Why X rather than Y? Explaining Neural Model' Predictions by Generating Intervention Counterfactual Samples

Thai Le, Suhang Wang, Dongwon Lee
(thaile,szw494,dongwon)@psu.com
The Pennsylvania State University

Abstract

Even though the topic of explainable AI/ML is very popular in text and computer vision domain, most of the previous literatures are not suitable for explaining black-box models' predictions on general data mining datasets. This is because these datasets are usually in high-dimensional vectored features format that are not as friendly and comprehensible as texts and images to the end users. In this paper, we combine the best of both worlds: "explanations by intervention" from causality and "explanations are contrastive" from philosophy and social science domain to explain neural models' predictions for tabular datasets. Specifically, given a model's prediction as label X, we propose a novel idea to intervene and generate minimally modified contrastive sample to be classified as Y, that then results in a simple natural text giving answer to the question **"Why X rather than Y?"**. We carry out experiments with several datasets of different scales and compare our approach with other baselines on three different areas: *fidelity*, *reasonableness* and *explainability*.

Keywords

explainability, counterfactual, data mining

ACM Reference Format:
Thai Le, Suhang Wang, Dongwon Lee. 2018. Why X rather than Y? Explaining Neural Model' Predictions by Generating Intervention Counterfactual Samples. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

1 Introduction

Neural networks, especially deep neural networks, have shown promising results in several domains such as computer vision [10, 12] and natural language processing [4, 24]. However, most neural networks are black-box models [8, 9], which lacks interpretability for end-users to understand the inner working of the model and why a data instance is classified as X rather than Y. The lack of interpretability results in distrust of the model, which obstructs the widely adoption of neural networks in many high-stakes scenarios such as healthcare [1, 18], financial market analysis [6] and traffic light control [26]. Thus, explainable artificial intelligence, which aims at providing easy-to-understand explanations on the inner working or the decision making of black-box models, have attracted increasing attention and many efforts have been taken [2, 8, 11, 14, 17, 19, 22, 29]. For example, Zeiler et al [29] investigated visualization technique that gives insight into the function of intermediate feature layers and the operation of the classifier to understand how convolutional neural network works. [11] and [14] study the visualization with heatmap of words to understanding neural models in natural language processing.

Feat	freq_credit	freq_technology	freq_!	class
$\boldsymbol{x}_1$	0.0	0.0	0.0	ham
$\tilde{\boldsymbol{x}}_1$	0.0	0.0	0.453	spam

Feat	freq_you	freq_direct	avg_longest_capital	class
$\boldsymbol{x}_2$	0.68	0.34	158.0	spam
$\tilde{\boldsymbol{x}}_2$	0.68	0.34	1.0	ham

Table 1: Example of generated contrastive samples on *spambase* dataset. Generated samples are only different from original sample at *single* feature. (unchanged features are randomly selected for illustration purpose)

Despite the initial success of the aforementioned approaches, the majority of them are designed for models trained on images or texts by visualization or approximating the decision boundaries, while little efforts are taken for providing end-users with explanations on neural networks in more general domains. The data representation of general domains are usually high dimensional vector features stored in tabular format, which then challenges many existing explanation algorithms designed for images or texts domains, due to many reasons. *First*, features of general domains (e..g tabular data) usually have high-dimensional features, many of which may have correlations. For each data instance, a complex model can focus on different features for making a decision. To illustrate, Table 1 shows that the model can focus on two different features, $freq_!$ and $avg\_longest\_capital$, to make decision on whether an email is spam or ham. Thus, we need to find instance dependent key features for giving explanations. *Second*, for images or texts, highlighting a patch of an image or a phrase of a sentence usually gives a clear and quick understanding of what the model is focusing on and why the model gives such prediction, as the image patch or phrase itself is self-explainable; while for more general domains, such visualization of features doesn't provide much insight of the model. For example, in the second example in Table 1, the model predict $\boldsymbol{x}$ as spam and the important feature used by a model is $avg\_longest\_capital$. Simply providing this feature to an end-user doesn't give easy-to-understand explanation. *Third*, approximating the decision boundaries cannot give end-users clear understanding of the decision making of the model as end-users in general domains usually lack the background of machine learning. Instead, an end-users are usually *more interested in **contrastive explanation**, i.e. why it is predicted as X rather* *than Y*. For example, table 1 shows that "if $freq_!$, or frequency of "!", was 45.3% higher, the email would have been classified as *spam rather than ham*". Thus, more efforts towards generating explanations that are easy for end-users to understand, especially in general domains, are in great demand. In particular, simple texts explanations of why X rather than Y based on only few instance dependent features will be desired to convince end-users to trust and adopt the model.

Therefore, in this paper, we investigate the novel problem of generating explanations to end-users on neural networks for more general domains. Similar to Database (DB) literatures [15, 20, 27] , we borrow the idea of "explanation by intervention" from causality [13, 21] to come up with a contrastive explanation: why a prediction is classified as X rather than Y. Specifically, we want to intervene and modify a few instance dependent key features to generate new sample such that it is classified to another class. Then, we aim to provide a simple text explanations of *why X rather than Y* basing on the newly generated sample. In general, we are faced with three challenges: (i) for each instance, how to find few instance dependent key features for contrastive sample generation; (ii) how to generate contrastive samples, i.e. ones that are predicted as Y rather than X, based on key features; and (iii) how to further use these samples to provide explanations on why the model predict X rather than Y. In an attempt to solve these challenges, we propose a novel framework, which generates explainable samples and their contrastive labels by selecting and perturbing only a few number of key features under both quantity, domain and correlation constraints, and use those samples to provide simple and intuitive explanation texts for neural networks' predictions. The main contributions of the paper are:

- We introduce an explanation for ML concept by marrying "contrastive explanation" and "explanation by intervention for ML", then extend it to a novel problem of generating contrastive sample to explain why a black-box model predict X rather than Y for data instances with tabular vector features.
- We propose a new *Generating Intervention Counterfactual Sample* (GICS) framework, which is able to find instance dependent key features of a data instance, generate contrastive samples based on these key features and provide explanations on why we predict X rather than Y with these contrastive samples; and
- We conduct extensive experiments on various real-world datasets to demonstrate the effectiveness of the proposed framework for generating reasonable contrastive samples and providing easy-to-understand explanations for black-box models. We show that our approach generates more concise and faithful explanations than the notorious LIME method.

2 Contrastive Explanation by Intervention for ML

Understanding the answer to the question "Why?" is crucial in many practical settings, e.g. in determining why a patient is diagnosed as cancer, why a banking customer should be approved for housing loan, etc. The answers to these "Why?" questions can really be answered by the study of causality, which depicts the relationship between an event and an outcome: the event is a cause if the outcome is the consequence of the event ([15, 21]). However, causality can only be established under a controlled environment, in which one alters a single input while keeping other constant, and observes changes of the output. Bringing causality into data-based studies such as DB or ML is a very challenging task, since causality cannot be achieved by using data alone ([15]). As the first step to understand causality in data-based fields, DB and ML researchers have tried to lower the bar of explanations, aiming to find subset of variables that are best correlated with the outputs. Specifically, DB literatures aim to provide explanations for an complex query outputs given all tuples stored in the database, and ML researchers are keen on explaining predictions of learned, complex models.

By borrowing the notion of *intervention* from causality literature, DB researchers have come up with a practical way of explaining a query result by removing, inserting, updating a set of tuples that satisfy a set of predicates on their attributes such that the output of the query will change ([15, 20, 27]). Similarly, by utilizing the same perspective, we want to formulate a definition of *explanation by intervention* for ML model at instance-level as follows.

DEFINITION 1 (CONTRASTIVE EXPLANATION (IN ML) BY INTERVENTION). *A predicate* **P** *of subset of features is an explanation of a prediction outcome* **X***, if changes of features satisfying the predicate* **P** *also changes the prediction outcome to* Y ≠ X*,* ***while keeping other features unchanged****.*

Given a prediction of a black-box model on an input, there will be possible many predicates **P** satisfying definition 1. Hence, it is necessary to have a measure to describe and compare how much predicate(s) **P** have influence on the final explanation. Following the related literatures of explanation from DB domain [27], we also formally define a scoring function $infl_\lambda(\mathbf{P})$ as measure on the influence of **P** on the explanation with a tolerance level λ as follows.

DEFINITION 2 (INFLUENCE SCORING FUNCTION).

$$infl_\lambda(\mathbf{P}) = \frac{\mathbb{I}(\boldsymbol{Y} \neq \boldsymbol{X})}{(Number\ of\ features\ in\ \mathbf{P})^\lambda} \tag{1}$$

with **X** and **Y** are predicted label before and after intervention respectively. The larger the score is, the more influential **P** has on the explanation. Hence, a $\lambda = 0$ would imply infinite tolerance on the number of features in **P**, $\lambda > 0$ would prefer a small size of **P** and $\lambda < 0$ would prefer a large size of **P**. In practice, a $\lambda > 0$ is preferable because a predicate **P** containing too many features would adversely affect the comprehension of the explanation, especially when being presented to the end-users.

Nevertheless, searching for a set of predicates **P** is a non-trivial problem. Basing on definition 1, we want to approach this problem from a generation perspective. Specifically, we want to intervene and modify a small subset of features of the original sample to generate a new sample that crosses the decision boundary of the black-box model. This subset of features and their new values will result in a predicate **P** that helps explain the black-box model's prediction. In other words, given an arbitrary sample classified as **X**, we want to generate its minimally modified contrastive sample *in terms of number of features* that is predicted as **Y**. This newly generated sample will help give answer to the question "Why X rather than Y?". Two examples for possible predicates to explain a spam black-box classifier are shown in table 1. Specifically, predicate

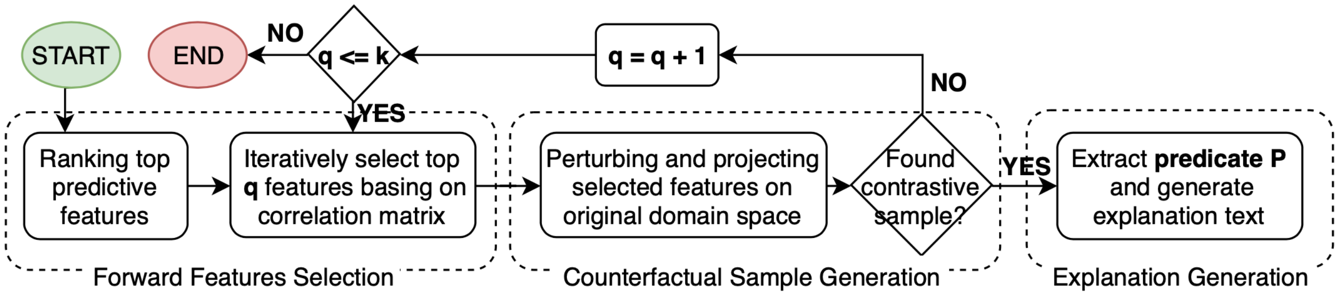

START
END
NO
q <= k
YES
q = q + 1
NO
Ranking top predictive features
Iteratively select top **q** features basing on correlation matrix
Perturbing and projecting selected features on original domain space
Found contrastive sample?
YES
Extract **predicate P** and generate explanation text
Forward Features Selection
Counterfactual Sample Generation
Explanation Generation

Figure 1: Illustration of GICS framework: a greedy approach to generate intervention counterfactual sample

$\mathbf{P}_1$ = (freq_! == 0.453) explains why sample $\boldsymbol{x}_1$ is classified as ham rather than spam, and predicate $\mathbf{P}_2$ = (avg_longest_capital == 1.0) explains why sample $\boldsymbol{x}_2$ is classified as spam rather than ham. $\tilde{\boldsymbol{x}}_1$ and $\tilde{\boldsymbol{x}}_2$ are generated samples that corresponds to predicate $\mathbf{P}_1$ and $\mathbf{P}_2$ respectively.

Based on this framework, Section 3 and 4 formulates this generation-based approach to explain neural network model' predictions in more details. Section 5 further extends the *scoring function* (Def. 2) to derive a more fine-grained influence measure and as well as describes different experimental results in four different areas including: *reasonableness*, *fidelity*, *accuracy* and *explainability* in details. Then, section 6 discusses some related works. Finally, section 7 and 8 concludes the paper with current limitations and potential future works.

3 Problem Formulation

From the introduction, we want to generate counterfactual samples to explain neural network models' predictions using the narrative "Why X rather than Y?". In this section, we want to formulate different characteristics of the samples we want to generate, and as well as to formally formulate our problem statement. Firstly, we want to generated samples that are counterfactual or contrastive. We define such characteristic as follows.

Definition 3 (Counterfatual sample). *Given an arbitrary $\boldsymbol{x} \in \mathcal{X}$, $\mathcal{X} \in \mathbb{R}^{NxM}$ and a ML model $f(\boldsymbol{x})$, $\tilde{\boldsymbol{x}}$ is call counterfactual or contrastive sample of $\boldsymbol{x}$ when*

$$f(\tilde{\boldsymbol{x}}) \neq f(\boldsymbol{x}) \tag{2}$$

As already noticed, above definition is borrowed from *adversarial perturbation* literature, in which **all** of the features are marginally perturbed to cross the decision boundary. However, our approach is different. Specifically, we want to generate $\tilde{\boldsymbol{x}}$ and its label $\tilde{\boldsymbol{y}} = f(\tilde{\boldsymbol{x}})$ such that it is minimally different from original input $\boldsymbol{x}$ in terms of *number of features*. Specifically, we desire to explain "Why X rather than Y?" by presenting an explanation in which if only a few features are corrected, e.g. if frequency of word 'money' is increased to value 0.4 in spam detection problem, the predicted label will change from one class to another, e.g. from "Spam" to "Ham". Hence, the less the number of features needed to change from $\boldsymbol{x}$ to generate $\tilde{\boldsymbol{x}}$, the more comprehensible, or "explainable" the sample becomes. To do this, we impose parameter k as an upper-bound on the number of perturbed features. Let's denote $\mathcal{S}(\tilde{\boldsymbol{x}})$ as the feature selector that returns list of features to perturb and $\mathcal{M}$ is the total number of features in the dataset, we want to:

$$\text{minimize} \quad |\mathcal{S}(\tilde{\boldsymbol{x}})| = \sum^{M} \mathbb{1}[\boldsymbol{x}^{\boldsymbol{m}} \neq \tilde{\boldsymbol{x}}^{\boldsymbol{m}}] \\ s.t. \, |\mathcal{S}(\tilde{\boldsymbol{x}})| \leq k \tag{3}$$

Not only we want to change just a few features, we also want those features to be uncorrelated. For example, the explanation "Had the frequency of "!" and "!!" is more than 0.3, the email would be classified as spam rather than ham" is not as informative as "Had the frequency of "!" and "wonder" is more than 0.3, the email would be classified as spam rather than ham". Intuitively, as much as to ensure that the selected features are prone to a contrastive class, we also want them to be uncorrelated. Because of this, we want $\mathcal{S}(\tilde{\boldsymbol{x}})$ to return a list of features such that the average correlation among pairwise of selected features, denoted as $\mathcal{I}(\tilde{\boldsymbol{x}})$, is minimized:

$$\text{minimize} \quad \mathcal{I}(\tilde{\boldsymbol{x}}) = \frac{1}{|\mathcal{S}(\tilde{\boldsymbol{x}})|^2} \sum_{i \in \mathcal{S}(\tilde{\boldsymbol{x}})} \sum_{j \in \mathcal{S}(\tilde{\boldsymbol{x}})} \text{SU}\,(\mathcal{X}^i, \mathcal{X}^j) \tag{4}$$

with SU() is a *symmetrical* correlation function to be introduced in Section 4.1 and $\mathcal{X}^i$ and $\mathcal{X}^j$ are feature ith and jth respectively in features set $\mathcal{X}$. In case $\mathcal{S}(\boldsymbol{x})$ only returns a single feature, $\mathcal{I}(\tilde{\boldsymbol{x}}) = 0$. This is novel from previous literature, in which they focus more on minimizing the difference between generated and original sample $\delta = \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|$. However, as long as the constraint on the **maximum** number of perturbed features is satisfied, minimizing distance δ will not be necessary making $\tilde{\boldsymbol{x}}$ more self-explanatory to the users, yet equation 8 will be so.

Furthermore, we also need to ensure that the final predicate **P** is realistic. For example, age feature should be in whole number larger than 0). Therefore, we want to generate a "reasonable" $\tilde{\boldsymbol{x}}$ such that it conforms to features domain constraints of the dataset.

From aforementioned analysis, we formulate our problem of generating counterfactual samples to explain black-box model' prediction as follows.

Problem 1. *Let's denote $f(\boldsymbol{x})$ as the black-box model of which predictions we want to explain. $f(x)$ can be any deep learning models (e.g. feed-forward network with multiple hidden layers). Denote $\mathcal{X} \in \mathbb{R}^{NxM} = \{x_1, x_2, ..x_n\}$, $\mathcal{Y} = \{y_1, y_2, ..y_n\}$ as the features and ground-truth labels respectively, data on which we will train $f(\boldsymbol{x})$, resulting in a set of predicted labels $\hat{\mathcal{Y}} = \{\hat{y}_1, \hat{y}_2, ..\hat{y}_n\}$. N, M is the total number of samples and features in the dataset respectively. Given $\boldsymbol{x}$ and the prediction $\hat{\boldsymbol{y}} = f(\boldsymbol{x})$, our goal is to generate new*

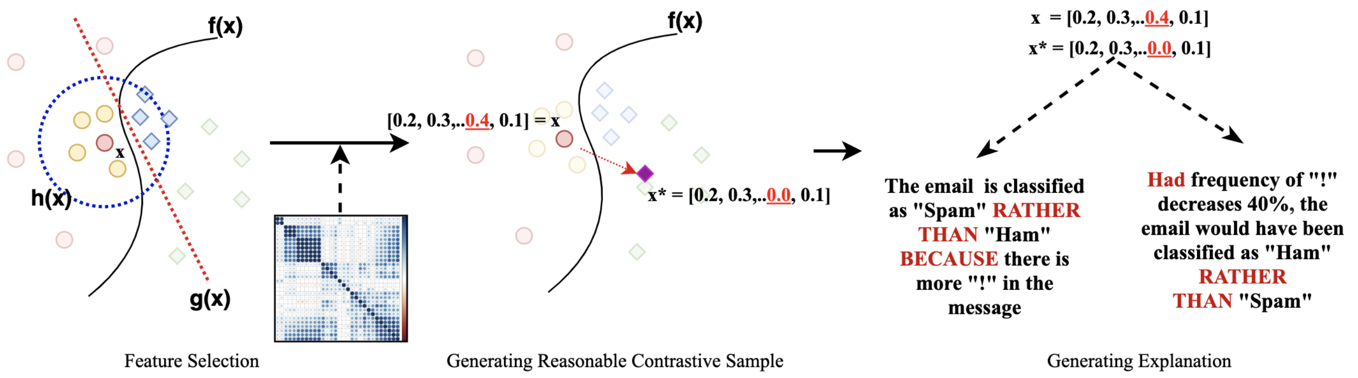

f(x)
h(x)
g(x)
x
Feature Selection
[0.2, 0.3,..0.4, 0.1] = x
f(x)
x* = [0.2, 0.3,..0.0, 0.1]
Generating Reasonable Contrastive Sample
x = [0.2, 0.3,..0.4, 0.1]
x* = [0.2, 0.3,..0.0, 0.1]
The email is classified as "Spam" RATHER THAN "Ham" BECAUSE there is more "!" in the message
Had frequency of "!" decreases 40%, the email would have been classified as "Ham" RATHER THAN "Spam"
Generating Explanation

Figure 2: Illustration of GICS-Local on a binary "Spam" classification task. Solid black line describes decision boundary of a trained black-box $f(x)$ model. Given x and label $"Spam" = f(x)$, its reasonable contrastive sample can be generated in three steps. *Feature Selection:* we use a *sampler function* $\mathcal{H}(x)$ to sample data from all predicted class "Spam" (yellow circles) and "Ham" (blue diamonds) around vicinity of x. Then, we use a *weak classifier* $\mathcal{G}(x)$ to train on the set of data points selected by $\mathcal{H}(x)$ together with x, resulting in a list of ranked features. These features are then filtered out and selected using their correlation matrix to make sure they are highly uncorrelated. *Generating reasonable contrastive sample*: using a customized adversarial generation process to generate contrastive sample $\tilde{x}$ by perturbing *only* the pre-selected k features from *feature selection* step such that $f(\tilde{x}) = "Ham"$. We can see that x^* is only different from x at one feature. *Generating explanations:* we use simple text templates to generate explanation text explaining Why x is classified as "Spam" rather than "Ham"

*counterfactual and reasonable sample $\tilde{\boldsymbol{x}}$ to explain such prediction by changing a **maximum** of k features, i.e. we want to generate $\tilde{\boldsymbol{x}}$ such that satisfies equations (2), (3), and (4).*

In the next next section, we will describe in details on how to generate such generated samples.

4 Generating Reasonable Contrastive Samples

It is observed from our problem (1) definition that directing solving $\tilde{\boldsymbol{x}}$ to satisfy equations (2), (3), and (4) is very challenging. Instead, we employ a greedy approach as depicted in figure 1. Specifically, we first rank all features according to their predictive power, top of which will be more vulnerable to decision boundary than the others. Next, we iteratively select a subset of *maximum* k features from the ranked list while making sure they are highly uncorrelated. Then, we perturb them accordingly by leveraging literature from *adversarial perturbation*, DeepFool [16] algorithm specifically. Throughout these steps, we want to minimize equation (3), (4) and at the same time ensure that counterfactual equation (2) is satisfied. Noticeable, we gradually increase the number of features to selected until reaching the upper-bound k or a sample satisfying all equations (2), (3) and (4) is found. Finally, we extract explanation predicate **P** and use simple text template to generate explanation text to the users to give answers to question "Why X rather than Y?" basing on the difference between $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$. We call this **G**enerating **I**nterventive **C**ontrastive **S**amples (GICS) framework to explain black-box models' predictions.

One first obvious challenge is how to select a sub set of features from $\boldsymbol{x}$ to perturb. The next section will describe this into great details.

4.1 Correlation-Based Forward Feature Selection

As different $\boldsymbol{x}$ might require different sub-set of features to perturb, the first challenge is to select which features to change to ensure not only $\tilde{\boldsymbol{x}}$ will cross the decision boundary of $f(\mathbf{x})$ but also selected features are not highly correlated. We employ a two-step approach for this (Fig. 1). Firstly, we rank all features according to their predictive power by using either *gradient-based* or *local-based* method. Next, we iteratively select top k features according to their ranking while making sure no single pair of the are highly correlated with each other. Figure 2 describes in details GICS with *local-based* predictive features ranking approach.

4.1.1 Ranking Predictive Features This paper introduces two predictive features ranking approach for black-box models. As described below, the *gradient-based* approach is naive and straightforward, we propose another *local-based* approach to introduce more variants into the overall GICS framework. In later experiments, we show that each ranking approach performs differently on various datasets.

(1) **Gradient-based**: The most straight-forward way is to rank all features according to their gradients with respect to the nearest counter-class that back-propagates through $f(\mathbf{x})$.
(2) **Local-based**: We want to approximate the *local* decision boundary of black-box model $f(\mathbf{x})$ in the features space close to $\mathbf{x}$ using an interpretable ML model $\mathcal{G}(\mathbf{x})$ (e.g. Logistic Regression, Decision Tree) by training $\mathcal{G}(\mathbf{x})$ on a subset of data points Q surrounding $\mathbf{x}$. If predictions of $\mathcal{G}(\mathbf{x})$ on Q is very close to that of $f(\mathbf{x})$, the selected set of features are more prone to change in $f(\mathbf{x})$ predictions. To sample set Q, we use a sampler function $\mathcal{H}(\mathbf{x})$ to sample j data points from *each of the predicted classes* by $f(\mathbf{x})$ on the training set. KNN algorithm can serve as the sampler function $\mathcal{H}(\mathbf{x})$ with different distance functions. We set $j = 4$ and use *Euclidean distance* on KNN throughout all experiments. There are many possible candidate for $\mathcal{G}(\mathbf{x})$ such as logistic regression, decision tree, SVM, all of which supports a score-based metric to rank all of the features (model coefficients, Gini score, distance to

decision boundary in cases of logistic regression, decision tree and SVM respectively).

4.1.2 Selecting Informative Features In both of two approaches, each feature is treated as independent with each other. However, if any two selected features are highly correlated, the final generated sample will be less informative.

From the problem definition (1), we further adjust the set of selected features basing on their correlation measures using forward selection approach. Specifically, given a list of ranked features according to their predictive powers and a features correlation matrix, we iteratively select one by one feature from the ranked list and disregard any subsequent features that are highly correlated with the selected ones using a correlation threshold $\boldsymbol{\gamma}$. Any values $\boldsymbol{\gamma} < 1.0$ ensures the correlations of any two features are bounded by $\boldsymbol{\gamma}$. In this case, we utilize an entropy-based *symmetrical uncertainty* [7] to calculate the correlation between features I and J as follows.

$$SU(\mathcal{X}^i, \mathcal{X}^j) = 2\left[\frac{IG(\mathcal{X}^i \mid \mathcal{X}^j)}{H(\mathcal{X}^i) + H(\mathcal{X}^j)}\right] = [0, 1] \qquad (5)$$

where $IG(\mathcal{X}^i \mid \mathcal{X}^j)$ is the *information gain* of $\mathcal{X}^i$ given $\mathcal{X}^j$, and $H(\mathcal{X}^i)$, $H(\mathcal{X}^j)$ is empirical entropy of $\mathcal{X}^i$ and $\mathcal{X}^j$ respectively. A value $SU = 0$ indicates the two features is independent and a value of $SU = 1$ indicates that $\mathcal{X}^i$ completely predicts the value of $\mathcal{X}^j$ [28]. In fact, *symmetrical uncertainty* does not assume the type of *linear correlation* as in using *linear correlation coefficients*, can work with *continuous features*, and it also takes into account the bias effects in which one features might have more different values than the other [28].

4.2 Counterfactual Sample Generation with Domain Constraints

Since our approach borrows ideas from adversaries generation, we generate our contrastive explainable samples by introducing various constraints specified in Section 3 into an adversarial generation algorithm. Particularly, we use *Projected Gradient Descent* to enforce those constraints. We further extend DEEPFOOL[16] algorithm, a simple and effective method in generating counter-class samples.

Specifically, we want to generate $\tilde{\boldsymbol{x}}$ close to the decision boundary of $f(\boldsymbol{x})$ that it is most vulnerable to. However, instead of changing all features of $\boldsymbol{x}$ as in *adversarial perturbation* literatures, we utilize DEEPFOOL[16] algorithm to generate $\tilde{\boldsymbol{x}}$ but by perturbing *only* variables returned by the aforementioned features selection step. Moreover, feature perturbation basing on gradient back-propagated by $f(\boldsymbol{x})$ does not always guarantee that resulted $\tilde{\boldsymbol{x}}$ still maintains in the original feature space. Thus, we project those adjusted features back on the original domain space of those features. This helps ensure that final $\tilde{\boldsymbol{x}}$ looks more real (e.g. age feature should be whole number and > 0). The domain space constraints, including maximum, minimum values of each features and as well as their data type (e.g. int, float, etc.) can be easily calculated from the original training set or manually set by domain experts. We further assign the class that $\tilde{\boldsymbol{x}}$ is most prone to change as its generated label.

To take the optimization of eq. (3) into consideration, we deploy a simple greedy loop over the features selection and counterfactual sample generation steps (Fig. 1). We first try to generate $\tilde{\boldsymbol{x}}$ by perturbing only 1 feature, and gradually increase the number of key features to be selected, i.e. $\mathbf{q}$ in Fig. 1, until it reaches the upper-bound k or $\tilde{\boldsymbol{x}}$ crosses the decision boundary. While this is not directly optimizing condition (3), later experiments show that this simple approach is able to generate $\tilde{\boldsymbol{x}}$ by changing a very few number of key features across various datasets.

4.3 Generating Explanation Text to the Users

Even though the generated counterfactual samples are already intuitive and self-explainable to the users (e.g. Table 1), we take a further step and use those samples to generate prediction explanation in natural text. Table 3 shows generated contrastive samples and as well as explanations for various datasets with $k = 5$.

To do this, for a specific prediction $\hat{\mathbf{y}} = f(\mathbf{x})$ and generated reasonable contrastive sample $\tilde{\mathbf{x}}$, we first calculate their feature differences, resulting in predicate $\mathbf{P}$ as defined in Def. 1 that explains such prediction. Then, we can translate $\mathbf{P}$ to text by using simple condition-based text templates such as *if...then...classified as X RATHER THAN Y*, or *had...,it would be classified as X RATHER THAN Y.* Different text templates can be selected randomly to induce diversity in explanations text. The difference in features values can be described in three different degrees of obscurity from (i) *extract value* (e.g. 0.007 point lower), to (ii) *magnitude comparison* (e.g. twice as frequent), or (iii) *relative comparison* (e.g. higher, lower). Which degree of details to best to use is highly dependent on specific feature, domain and as well as the choice of end-user, yet they do not need to be consistent among perturbed features in a single explanation text.

5 Experiments

In this section, we experiment our approach on various problem domains and draw comparison with other baselines on three aspects: *fidelity*, *reasonableness* and *explainability*. Since section 4.1.1 introduces two methods of ranking predictive features, namely *gradient-based* and *local-based*, we experiment our framework with such two variants and call them GICS-GRADIENT and GICS-LOCAL respectively.

5.1 Datasets

Table 2 describes different datasets, their statistics that we use throughout the paper. These datasets are retrieved from UCI Machine Learning Repository [5], and are from a variety of domains. We further group our datasets into three scale levels according to the total number of features, and split each of them into train, development (dev), and test set with ratio 80%:10%:10% respectively.

Even though there are many complex deep learning models proposed to tackle problems required raw input data such as texts, images and audios, Table 2 shows that simple feed-forward neural networks (e.g. logistic regression in its simplest form), can also perform well on tabular datasets as well. For the sake of simplicity, we utilize simple 2-hidden-layers fully-connected neural networks as the black-box model whose predictions we want to explain. We first set the number of nodes in the hidden layers as the number of the features, then using dev set to gradually adjust and find the best settings. We also use early-stopping criteria with *Cross Entropy Loss* on dev set to prevent overfitting. We use such model to train on the train set and test on the test set and report both *average accuracy* and *weighted average F1 score* across 10 different runs. Since our approach leverages neural network model to generate contrastive

Table 2: Details of Experiment Datasets and Prediction Performance on Test Set

# of Features	Dataset	# Classes	# of Features	# of Samples	# of Train	# of Validation	# of Test	Avg. Accuracy	Avg. F1
Less than 30	eegeye	2	14	14980	12133	1349	1498	0.858	0.858
	diabetes	2	8	768	621	70	77	0.779	0.777
	cancer95	2	9	699	566	63	70	0.963	0.963
	phoneme	2	5	5404	4376	487	541	0.774	0.772
	segment	7	19	2310	1871	208	231	0.836	0.817
	magic	2	10	19020	15406	1712	1902	0.862	0.859
Between 30 and 100	biodeg	2	41	1055	854	95	106	0.853	0.851
	spam	2	57	4601	3726	414	461	0.932	0.932
	cancer92	2	30	569	460	52	57	0.958	0.958
More than 100	mfeat	10	216	2000	1620	180	200	0.943	0.936
	musk	2	166	476	385	43	48	0.783	0.789

Table 3: Examples of Generated Contrastive Samples and Their Explanation Texts

Dataset	Features/*Prediction*	Type	Original	Generated	Changes	Explanation Text
cancer95	bare_nuclei *diagnosis*	int	1 Benign	10 Malignant	↑ 9	"if there were 9 more bare nucleus, the patient would have been classified as malignant *RATHER THAN* benign"
cancer92	smoothness_worst *diagnosis*	float	0.149 Malignant	0.142 Benign	↓ 0.007	"had the largest mean value for local variation in radius lengths were 0.007 point lower, the patient would be diagnosed as *benign RATHER THAN malignant* for breast cancer"
spam	word_freq_credit word_freq_money *class*	float float	0.470 0.470 Spam	0.225 0.190 Ham	↓ 0.245 ↓ 0.280	"The message is classified as spam *RATHER THAN* ham because the word 'credit' and 'money' is used as twice as frequent as that of ham message"

samples, it is also applicable for a variety of datasets and other neural network architectures as well.

5.2 Compared Methods

Since our proposed framework combines the best of both world: adversarial generation and black-box model explanation, we select various relevant baselines from the two aspects.

- NEARESTCT: Instead of generating synthetic contrastive sample for explanation for data point $\boldsymbol{x_i}$, its the *nearest* contrastive sample from the *training set* can be be selected to provide contrastive explanation for the prediction $f(x_i)$. We called this trivial baseline *Nearest Contrast* (NEARESTCT)
- DEEPFOOL[16]: An effective approach that was originally proposed for untargeted attack by generating adversarial samples. Since our generation approach bases on DEEPFOOL, we use the same hyper-parameters (*overshoot=0.02* and *max_iters = 200*) to do experiments. Even though DEEPFOOL is not designed for generating samples to explain predictions, we consider this as a baseline that intervenes on *every features* to generate counterfactuals.
- LIME [19]: A *local interpretable model-agnostic explanations* approach that provides explanation for individual prediction. This approach replies on visualization of feature importance scores (for text and tabular data), and as well as feature heatmap (for image data) to deliver explanation. We apply this approach on our tabular datasets to do comparison. We use an out-of-the-box toolkit[1] to run experiments for comparison. LIME is selected mainly due to its popularity as a baseline for ML explanation approach.

[1]https://github.com/marcotcr/lime

5.3 Quantitative Analysis & Results

In this section, we want to quantitatively examine the explanation of neural model' predictions on all data points of the test set using GICS-GRADIENT, GICS-LOCAL, DEEPFOOL, LIME, and NEARESTCT methods. We experiment average all of the results across 3 runs for each of the methods. We set $\boldsymbol{k} = 5, \boldsymbol{\gamma} = 0.5$ in all of the experiments. For fair comparisons, we place an upper-bound on the number of iterations in which $\tilde{\boldsymbol{x}}$ is adjusted during the generation to 200 steps in both DEEPFOOL, GICS-GRADIENT and GICS-LOCAL.

To thoroughly examine the proposed approach, we come up with three different analytical questions (AQs) to assess the effectiveness of GICS in comparison with the baselines thereof as follows.

AQ.1 **Fidelity**: How accurate are the labels of generated samples in terms of the black-box model's decision boundary, i.e. how faithful their labels are according to the black-box model?
AQ.2 **Reasonableness**: How similar are generated samples to the original data
AQ.3 **Explainability**: derived from Def. 2, how much explainability the generated samples give answer to questions: "*Why X rather than Y?*.

Since only DEEPFOOL and NEARESTCT are generation-based explanation methods, we only compare and analyze LIME w.r.t *explainability* (AQ.3) in a case study (Section 5.4).

Table 4: Experimental Results on Data Mining Datasets.

		# Features < 30						30 ≤ # Features < 100			100 ≤ # Features	
Statistics	Dataset	**eegeye**	**diabetes**	**cancer95**	**phoneme**	**segment**	**magic**	**biodeg**	**spam**	**cancer92**	**mfeat**	**musk**
$\mathcal{R}_{\text{fidelity}}$	NearestCT	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	DeepFool	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	GICS-Local	**1.0**	0.99	0.95	**1.0**	0.65	0.93	0.71	0.79	0.44	0.48	0.55
	GICS-Gradient	**1.0**	0.92	0.89	**1.00**	0.39	0.92	0.93	**1.00**	0.39	0.79	0.70
$\mathcal{R}_{\text{avg\#Feats}}$	NearestCT	13.56	6.93	5.92	4.82	16.10	9.97	20.53	17.50	29.97	204.22	147.86
	DeepFool	14.00	8.00	9.00	5.00	19.00	10.00	41.00	57.00	30.00	216.00	166.00
	GICS-Local	1.15	**1.55**	2.7	**1.25**	**2.42**	1.68	3.07	2.95	**3.95**	3.28	3.74
	GICS-Gradient	**1.0**	1.96	**2.66**	1.3	3.84	**1.6**	**1.93**	**1.09**	4.5	**2.76**	**2.85**
$\mathcal{R}_{\text{domain}}$	NearestCT	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	DeepFool	**1.0**	0.87	0.89	**1.0**	0.84	0.97	0.76	0.65	0.58	0.99	0.96
	GICS-Local	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	GICS-Gradient	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
$\mathcal{R}_{\text{informative}}$	NearestCT	0.69	0.44	0.64	0.12	0.19	0.04	0.44	0.62	0.02	0.58	0.28
	DeepFool	**0.7**	0.47	**0.69**	0.12	0.39	0.05	0.75	0.81	0.02	0.60	0.30
	GICS-Local	0.64	**0.79**	0.51	**0.81**	**0.84**	0.72	0.65	0.59	**0.28**	**0.7**	0.54
	GICS-Gradient	0.64	0.68	0.60	0.79	0.59	**0.77**	**0.82**	**0.95**	0.1	0.64	**0.58**
$\mathcal{R}_{\text{reasonable}}$	NearestCT	0.33	0.60	9.31	13.64	0.12	0.04	1.46	0.09	0.00	0.00	0.00
	DeepFool	0.32	0.46	6.61	15.23	0.13	0.04	0.87	0.02	0.00	0.00	0.00
	GICS-Local	1.57	**4.82**	16.83	**335.83**	**3.28**	3.65	10.69	0.33	**0.11**	**0.27**	0.13
	GICS-Gradient	**3.92**	2.96	**18.17**	297.91	1.61	**4.31**	**25.15**	**2.24**	0.05	**0.27**	**0.18**
$\mathcal{R}_{\text{explainability}}$	NearestCT	0.33	0.60	9.31	13.64	0.12	0.04	1.46	0.09	0.00	0.00	0.00
	DeepFool	0.32	0.46	6.61	15.23	0.13	0.04	0.87	0.02	0.00	0.00	0.00
	GICS-Local	1.57	**4.78**	15.91	**335.83**	**2.17**	3.41	7.64	0.26	**0.05**	0.13	0.07
	GICS-Gradient	**3.92**	2.73	**16.03**	297.2	0.64	**3.95**	**23.5**	**2.23**	0.02	**0.21**	**0.13**

Results in **bold** and underline indicates the best and second best result for each statistic.

5.3.1 *AQ.1: Fidelity*

By imposing both features and domain constraints (eq. (3) and (4) respectively) during generation, it is not always guaranteed that the generated samples will cross the decision boundary of the black-box model. Therefore, it is necessary to assess how accurate the generated contrastive samples are according to the black-box model's decision boundary, i.e. whether they are really classified as contrastive samples. This is called $\mathcal{R}_{\text{fidelity}}$ score. In fact, this is the accuracy of generated samples' labels w.r.t their predictions by the black-box model:

$$\mathcal{R}_{\text{fidelity}} = \frac{1}{N}\sum^{N} \mathbb{1}[\tilde{\boldsymbol{y}} = f(\tilde{\boldsymbol{x}})] \tag{6}$$

Table 4 describes the $\mathcal{R}_{\text{fidelity}}$ score of all methods across all datasets. NearestCT always have the perfect $\mathcal{R}_{\text{fidelity}}$ score because it selects samples directly from contrastive classes. Since DeepFool is free to change every features without any constraints, it shows to be an effective method to generate counterfactual samples. Different from these two baselines, GICS-Gradient and GICS-Local has has to satisfy the domain constraints, minimize number of features, and as well as features correlations. Experiments show that the first 2 conditions impose the most adversarial effects on $\mathcal{R}_{\text{fidelity}}$. Nevertheless, they still perform reasonably well in most of the datasets, showing an average $\mathcal{R}_{\text{fidelity}}$ of around 80% for both GICS-Gradient and GICS-Local.

5.3.2 *AQ.2: Reasonableness*

First we define reasonableness score of a set of generated samples as $\mathcal{R}_{\text{reasonable}}$ score. Given each data point $\boldsymbol{x} \in \mathcal{X}$, we want to generate a set of counterfactual explainable samples $\tilde{\mathcal{X}}$ with desired properties (i) they are still within the domain of $\mathcal{X}$ and (ii) perturbed features are informative or uncorrelated with each others. These conditions are quantified by $\mathcal{R}_{\text{domain}}$, and $\mathcal{R}_{\text{informative}}$. These three conditions are realized by equation (7) and (8) as follows.

$$\mathcal{R}_{\text{domain}} = \frac{1}{N}\sum_{\tilde{\boldsymbol{x}} \in \tilde{\mathcal{X}}} \mathbb{1}[\tilde{\boldsymbol{x}} \in \text{dom}(\mathcal{X})] \tag{7}$$

$$\mathcal{R}_{\text{informative}} = (1 - \frac{1}{N}\sum_{\tilde{\boldsymbol{x}} \in \tilde{\mathcal{X}}} \mathcal{I}(\tilde{\boldsymbol{x}})) \tag{8}$$

$$\mathcal{R}_{\text{reasonable}} = \mathcal{R}_{\text{domain}}\ \mathcal{R}_{\text{informative}} \tag{9}$$

Table 4 summaries the results. As expected, DeepFool performed worst on $\mathcal{R}_{\text{domain}}$ since the generation can moving $\tilde{\boldsymbol{x}}$ much further away from the original distribution. Because of correlation-aware feature selection mechanism, GICS-Gradient and GICS-Local were able to select uncorrelated features, hence was able to generate samples that are much more informative compared when such mechanism is not available (e.g. DeepFool) in most of the datasets.

Overall, GICS-Gradient and GICS-Local was designed by taking "reasonableness" of generated samples into consideration, hence they consistently outperform their counterparts in $\mathcal{R}_{\text{reasonable}}$ by large margins. This shows that samples generated proposed by our framework it not only contrastive but also reasonable.

5.3.3 AQ.3: explainability Extended from influence score with tolerance parameter $\lambda = 1$ (Def. 2), explainability is to measure how well the generated samples can explain the a specific prediction, or give answer to question "Why X rather than Y?". In our problem formulation, influence is indeed proportional to how faithful the generated samples are to the black-box model's decision boundary ($\mathcal{R}_{\text{fidelity}}$), how comprehensible in terms of number of perturbed features ($\mathcal{R}_{\text{avg\#Feats}}$), and also how real they are what they claim to be ($\mathcal{R}_{\text{reasonableness}}$). Because of this, we further quantify explainability as $\mathcal{R}_{\text{explainability}}$ as follows:

$$\mathcal{R}_{\text{avg\#Feats}} = \frac{1}{N} \sum_{\tilde{\boldsymbol{x}} \in \tilde{X}} |\mathcal{S}(\tilde{\boldsymbol{x}})| \tag{10}$$

$$\mathcal{R}_{\text{explainability}} = \frac{\mathcal{R}_{\text{fidelity}} \, \mathcal{R}_{\text{reasonableness}}}{\mathcal{R}_{\text{avg\#Feats}}} \tag{11}$$

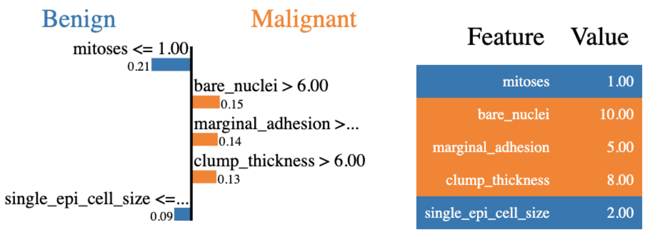
Intuitively, $\mathcal{R}_{\text{explainability}}$ describes the capability to generate new counterfactual samples that captures the two most important qualities of generated samples, namely "contrastive" and "reasonable" by changing a minimal number of features. Hence, the larger the score the better. Table 4 describes all the results.

Regarding $\mathcal{R}_{\text{avg\#Feats}}$, our GICS framework dominates the two baselines DEEPFOOL and NEARESTCT by large margin. Specifically, our approach is able to generated counterfactual samples with much less number of perturbed features, averaging of around less than 2.5 features across all of the datasets. Interestingly, GICS with GICS-GRADIENT was able to select on average less than 3 features out of a total 216 and 166 features in case of *mfeat* and *musk* dataset respectively, while making sure that no single selected pairs of features are highly uncorrelated (correlation < threshold $\boldsymbol{\gamma} = 0.5$

Regarding explainability, our framework is able to generate highly more explainable counterfactual samples than DEEPFOOL, NEARESTCT even when taking $\mathcal{R}_{\text{fidelity}}$ into account, which is usually the strong point of approaches basing on *adversarial perturbation* literature.

Since LIME is not generation-based explanation algorithm, it is very challenging to come up with quantitative comparisons. To compare explainability between approach and Lime baseline, we next present a qualitative case study in the following section.

5.4 Case Study



Feature	Value
mitoses	1.00
bare_nuclei	10.00
marginal_adhesion	5.00
clump_thickness	8.00
single_epi_cell_size	2.00

Figure 3: Explanation produced by LIME

A thorough user-experiment to compare between the two methods is our ultimate goal, yet we want to first draw some observations on the merits and dis-merits of two methods. In this section, we want to compare our GICS framework with LIME [19] in terms of *explainability* from the users' perspectives. While our approach is built specifically for neural network black-box models, LIME works with ML models that provide prediction probabilities. However, both methods are *instance-based* explanation model, or developed to explain individual predictions. *Overall, our approach provides more concise, faithful explanations, and is a better fit to explain individual black-box model's predictions.*

We select *cancer95* dataset to experiment. We first train a neural-network model using the train set, and use samples from test set to do predictions. We sequentially apply LIME and GICS on the trained model to explain such predictions. We set parameter *top_labels*=1 and ask LIME to return the top 5 most important features. Figure 3 depicts explanation produced by LIME on a patient predicted diagnosed as Malignant by the model. From Fig. 3, explanation from LIME can be interpreted as *"if bare_nuclei is less than or equal 6.0, on average, this prediction would be 0.15 less Malignant"*. With the same prediction, GICS generates an explanation text as follows:*"Had bare_nuclei is 7.0 point lower and clump_thickness is 9.0 point lower, the patient would have been diagnosed as* ***Benign rather than Malignant****"*

As we can see, explanation text generated by our framework does not introduces foreign scores as found in LIME' interpretation (e.g. 0.15 less Malignant), hence makes the explanation more comprehensible for the end users. While both methods provide some intuition on the feature importance, importance score provided by LIME is *only* a local approximation of the black-box model, while the features and the decision threshold provided by GICS (e.g. 7.0 point lower for bare_nuclei) is faithful to the black-box model (*fidelity score* is 1.0). From figure 3 it is not very clear which and how single or combinations of features are vulnerable to the contrastive class, but this is very vivid in case of GICS. In summary, explanation generated by our approach is more concise and faithful to the decision boundary of the black-box model, hence GICS framework is a better fit for sample-level explanation.

5.5 Parameter Sensitivity Analysis

5.5.1 Effects of K One of the important factor that largely affect the explainability of the approach is the value of parameter k, or the **maximum** number of features allowed to be change during the contrastive samples generation process. While a small k is more preferable, it would become more challenging for the algorithm to ensure perturbed data points really cross the decision boundary, especially in case of large datasets (e.g. more than 100 features). This will eventually hurt the $\mathcal{R}_{\text{fidelity}}$. In this section we want to see how different k responses to the generated samples' *fidelity* and the number of perturbed features. For each dataset, we next train a neural network model and use this trained model for testing with all values of k = *{1, 2, 3,.. 10}* and plot it against respective $\mathcal{R}_{\text{fidelity}}$ and $\mathcal{R}_{\text{avg\#Feats}}$. Figure 4 summaries the results. We report two distinctive patterns between GICS-GRADIENT and GICS-LOCAL from various dataset as in Figure 4: (i) both approaches witness gradually increment in *fidelity score* and *Avg#Feats*, with either one of them dominates the performance (e.g. *musk, cancer92* dataset), (ii) one of them greatly out-weights the other (e.g. *spam, page* dataset). Overall, by increasing k value, generated samples are more faithful to the black-box model' decision boundary, yet the average number of

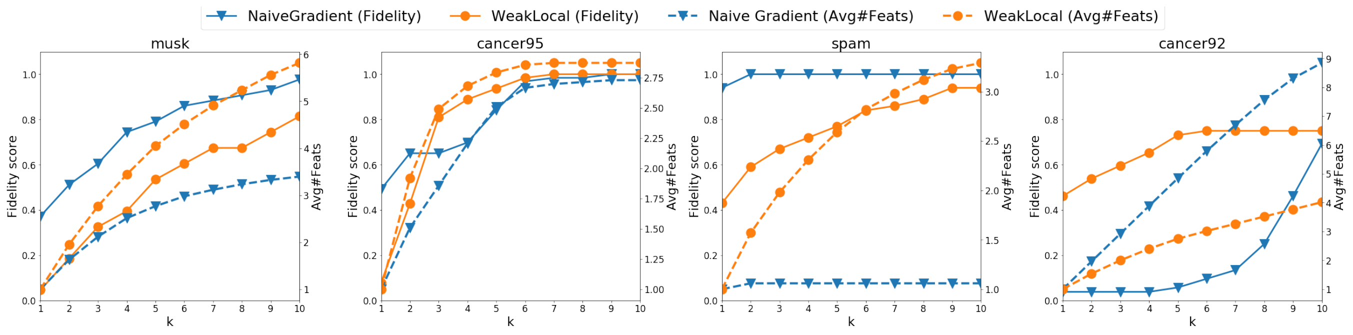


Figure 4: Effects of Parameter k on *Fidelity* and $\mathcal{R}_{\text{avg\#Feats}}$ Score

Table 5: Effects of Parameter γ on Performance of GICS w.r.t $\mathcal{R}_{\text{informative}}$ score. $\gamma = 1.0$ means the correlation-based features selection is disabled.

Dataset	Method	1.0	0.7	0.5	0.3
musk	GICS-Gradient	0.51	0.51	**0.58**	**0.58**
	GICS-Local	0.36	0.36	**0.54**	**0.54**
segment	GICS-Gradient	0.57	0.57	**0.59**	**0.59**
	GICS-Local	0.79	0.79	**0.84**	**0.84**

features needed to change to achieve so also increases, hence reduce explainability (Figure 4).

5.5.2 *Effects of Correlation Threshold γ*

Correlation threshold is set to ensure that no pairs out of selected features are highly correlated, hence making generated samples more informative to the users (Section 4.1.2). Similar with other experiments, we keep other parameters the same while changing $\gamma = \{1.0, 0.7, 0.5\}$ and report the changes in $\mathcal{R}_{\text{informative}}$ score. We observed that γ is not very sensitive, showing the best value of $\gamma \leq 0.5$, which can be explained that features are usually more or less correlated at specific level. However, by setting $\gamma = 0.5$, we can observe larger improvement on $\mathcal{R}_{\text{informative}}$ in some datasets as described in table 5.

6 Related works

Regarding *explanation by intervention*, our definition (1) relates to *Quantitative Input Influence* [3], a general framework to quantity the influence of a set of inputs on outcomes. However, [3] proposes two separate steps approach: (i) changes each individual feature by replacing it with random value, and (ii) *then* see how the outcome, i.e. prediction, changes accordingly, while our approach proposes a more systematic way by generating a whole new sample conditioned on outcome changes (*X rather than Y*).

A few of previous literatures (e.g. [25, 30]) largely propose to generate counterfactual sample with (i) minimal changes or correction from its original input by minimizing the distance: $\delta = \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|$ and with (ii) minimal number of features needed to change to achieve the aforementioned correction. While [25] tries to use δ with ℓ_1 norm to induce sparsity with the hope that (ii) can be achieved, [30] approaches the problem in a reverse fashion in which it tries to search for minimal δ w.r.t to a pre-defined n number of features to be changed. Regardless, minimizing δ distance does not always guarantee that the generated samples will be look as similar as other samples in the same contrastive class. [23] also proposed a method to use decision tree to search for a decisive threshold on feature's values at which the it will change prediction, and use such threshold to generate explanation for black-box model' predictions. While sounds similar to our approach, explanation produced on such threshold and feature is only an approximation and not faithful to the black-box model. As described in the case study (Section 5.4), explanations produced by [19] also share the same dis-merits. In this paper, we take a novel approach to generate counterfactual samples that are not only counterfactual, but also both "real" and faithful to the black-box model's prediction.

7 Limitation & Future Direction

Our work inhibits different limitations: (i) our method requires input features to be interpretable. While this is not always true, for example in case the input features are word-embedding vectors instead of bag-of-words features, semantic inputs are still very important to many ML tasks, (ii) other features selection and ranking methods should be tested and analyzed, (iii) this paper only focuses on the first step of a intervention-based explanation approach: intervening a subset of selected features without considering dependencies among all inputs after such intervention. This might create undesirable samples that are unrealistic (e.g. 5 years old patient with 6" in height) . As a next step, we plan to incorporate conditional dependencies among features to make sure that the generated samples are *more reasonable and realistic*. In other words, we want to consider dependencies of more than just pairwise of features during features selection, and to extend the current intervention predicate **P** to include other dependent features as well. Moreover, we also plan to carry out qualitative experiments with user-studies to gain more insights on how end-users can best perceive explanation from a black-box ML model, and as well as to extend our framework to deal with other types of data such as categorical, sequence, etc.

8 Conclusion

In this paper, we borrow "contrastive explanation" and "explanation by intervention" concepts from previous literatures, basing on which we present a generative-based approach to explain black-box model's predictions. We introduce a sample-based explanation method called GICS that can generate counterfactual and reasonable synthetic samples. By extending *adversarial perturbation* literature with various conditions and constraints, GICS was able to generate samples that can facilitate explanation: *"Why X rather than Y""* for a specific prediction.

References

[1] Babak Ehteshami Bejnordi, Mitko Veta, Paul Johannes Van Diest, Bram Van Ginneken, Nico Karssemeijer, Geert Litjens, Jeroen AWM Van Der Laak, Meyke Hermsen, Quirine F Manson, Maschenka Balkenhol, et al. 2017. Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. *Jama* 318, 22 (2017), 2199–2210.

[2] Lingyang Chu, Xia Hu, Juhua Hu, Lanjun Wang, and Jian Pei. 2018. Exact and consistent interpretation for piecewise linear neural networks: A closed form solution. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1244–1253.

[3] Anupam Datta, Shayak Sen, and Yair Zick. 2016. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE symposium on security and privacy (SP)*. IEEE, 598–617.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[5] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[6] Thomas Fischer and Christopher Krauss. 2018. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research* 270, 2 (2018), 654–669.

[7] Brian P Flannery, Saul A Teukolsky, William H Press, and William T Vetterling. 1988. *Numerical recipes in C: The art of scientific computing*. Vol. 2. Cambridge University Press.

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

[9] David Gunning. 2017. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web* 2 (2017).

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[11] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078* (2015).

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[13] David Lewis. 2013. *Counterfactuals*. John Wiley & Sons.

[14] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066* (2015).

[15] Alexandra Meliou, Sudeepa Roy, and Dan Suciu. 2014. Causality and explanations in databases. *Proceedings of the VLDB Endowment* 7, 13 (2014), 1715–1716.

[16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2574–2582.

[17] Vitali Petsiuk, Abir Das, and Kate Saenko. 2018. Rise: Randomized input sampling for explanation of black-box models. In *BMVC*.

[18] Daniele Ravì, Charence Wong, Fani Deligianni, Melissa Berthelot, Javier Andreu-Perez, Benny Lo, and Guang-Zhong Yang. 2016. Deep learning for health informatics. *IEEE journal of biomedical and health informatics* 21, 1 (2016), 4–21.

[19] 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, San Francisco, CA, USA, August 13-17, 2016*. 1135–1144.

[20] Sudeepa Roy and Dan Suciu. 2014. A formal approach to finding explanations for database queries. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 1579–1590.

[21] Craig Silverstein, Sergey Brin, Rajeev Motwani, and Jeff Ullman. 2000. Scalable techniques for mining causal structures. *Data Mining and Knowledge Discovery* 4, 2-3 (2000), 163–192.

[22] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).

[23] Jasper van der Waa, Marcel Robeer, Jurriaan van Diggelen, Matthieu Brinkhuis, and Mark Neerincx. 2018. Contrastive explanations with local foil trees. *arXiv preprint arXiv:1806.07470* (2018).

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[25] 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.

[26] Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui Li. 2018. Intellilight: A reinforcement learning approach for intelligent traffic light control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2496–2505.

[27] Eugene Wu and Samuel Madden. 2013. Scorpion: Explaining away outliers in aggregate queries. *Proceedings of the VLDB Endowment* 6, 8 (2013), 553–564.

[28] Lei Yu and Huan Liu. 2003. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)*. 856–863.

[29] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*. Springer, 818–833.

[30] Xin Zhang, Armando Solar-Lezama, and Rishabh Singh. 2018. Interpreting neural network judgments via minimal, stable, and symbolic corrections. In *Advances in Neural Information Processing Systems*. 4874–4885.