# CERTIFAI: Counterfactual Explanations for Robustness, Transparency, Interpretability, and Fairness of Artificial Intelligence models

**Shubham Sharma**[1], **Jette Henderson**[2], **Joydeep Ghosh**[2]
[1]Department of Electrical and Computer Engineering, University of Texas at Austin
[2]CognitiveScale
shubham_sharma@utexas.edu, jhenderson@cognitivescale.com, jghosh@utexas.edu

## Abstract

As artificial intelligence plays an increasingly important role in our society, there are ethical and moral obligations for both businesses and researchers to ensure that their machine learning models are designed, deployed, and maintained responsibly. These models need to be rigorously audited for fairness, robustness, transparency, and interpretability. A variety of methods have been developed that focus on these issues in isolation, however, managing these methods in conjunction with model development can be cumbersome and time-consuming. In this paper, we introduce a unified and model-agnostic approach to address these issues: Counterfactual Explanations for Robustness, Transparency, Interpretability, and Fairness of Artificial Intelligence models (CERTIFAI). Unlike previous methods in this domain, CERTIFAI is a general tool that can be applied to any black-box model and any type of input data. Given a model and an input instance, CERTIFAI uses a custom genetic algorithm to generate counterfactuals[1]: instances close to the input that change the prediction of the model. We demonstrate how these counterfactuals can be used to examine issues of robustness, interpretability, transparency, and fairness. Additionally, we introduce CERScore, the first black-box model robustness score that performs comparably to methods that have access to model internals.

## 1 Introduction

As the adoption of machine learning models to everyday tasks grows rapidly, so does the need to consider the ethical, moral, and social consequences of the decisions made by such models. Several important questions arise in a variety of applications, such as the following: 1) how did the model predict what it predicted?, 2) if a person got an unfavorable outcome from the model, what can they do to change that?, 3) has the model been unfair to a particular group?, and 4) how easily can the model be fooled? Researchers are actively building separate approaches to answer each of these questions.

[1] Counterfactuals have a well-established meaning in the causality literature. However, we are using "counterfactual" in the counterfactual explanation sense, one that has been recently initiated in the explainability literature [Wachter *et al.*, 2017] where the model implies a machine learning model and not a causal model and hence no causal assumptions are made here

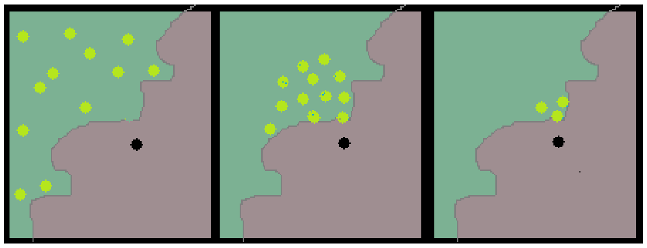
Figure 1: The CERTIFAI counterfactual generation process. The decision boundary for a binary classifier is shown, with the input instance in black. We sample a set of points (left) in the feature space with a constraint that they must lie on the other side of the decision boundary (green points). The algorithm then evolves these samples (middle) to generate individuals that lie closer to the input point but on the other side of the decision boundary. Finally, a smaller set, the size of which is user-defined, of counterfactuals is generated (right).

One promising vein of research in explainability, first introduced by [Wachter *et al.*, 2017], is generating counterfactuals. Given an input data point and a black-box machine learning model (i.e. we only have access to the model's prediction for any input), a *counterfactual* is defined as a generated data point that is as close to the input data point as possible but for which the model gives a different outcome. For example, if a user was denied a loan by a machine learning model, an example counterfactual explanation could be: "Had your income been $5000 greater per year and your credit score been 30 points higher, your loan would be approved." [Wachter *et al.*, 2017] argue that counterfactuals are a way of explaining model results to users such that they can identify actionable ways of changing their behaviors to obtain favorable predictions. In addition to providing counterfactuals for explainability, we show how counterfactual explanations can be used to audit fairness and robustness of a model.

As promising as the original method [Wachter *et al.*, 2017] and subsequent methods of generating counterfactuals [Ustun *et al.*, 2019; Russell, 2019] are, they are also limited in that some only work for linear models, while others cannot deal

| Method | Black-box | Model-Agnostic | Mixed-data | Explainability | Fairness | Robustness |
|---|---|---|---|---|---|---|
| CERTIFAI | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| [Ustun *et al.*, 2019] | ✓ |  | ✓ | ✓ | ✓ |  |
| [Wachter *et al.*, 2017] | ✓ | ✓ |  | ✓ | ✓ |  |
| [Russell, 2019] | ✓ |  | ✓ | ✓ | ✓ |  |
| [Ribeiro *et al.*, 2016] | ✓ | ✓ | ✓ | ✓ |  |  |
| [Guidotti *et al.*, 2018a] | ✓ | ✓ | ✓ | ✓ |  |  |
| [Carlini and Wagner, 2017] |  | ✓ |  |  |  | ✓ |
| [Weng *et al.*, 2018] |  | ✓ |  |  |  | ✓ |

Table 1: Comparison of related work with our approach. We consider the approach most similar to ours. Mixed-data means the method can work with both discrete and continuous data, without any discretization or assumptions.

with different data types. To resolve these limitations, we introduce CERTIFAI, a novel, flexible, model-agnostic technique for generating counterfactuals via a custom genetic algorithm. The meta-heuristic evolutionary algorithm starts by generating a random set of points such that they do not have the same prediction as the input point. A subsequent evolutionary process results in a set of points close to the input that maintain the prediction constraint. Figure 1 shows an example of three counterfactuals (green points) generated for a given input (black point).

A major advantage of using the genetic algorithm to generate counterfactuals is that it can generate counterfactuals for linear and non-linear models (e.g. deep networks) and for any input form (from mixed tabular data to image data) without any approximations to or assumptions for the model. Moreover, end-users can 1) define a range for any feature, and 2) restrict the features that can change. CERTIFAI simply constrains the values of the sampled points based on those choices, allowing the generated counterfactuals to reflect a user's understanding of how much it is *possible* for them to change their features.

CERTIFAI can be used to audit any black-box classifier, providing three tools that are based on a single underlying algorithm. The major contributions of this paper are summarized as follows:

- Counterfactuals are generated using a custom genetic algorithm, which is model-agnostic, flexible, and can be used to provide explanations.
- Counterfactuals are shown to be effective adversarial examples. They are also used to generate the Counterfactual Explanation-based robustness scores (CERScore), which to the best of our knowledge is the first ever black-box model robustness score.
- Counterfactuals can be used to evaluate fairness with respect to a user's input as well as the fairness of the model towards groups of individuals.

## 2 Related Work

Table 1 summarizes the key features of CERTIFAI and the work most related to CERTIFAI. Other general methods on explainability, fairness and robustness have been described by [Guidotti *et al.*, 2018b],[Binns, 2017], and [Akhtar and Mian, 2018] respectively. As we can see, there is no prior art that handles a diverse set of desirable properties needed to develop a responsible AI system. [Wachter *et al.*, 2017] introduced counterfactual explanations, however, their optimization formulation cannot handle categorical data (i.e. the optimization does not solve for such data; the values are found in a brute-force fashion). Methods in [Ustun *et al.*, 2019] and [Russell, 2019] only work for linear models. [Guidotti *et al.*, 2018a] uses a genetic algorithm to generate neighbors around the input and then use decision trees to locally approximate the model. However, local approximations might be at the cost of model accuracy, and they define counterfactuals based on the minimum number of splits in the trained decision tree, which might not always be the smallest change to the input. The closest work to generating adversarial examples is by [Carlini and Wagner, 2017]. They work on a white-box and simpler convolution models, and the distance metrics might not be apt to measure image similarity. [Wang *et al.*, 2002]. [Weng *et al.*, 2018] define a robustness score CLEVER. However, they have access to the model gradients.

## 3 The CERTIFAI framework

In this section, we formulate a custom genetic algorithm to find counterfactual(s). Consider a black-box classifier $f$ and an input instance $\mathbf{x}$. Let the counterfactual be a feasible generated point $\mathbf{c}$. Then the problem can be formulated as:

$$\begin{aligned} &\min_{c} d(\mathbf{x}, \mathbf{c}) \\ &\text{s.t.} f(\mathbf{c}) \neq f(\mathbf{x}) \end{aligned} \tag{1}$$

where $d(\mathbf{x}, \mathbf{c})$ is the distance between $\boldsymbol{x}$ and $\boldsymbol{c}$. To avoid using any approximations to or assumptions for the model, we use a genetic algorithm to solve Equation 1. The custom genetic algorithm works for any black-box model and input data type, and it is model-agnostic. Additionally, it provides a great deal of flexibility in counterfactual generation.

CERTIFAI's genetic algorithm solves the optimization problem in Equation 1 through a process of natural selection. The only mandatory inputs for the genetic algorithm are the black-box classifier $f$ and an input instance $\mathbf{x}$. Generally, for an $n$-dimensional input vector $\mathbf{x}$, let $W \in \mathbb{R}^n$ represent the space from which individuals can be generated and $P$ be the set of points with the same prediction as $\mathbf{x}$:

$$P = \{\mathbf{p} | f(\mathbf{p}) = f(\mathbf{x}), \mathbf{p} \in W\}. \tag{2}$$

The possible set of individuals $\mathbf{c} \in I$ are defined such that

$$I = W \setminus P. \tag{3}$$

Each individual $\mathbf{c} \in I$ is a candidate counterfactual. The goal is to find the fittest possible $\boldsymbol{c}^*$ to $\mathbf{x}$ constrained on $\boldsymbol{c}^* \in I$. The fitness for an individual $\mathbf{c}$ is defined as:

$$fitness = \frac{1}{d(\mathbf{x}, \mathbf{c})}. \tag{4}$$

Here $\boldsymbol{c}^*$ will then be the point closest to $\mathbf{x}$ such that $\boldsymbol{c}^* \in I$. For a multi-class case, if a user wants the counterfactual $\mathbf{c}$ to be belong to a particular class $j$, we define $Q$ as:

$$Q = \{\mathbf{q} | f(\mathbf{q}) = j, \mathbf{q} \in W\}. \tag{5}$$

Then Equation 3 becomes:

$$I = (W \setminus P) \cap Q. \tag{6}$$

The algorithm is carried out as follows: first, a set $I_c$ is built by randomly generating points such that they belong to $I$. Individuals $\boldsymbol{c} \in I_c$ are then evolved through three processes: selection, mutation, and crossovers. Selection chooses individuals that have the best fitness scores (Equation 4). A proportion of these individuals (dependent on $p_m$, the probability of mutation) are then subjected to mutation, which involves arbitrarily changing some feature values. A proportion of individuals (dependent on $p_c$, the probability of crossover) are then subjected to crossover, which involves randomly interchanging some feature values between individuals. The population is then restricted to the individuals that meet the required constraint (Equation 3 or Equation 6), and the fitness scores of the new individuals are calculated. This is repeated until the maximum number of generations is reached. Finally, the individual(s) $\boldsymbol{c}^*$ with the best fitness score(s) is/are chosen as the desired counterfactual(s)[2].

## 3.1 Choice of distance function

The choice of distance function used in Equation 1 depends on the details provided by the model creator and the type of data being considered. If the data is tabular, [Wachter *et al.*, 2017] demonstrated how the $L_1$ norm normalized by the median absolute deviation (MAD) is better than using the $L_1$ or $L_2$ norm for counterfactual generation. For tabular data, the $L_1$ norm for continuous features (NormAbs) and a simple matching distance for categorical features (SimpMat) are chosen as default. In the absence of training data, normalization using MAD is not possible. However in model development and our experimenets where there is access to training data, normalization is possible. The distance metric used is:

$$d(\mathbf{x}, \mathbf{c}) = \frac{n_{con}}{n}\text{NormAbs}(\mathbf{x}, \mathbf{c}) + \frac{n_{cat}}{n}\text{SimpMat}(\mathbf{x}, \mathbf{c}) \tag{7}$$

where $n_{con}$ and $n_{cat}$ are the number of continuous and categorical features, respectively, and $n$ is the total number of features ($n_{con} + n_{cat} = n$).

[2] p$_m$=0.2 and p$_c$=0.5, which is standard in literature. The population size is the square of the input feature size with a maximum cap of 30,000. Grid-search is used to find the number of generations

For image data, the Euclidean distance and absolute distance between two images are not good measures of image similarity [Wang *et al.*, 2002]. Hence, we use SSIM (Structural Similarity Index Measure) [Wang *et al.*, 2003], which has been shown to be a better measure of what humans consider to be similar images [Wang *et al.*, 2002]. SSIM values lie between 0 and 1, where a higher SSIM value means that two images look more similar to each other. For the input image $\mathbf{x}$ and counterfactual image $\mathbf{c}$, the distance is:

$$d(\mathbf{x}, \mathbf{c}) = \frac{1}{\text{SSIM}(\mathbf{x}, \mathbf{c})}. \tag{8}$$

## 3.2 Improving counterfactuals with constraints

Apart from the input instance and black-box model, additional inputs help the algorithm produce better results. Auxiliary constraints are incorporated by restricting the space defined by the set $W$: the space from which individuals can be generated, to ensure feasible solutions. For an $n$-dimensional input, let $W$ be the Cartesian product of the sets $W_1$,$W_2$,...,$W_n$. For continuous features, $W_i$ can be constrained as $W_i \in [W_{imin}, W_{imax}]$, and categorical features can be constrained as $W_i \in \{W_1, W_2, ..., W_j\}$. However, certain variables might be immutable (e.g., race). In these cases, a feature $i$ for an input $\mathbf{x}$ can be muted by setting $W_i = x_i$.

An example of the benefits of such constraints would be when a user may not want an explanation of an income change from \$10,000 to \$900,000 if that is not possible, so $W_i \in [\$10000, \$15000]$ might be an appropriate constraint. The number of counterfactuals $k$ can also be set. CERTIFAI chooses the top $k$ individuals ($k = 1$ as default) where different features have changed, so the end-user can get multiple diverse explanations of different kinds.

## 3.3 Robustness

Machine learning models are prone to attacks and threats. For example, deep learning models have performed exceedingly well for image recognition tasks, but it has been widely shown [Carlini and Wagner, 2017], [Nguyen *et al.*, 2015] that these networks are prone to adversarial attacks. Two images may look the same to a human, but when presented to a model, they can produce different outcomes. A counterfactual is a generated point close to an input that changes the prediction and is therefore an adversarial example.

Given two black-box models, if the counterfactuals across classes are farther away from the input instances on average for one network as compared to the other network, that network would be harder to fool. Since CERTIFAI directly gives a measure of distance d($\mathbf{x}$,$\mathbf{c}$), this can be used to define the robustness score for a classifier. Using this distance, we introduce Counterfactual Explanation-based Robustness Score (CERScore), the first ever black-box model robustness score. Given a model, the CERScore is defined as the expected distance between the input instances and their corresponding counterfactuals:

$$CERScore(model) = \mathop{\mathbb{E}}_{X}[d(\mathbf{x}, \mathbf{c}^*)]. \tag{9}$$

To be able to better compare models trained on different data sets, the CERScore can be normalized by the expected value of the distance between data points in each class over all classes $k$, and hence we get the normalized CERScore NCER-Score (abbreviated as NC) as:

$$NC = \frac{\mathbb{E}_X[d(\mathbf{x}, \mathbf{c}^*)].}{\sum_{k=1}^{K} P(x \in \text{class}_k)\mathbb{E}[d(x_i, x_j); x_i, x_j \in \text{class}_k]} \quad (10)$$

(i.e., we normalize by dividing by the expected distance between two datapoints drawn from the same class). A higher CERScore implies that the model is more robust. Note that the normalized CERScore can be greater than 1. Unlike [Weng *et al.*, 2018], CERTIFAI only needs model predictions and not the model internals.

## 3.4 Fairness

The fitness measure (Equation 4) and CERScore can also be used to investigate fairness from individual and group perspectives, respectively. For a given individual instance, if the genetic algorithm can generate different counterfactuals with different values of a protected feature (e.g., race, age), and as a result the user can achieve the desired outcome more easily than when those features could not be changed, then the individual could claim the model is unfair to their case. Additionally, CERTIFAI can be used by model developers to audit the fairness for different groups of observations. If the fitness measure is markedly different for counterfactuals generated for the different partitions of a feature's domain value, this could be an indication the model is biased towards one of the partitions. For example, if the gender feature is partitioned into two values (male and female), and the average fitness values of generated counterfactuals are lower for females than for males, this could be used as evidence that the model is not treating females fairly. Using counterfactuals and the distance function, we can calculate the overall burden for a group, measured as:

$$Burden(g) = \mathop{\mathbb{E}}_{g}[d(\mathbf{x}, \mathbf{c}^*)] \quad (11)$$

where $g$ is a partition defined by the distinct values for a specified feature set. Note, burden is related to CERScore as it is the expected value over a group. Most fairness auditing models focus on single features (e.g., [Hardt *et al.*, 2016; Donini *et al.*, 2018]). Burden, however, does not have that limitation and can be applied to any combination of features.

## 4 Experiments

We demonstrate the applications and flexibility of CERTIFAI to explainability, transparency, fairness, and robustness.

### 4.1 Robustness

In this section, we demonstrate how CERTIFAI produces adversarial examples, and we use CERScore from Section 3.3 to measure a network's resistance to adversarial attacks.

**Generating Adersarial Examples**

We consider the MNIST dataset [LeCun, 1998] which contains 60000 (size 28x28) training images of digits. We use it to train a convolutional neural network, consisting of one convolution layer, two dense layers, and intermediate pooling and dropout layers. We achieve a 99.46% accuracy on the test set using this architecture. Then we select random images from the MNIST dataset and the model above and find the counterfactual image for each image, using SSIM (equation (8)). Every pixel is considered to be a feature and hence, every individual in the population is a 784 dimension vector. We use an initial population size of 30,000 individuals and run the experiment for 1,000 generations.

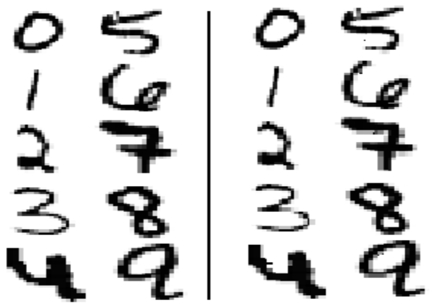

Figure 2: Adversarial examples for the MNIST dataset. The images on the left are original inputs for which the model has a correct prediction and the images on the right are counterfactual images (adversarial examples) for which the prediction has changed.

| Model | CERScore | CI | CLEVER |
|---|---|---|---|
| Inception-v3 | 1.17 | 1.09-1.25 | 0.229 |
| Resnet-50 | 1.06 | 1.05-1.08 | 0.137 |
| MobileNet | 1.08 | 1.06-1.09 | 0.151 |

Table 2: Robustness score and 95 percent confidence intervals (CI) for those scores for 3 deep learning models and the corresponding CLEVER scores

Figure 2 shows an example of ten generated counterfactuals (right) and their original counterpart (left) images. The counterfactual images on the right look nearly identical to the input images on the left, however, the model predicts a different outcome for the images on the left and right. The imperceptibly different images give credence to the idea of using a genetic algorithm formulation to produce counterfactuals. Additionally, our approach towards generating these images is model-agnostic and does not require any approximations, unlike [Carlini and Wagner, 2017]. The generated images show how a network can easily be fooled and demonstrate that there is a major problem in deploying such highly-accurate networks to image-based decision making applications (eg. face recognition). Moreover, different kinds of adversarial attacks can be generated by simply changing the distance function in Equation 4.

**Evaluating Deep Networks**

In this section, we evaluate how well CERScore, introduced in Section 3.3, can give an informative measure of robustness. We consider the same networks as in [Weng *et al.*, 2018]: Inception-v3 [Szegedy *et al.*, 2016] , ResNet-50 [He *et al.*, 2016] and MobileNet [Howard *et al.*, 2017] pre-trained

| **Data set** | **Num. obs.** | **Num. features** | **DT NCERS.** | **DT Acc.** | **SVM NCERS.** | **SVM Acc.** | **MLP NCERS.** | **MLP Acc.** |
|---|---|---|---|---|---|---|---|---|
| Pima Diabetes | 768 | 8 | 0.074 | 73.25 | 0.387 | 81.42 | **0.486** | 98.61 |
| Breast Cancer | 569 | 32 | 0.081 | 95.80 | **0.121** | 96.50 | **0.124** | 96.50 |
| Iris | 150 | 4 | 0.132 | 95.67 | 0.235 | 95.67 | **0.241** | 95.67 |

Table 3: Descriptions of data sets, and NCERScore (NCERS.) and test set accuracy (Acc.) for three models: decision tree (DT), SVM with RBF kernel (SVM), and Multilayer Perceptron (MLP).

on ImageNet [Deng *et al.*, 2009], where they define the CLEVER score for robustness. Unlike CLEVER, we consider the model to be a black-box (only relying on its predictions). Ideally, to derive a measure of robustness for a model, all images from all classes should be considered, their counterfactuals should be generated, and the CERScore should then be calculated. However, since the number of training samples for a deep network is in the order of millions, it is not computationally feasible to calculate the score for each example. Hence, we consider a subset of classes and images to calculate the CERScore. We sampled $n$=50 random images from every class across $k$=100 random classes. We generate the counterfactuals for all 5,000 images such that the counterfactual gives a prediction of the second most likely class (by generating individuals constrained on belonging to that class as in Equation 6) and empirically estimate the CERScore as:

$$CERScore = \frac{1}{nk}\sum_{i=1}^{k}\sum_{j=1}^{n} d(\mathbf{x}_{ij}, \mathbf{c}^*_{ij}) \qquad (12)$$

where $\mathbf{x}_{ij}$ is the $j^{th}$ input instance belonging to predicted class $i$, and $\mathbf{c}^*_{ij}$ is the corresponding counterfactual. The CERScores are shown in Table ??. One way to interpret the score is that on average, the SSIM score for Inception-v3 is 1/1.17 = 0.85, where an SSIM score of 1 means the images look exactly the same and an SSIM score of 0 means the images are highly different. Hence, adversarial attacks for Inception-v3 could be more easily identified than for the other models. We also show the 95% confidence interval where we have assumed the distribution of distances between the images and their counterfactuals follows a normal distribution. The confidence intervals are tight around the CERScores.

We also compare CERScore with CLEVER scores [Weng *et al.*, 2018] for the same images, considering the top-2 class attack. The CLEVER scores are also reported in Table ??. The CERScore implies that Inception-v3 is most robust and Resnet-50 is least robust, which is similar to what the CLEVER scores suggest. Hence, even though CERTIFAI does not access any model weights, it is able to evaluate a model's robustness to adversarial attacks.

**Robustness of Classic Classifiers**

Next, we use NCERScore (Equation 10) to compare the robustness of different models trained on different data sets. We train three models (decision trees (DT), Support Vector Machines with RBF kernel (SVM), and multilayer perceptrons (MLP)) on the three data sets listed in Table ??. We report the NCERScore and the accuracy on the test set in Table ??. Across all data sets, the neural network has the highest NCERScore and is therefore the most robust of the classifiers for these data sets. In the Pima diabetes data set, the accuracy of the decision tree is much lower than the other models, which suggests this simple model cannot adequately capture the class separation. Hence, more points would be concentrated near the decision boundaries, resulting in a lower NCERScore.

For the Iris data set, while it is a relatively simple data set (even the decision tree performs well), the decision tree has the lowest NCERScore while the scores for SVM and MLP are similar. In Figure 3, we plot input points for two features of the Iris data set and the decision boundary for each model. Looking closely, the points for the decision tree are closer on average to the decision boundaries as compared to the other two models (i.e. the densely clustered white and red points are closer to decision boundaries), which suggests the model is more prone to being fooled. The decision boundaries for both SVM and MLP are nearly identical around the input points, which results in similar robustness scores. Similar results can be seen for the cancer data set. Using these results, a model developer can choose the most robust model based on the NCERScore.

| Person | Feature(s) | Original | Counterfactual |
|---|---|---|---|
| 1 | Glucose (CWC) | 115 | 71 |
| | BMI (CUC) | 35.3 | 10.1 |
| 2 | Glucose (CWC) | 168 | 89 |
| | Age (CUC) | 34 | 44 |

Table 4: Counterfactual explanations for the Pima Indian diabetes dataset. CWC: counterfactuals with constraints on feature values and CUC: unconstrained counterfactuals. Unconstrained features lead to infeasible solutions (BMI 10.1) or unchangeable features (age) being changed.

## 4.2 Explainability

Counterfactuals are used to provide explanations and transparency to a user on how much change is needed for them to obtain a favorable prediction. We show the importance of using the constraints to improve explanations, the use of multiple counterfactual explanations for a single instance, and how these can also be used to estimate feature importance.

**Datasets and Models**

We use the Pima Indian dataset [Smith *et al.*, 1988] and the UCI adult dataset [Kohavi, 1996] in the following experiments. The Pima Indian dataset consists of 768 data samples

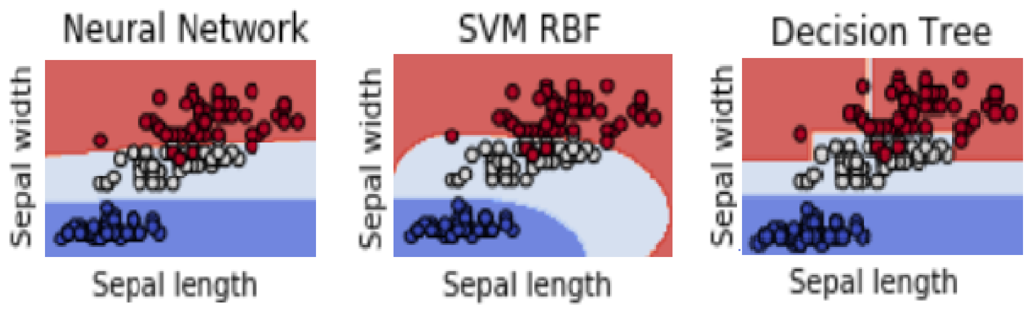

Figure 3: Decision boundaries and input points for the Iris flowers data set (3 classes) for 3 models: Neural Network, SVM with RBF kernel and Decision tree (DT), visualized using two features from the data set. DT has closer points to the decision boundaries on average.

| Person | Feature(s) | Original | Counterfactual |
|---|---|---|---|
| 1 | Education<br>Occupation | 12th<br>Tech-suppt | Bachelors<br>Exec-managerial |
| 1 | Hrs-per-week<br>Workclass | 50<br>Local-gov | 70<br>Private |

Table 5: Two explanations for the same person from the UCI adult dataset, with constraints on feature values.

and 8 features where 6 features are continuous and integer-valued, and 2 features are continuous float-valued. The task is to predict the risk of diabetes (1: At risk, 0:Not at risk). We train a 4 layer neural network with an input layer, 2 hidden layers of 20 neurons each, and an output layer with a 80-20 training-test split. The accuracy of the model is 99.6% on the test set. An initial population of 500 individuals is considered and the evaluation is done across 300 generations. The UCI adult dataset consists of 48842 samples with 14 categorical and continuous integer features and a binary outcome of predicted income (>50k or <=50k). Since the dataset contains many categorical variables, finding a counterfactual using [Wachter *et al.*, 2017] would not be feasible. We train a 6 layer neural network with an input layer, 4 hidden layers of 80 neurons each, and an output layer with a 80-20 training-test split. The accuracy of the model is 99.20% on the test set. Since the dataset is larger, an initial random population of 1000 individuals is considered and the evaluation is done across 500 generations. The negative outcome is considered to be income <=50k, and we find the counterfactuals for those. We only consider those input instances where the model prediction matches the ground-truth.

## Importance of Constraints

We consider two cases of counterfactual generation, counterfactuals with constraints (CWC) and counterfactuals unconstrained (CUC) for users with a prediction of high diabetes risk. CWC corresponds to a user or model creator providing a range of values for features. CUC corresponds to a user only providing the black-box model and the input instance

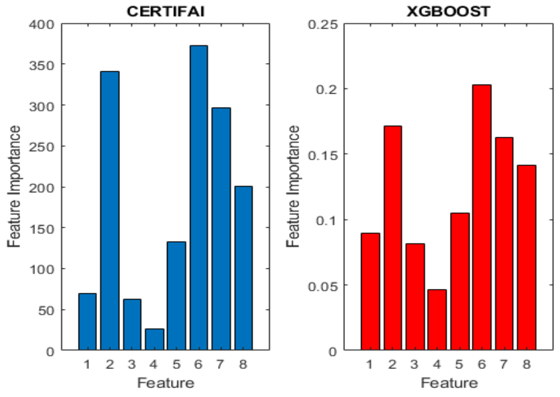

Figure 4: Feature importance for the model, trained on the Pima Indian diabetes dataset, measured by the number of times a feature changed to generate the counterfactual (left) and feature importance by XGBoost (right).

without any constraints on the feature values. We show features for which the values have changed (between the input and counterfactual), all other values remained constant.

As shown in Table 4, for person 1, when we provide constraints (CWC), the explanation is: *Had your glucose been less by 34, you wouldn't have been at the risk of diabetes*. All other feature values for the user remained constant. Without constraints, the explanation shows that the BMI would have to be decreased to 10.1. While this is a smaller change in magnitude as compared to changing the glucose level, achieving a BMI of 10.1 is not feasible, and hence it is important to use the flexibility of our approach to add additional constraints that ensure feasibility. Similarly, for person 2, the age is suggested to be changed, which is not feasible.

## Measuring feature importance

From a model developer's perspective, counterfactuals can show the importance of every feature value to the prediction and hence provide transparency. If CERTIFAI is changing a particular feature more often than another feature when comparing the input and counterfactual, that feature is more significant for a model. For the Pima Indian diabetes dataset, we

| Person | Feature | FitnessM | FitnessU |
|---|---|---|---|
| 1 | Race | 0.63 | 0.87 |
| 2 | Gender | 0.41 | 0.62 |
| 3 | Race | 0.81 | 0.81 |

Table 6: Fitness values when race and gender attributes are muted (FitnessM) and unmuted (FitnessU) for three people.

generate counterfactuals for all samples (irrespective of prediction) and analyze the number of times every feature value has changed, as shown in Figure 4. Interestingly, the importances are qualitatively similar to those returned by Python's XGBoost [Chen and Guestrin, 2016] library (also shown in Figure 4). Specifically, feature 5 (BMI) and feature 2 (Glucose) are the most important in predicting diabetes risk. This analysis can be extended to the multi-class case by constraining sampled individuals such that they belong to a desired class (Equation 6)

**Multiple counterfactual explanations**

Multiple explanations are helpful to a user so that they can receive a diverse set of changes that could be made to achieve a desired outcome. The UCI adult dataset (CWC case) is considered and features such as native-country are muted and a set range is given for features like hours-per-week. We run the genetic algorithm for the input instance and select the best two individuals that have different changes in feature indices. The advantage of our approach is that we only need to run the algorithm once, and we can generate many explanations, as opposed to [Russell, 2019] where the IP solver needs to be run multiple times to generate multiple explanations.

To underscore the benefits of suggesting alternative counterfactuals, Table 5 shows two sets of explanations that are generated by CERTIFAI for the same person. Multiple explanations, the number of which is set by the user, allow a user to decide which counterfactual may be the most actionable.

## 4.3 Fairness

We evaluate fairness from an individual's perspective and from a model developer's perspective. To see if the model is unfair towards any instance, we consider 100 random instances of the UCI adult dataset where the prediction was unfavorable and run the algorithm twice, once when the sensitive attribute is not allowed to change and once when it is, and record the fitness values. We do this for two sensitive attributes, race and gender.

The results for three such instances are shown in Table 6. FitnessM refers to the fitness value when the race feature is muted for an individual and FitnessU corresponds to the feature being unmuted. The fitness for the first 2 people increases substantially when these protected features are allowed to change and hence for these instances, there is evidence that the model has not been fair. For the third person, the evidence suggests that the model has been fair.

A model developer can use the idea of burden (Equation 11) to evaluate how fair a model is being to groups of individuals. To demonstrate the idea of burden, we consider

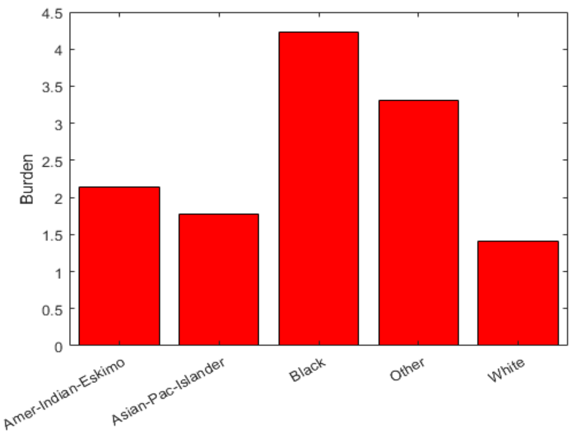

Figure 5: Burden on different groups belonging to a particular race in the UCI adult dataset, found using the distance between the input instances and counterfactuals (Equation 11)

the attribute race in the UCI adult dataset and take all training examples that have an unfavorable outcome. Results of our experiments are shown in Figure 5. As we can see, the burden on Black race and the Other race is more than the other races. This means that on average, these groups would have to make more changes to achieve a desired prediction as compared to others. Hence the model imposes a burden on these groups, which could imply that the model has been unfair.

## 5 Conclusion and Future Work

In this paper, we introduced CERTIFAI, a model-agnostic, flexible, and user-friendly technique that helps build responsible artificial intelligence systems. We demonstrate the flexibility that the genetic algorithm brings to provide feasible counterfacual explanations to a user. We show how individual and group fairness can be measured using the fitness values obtained during counterfactual generation. Finally, we show how these counterfactuals are effective adversarial examples and we define CERScore, the first ever measure of robustness for a black-box model. We are currently developing the User-Interface to CERTIFAI. Future work involves speeding up the genetic algorithm by techniques like [Harik *et al.*, 1999] and [Mitchell *et al.*, 1994] A comparison between the introduced fairness metric and previous metrics would also be useful. It would also be interesting to see how our adversarial examples perform with strategies [Papernot *et al.*, 2016], [Madry *et al.*, 2017] that are aimed to handle adversarial attacks.

## References

[Akhtar and Mian, 2018] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

[Binns, 2017] Reuben Binns. Fairness in machine learning: Lessons from political philosophy. *arXiv preprint arXiv:1712.03586*, 2017.

[Carlini and Wagner, 2017] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks, 2017.

[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system, 2016.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. 2009.

[Donini *et al.*, 2018] Michele Donini, Luca Oneto, Shai Ben-David, John S Shawe-Taylor, and Massimiliano Pontil. Empirical risk minimization under fairness constraints, 2018.

[Guidotti *et al.*, 2018a] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*, 2018.

[Guidotti *et al.*, 2018b] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):93, 2018.

[Hardt *et al.*, 2016] Moritz Hardt, Eric Price, Nati Srebro, et al. Equality of opportunity in supervised learning, 2016.

[Harik *et al.*, 1999] Georges R Harik, Fernando G Lobo, and David E Goldberg. The compact genetic algorithm. *IEEE transactions on evolutionary computation*, 3(4):287–297, 1999.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2016.

[Howard *et al.*, 2017] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[Kohavi, 1996] Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid., 1996.

[LeCun, 1998] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[Madry *et al.*, 2017] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[Mitchell *et al.*, 1994] Melanie Mitchell, John H Holland, and Stephanie Forrest. When will a genetic algorithm outperform hill climbing, 1994.

[Nguyen *et al.*, 2015] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, 2015.

[Papernot *et al.*, 2016] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks, 2016.

[Ribeiro *et al.*, 2016] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier, 2016.

[Russell, 2019] Chris Russell. Efficient search for diverse coherent explanations, 2019.

[Smith *et al.*, 1988] Jack W Smith, JE Everhart, WC Dickson, WC Knowler, and RS Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus, 1988.

[Szegedy *et al.*, 2016] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2016.

[Ustun *et al.*, 2019] Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification, 2019.

[Wachter *et al.*, 2017] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: automated decisions and the gdpr. *Harvard Journal of Law & Technology*, 31(2):2018, 2017.

[Wang *et al.*, 2002] Zhou Wang, Alan C Bovik, and Ligang Lu. Why is image quality assessment so difficult?, 2002.

[Wang *et al.*, 2003] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment, 2003.

[Weng *et al.*, 2018] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv preprint arXiv:1801.10578*, 2018.