likely or practically possible. For example, in the case of Adult-Income dataset, it would not make sense to generate a counterfactual example, which requires an individual to change his/her *race* to earn \$50,000 per year. Fulfilling all these criteria forces counterfactual examples to better approximate the actual instances. As a result, these counterfactuals can be good candidates to augment the dataset.

Generative adversarial networks (GANs)

GANs (Goodfellow et al. 2014) are a neural network architecture that has shown impressive improvements over previous generative methods (Doersch 2016) especially for image data (Frid-Adar et al. 2018; Mariani et al. 2018). A GAN composes of two deep neural networks, which are the generator model and the discriminator model (Mirza and Osindero 2014). Both of networks are simultaneously trained. The task for the generator model is to generate samples, which cannot be distinguished from real samples by the discriminator model. The generator generates a batch of samples, and these, along with real examples from the domain, are provided to the discriminator and classified as real or fake. The discriminator is then updated to get better at discriminating real and fake samples in the next round, and importantly, the generator is updated based on how well, or not, the generated samples fooled the discriminator (Goodfellow 2016). In this way, the two models are competing against each other, they are adversarial in the game theory sense, and are playing a zero-sum game (Gibbons 1992). At a limit, the generator generates perfect replicas from the input domain every time, and the discriminator cannot tell the difference and predicts "unsure" (e.g. 50% for real and fake) in every case. The training drives the discriminator to attempt to learn to correctly classify samples as real or fake. Simultaneously, the generator attempts to fool the classifier into believing its samples are real. At convergence, the generator's samples are indistinguishable from real data, and the discriminator outputs 1/2 everywhere. The discriminator may then be dis-

Table 2: Comparing the accuracy of GAN and CFE as a Data Augmentation Technique.

Scenario	**Decision Tree**		**Rand. Forest**		**Bagging**	
	CFE	GAN	CFE	GAN	CFE	GAN
Experiment 7	77.18%	59.39%	79.72%	64.66%	80.18%	67.66%
Experiment 8	77.91%	59.97%	80.03%	70.48%	80.75%	73.26%
Experiment 9	76.82%	58.26%	79.01%	69.09%	80.02%	71.74%
Experiment 10	76.11%	64.25%	78.17%	69.25%	79.88%	68.4%

carded (Bengio, Goodfellow, and Courville 2017).

The counterfactual generation method, on the other hand, does not require access to the data or the model. The method can do its job by only accessing the model's prediction function. Once, the method has access to the prediction function, it can generate n ($n \in N \cup \{0\}$) number of CFEs for each of the instances following the criteria stated in Section and a loss function like Eq. (1). Usually, we should be able to find n CFEs for each of the instances. However, there might be cases where we might not find CFEs for some instances. In this case, the burden lies more with the quality of those instances than on the CFEs generation model.

Experimental result comparison with GANs

Recently, we have seen a flurry of works using GANs image data augmentation (Wu et al. 2018; Tanaka and Aranha 2019; Han et al. 2019; Singh, Dutta, and Saha 2019).However, GANs are also being considered for tabular data augmentation (Ba 2019; Xu and Veeramachaneni 2018).To get a sense of which of the two approaches perform better, we compare our counterfactual data augmentation approach with GANs for tabular data as shown in Table 2. We use Tabular GAN (TGAN) (Xu and Veeramachaneni 2018) to generate the data that we use for data augmentation in different steps.

This time as well we use the Adult-Income dataset and generate CFEs and GAN data in different scenarios. We ran our experiments using four different scenarios. In this case, Experiment 9, Experiment 10, Experiment 11, and Experiment 12 correspond to Experiment 3, Experiment 4, Experiment 5, and Experiment 6, respectively but this time the performance is compared with the GAN generated dataset.

Experiment 7: We combined 20% of the original Adult-Income dataset with the generated GAN data to form our data set and evaluated the performance using the three models. Once again, we used an 80:20 train/test split. The result is shown Table 2 along with the similar experiment done with CFEs in Experiment 3.

Experiment 8: Similar to Experiment 4, we trained the models using only the generated GAN data, and for the test set, we used the original data from the Adult-Income dataset. The amount of test data was equal to the 20% of the GAN data used to train the models in this experiment. The result is shown Table 2 along with the similar experiment done with CFEs in Experiment 4.

Experiment 9: In this case, we used a much smaller dataset *i.e.*, only 10% of the Adult-Income dataset to train the GAN discriminator. We then combined the resulting GAN data with the 20% of the original Adult-Income dataset to train and test the three models using an 80:20 train/test split. The result is shown Table 2 along with the similar experiment done with CFEs in Experiment 5.

Experiment 10: Similar to Experiment 6, we trained the models using only the GAN data that were generated in Experiment 9. We tested the models using 20% of the original data from the Adult-Income dataset. The result is shown Table 2 along with the similar experiment done with CFEs in Experiment 6.

From Table 2, we observe that counterfactual examples approach for dataset augmentation outperforms GAN approach in each of the experiments. Though we see some performance improvements on experiment basis for GAN dataset augmentation approach, those improvements are dwarfed by the counterfactual examples approach in each of the experiments. For example, counterfactual examples approach for dataset augmentation offers 29.95%, 23.29%, 18.5% better accuracy for Decision Tree, Random Forest, and Bagging, respectively in Experiment 7. Moreover, the performance of GAN is even worse than that of small dataset without augmentation. For example, the worst performer with the small dataset was decision tree with 72.78% accuracy and none of the data augmentation scenario with GAN can even beat that. The very reason behind the poor performance of data augmentation with GAN might be that GAN data looks too much like the original data and does not bring any new information to the dataset, whereas CFEs apparently do contribute useful information and put pints in space. Additionally, GAN is data hungry process, which cannot perform well in a data starved environment.

Conclusion and Future Work

Counterfactual examples as a mean of interpretabiltiy and explainability in machine learning models has garnered a lot of attention in recent literary work. As a possible application area of counterfactual examples, we wanted to explore whether data augmentation can be of those. Our experimental results show some positive signs in this endeavor. We compared counterfactual dataset augmentation approach with GAN dataset augmentation approach and we observed that counterfactual approach is more effective. In this case, counterfactual approach for tabular dataset augmentation looks promising. In the future, we want to experiment with other datasets from different application areas. Additionally, we want to consider other models and other counterfactual example generation techniques. Other than merely augmenting the dataset, we want to explore how CFEs can also be used to address class imbalance.

References

Ba, H. 2019. Improving detection of credit card fraudulent transactions using generative adversarial networks. *arXiv preprint arXiv:1907.03355*.

Barocas, S.; Selbst, A. D.; and Raghavan, M. 2020. The hidden assumptions behind counterfactual explanations and principal reasons. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 80–89.

Bengio, Y.; Goodfellow, I.; and Courville, A. 2017. *Deep learning*, volume 1. MIT press Massachusetts, USA:.

Buchanan, B. G., and Shortliffe, E. H. 1984. Rule-based expert systems: the mycin experiments of the stanford heuristic programming project.

Doersch, C. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

Frid-Adar, M.; Klang, E.; Amitai, M.; Goldberger, J.; and Greenspan, H. 2018. Synthetic data augmentation using gan for improved liver lesion classification. In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, 289–293. IEEE.

Gibbons, R. 1992. Game theory for applied economics. Princeton University Press.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

Goodfellow, I. 2016. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*.

Guidotti, R.; Monreale, A.; Ruggieri, S.; Pedreschi, D.; Turini, F.; and Giannotti, F. 2018. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*.

Han, C.; Rundo, L.; Araki, R.; Nagano, Y.; Furukawa, Y.; Mauri, G.; Nakayama, H.; and Hayashi, H. 2019. Combining noise-to-image and image-to-image gans: Brain mr image augmentation for tumor detection. *IEEE Access* 7:156966–156977.

Karimi, A.-H.; Barthe, G.; Balle, B.; and Valera, I. 2020. Model-agnostic counterfactual explanations for consequential decisions. In *International Conference on Artificial Intelligence and Statistics*, 895–905.

Kaushik, D.; Hovy, E.; and Lipton, Z. C. 2019. Learning the difference that makes a difference with counterfactually-augmented data. *arXiv preprint arXiv:1909.12434*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kohavi, R., and Becker, B. 1996. UCI machine learning repository. https://archive.ics.uci.edu/ml/datasets/adult.

Kohavi, R. 1996. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, 202–207.

Lundberg, S. M., and Lee, S.-I. 2017. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, 4765–4774.

Mahajan, D.; Tan, C.; and Sharma, A. 2019. Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv preprint arXiv:1912.03277*.

Mariani, G.; Scheidegger, F.; Istrate, R.; Bekas, C.; and Malossi, C. 2018. Bagan: Data augmentation with balancing gan. *arXiv preprint arXiv:1803.09655*.

Mirza, M., and Osindero, S. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Molnar, C. 2019. *Interpretable Machine Learning*. https://christophm.github.io/interpretable-ml-book/.

Mothilal, R. K.; Sharma, A.; and Tan, C. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 607–617.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 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*, 1135–1144.

Russell, C. 2019. Efficient search for diverse coherent explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 20–28.

Selbst, A. D., and Barocas, S. 2018. The intuitive appeal of explainable machines. *Fordham L. Rev.* 87:1085.

Singh, A.; Dutta, D.; and Saha, A. 2019. Migan: malware image synthesis using gans. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 10033–10034.

Tanaka, F. H. K. d. S., and Aranha, C. 2019. Data augmentation using gans. *arXiv preprint arXiv:1904.09135*.

Tieleman, T., and Hinton, G. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4(2):26–31.

Ustun, B.; Spangher, A.; and Liu, Y. 2019. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 10–19.

Wachter, S.; Mittelstadt, B.; and Russell, C. 2017. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.* 31:841.

Wu, E.; Wu, K.; Cox, D.; and Lotter, W. 2018. Conditional infilling gans for data augmentation in mammogram classification. In *Image Analysis for Moving Organ, Breast, and Thoracic Images*. Springer. 98–106.

Xu, L., and Veeramachaneni, K. 2018. Synthesizing tabular data using generative adversarial networks. *arXiv preprint arXiv:1811.11264*.

Zhu, H. 2016. Predicting earning potential using the adult dataset.

Zmigrod, R.; Mielke, S. J.; Wallach, H.; and Cotterell, R. 2019. Counterfactual data augmentation for mitigating gender stereotypes in languages with rich morphology. *arXiv preprint arXiv:1906.04571*.