A Step towards Global Counterfactual Explanations: Approximating the Feature Space through Hierarchical Division and Graph Search

Maximilian Becker, maximilian.becker@kit.edu*,
Nadia Burkart, nadia.burkart@iosb.fraunhofer.de†,
Pascal Birnstill, pascal.birnstill@iosb.fraunhofer.de†,
Jürgen Beyerer, juergen.beyerer@iosb.fraunhofer.de*†
*Vision and Fusion Lab, Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology (KIT)
†Fraunhofer Institute for Optronics, System Technologies and Image Exploitation IOSB

***Abstract*—The field of Explainable Artificial Intelligence (XAI) tries to make learned models more understandable. One type of explanation for such models are counterfactual explanations. Counterfactual explanations explain the decision for a specific instance, the factual, by providing a similar instance which leads to a different decision, the counterfactual. In this work a new approaches around the idea of counterfactuals was developed. It generates a data structure over the feature space of a classification problem to accelerate the search for counterfactuals and augments them with global explanations. The approach maps the feature space by hierarchically dividing it into regions which belong to the same class. It is applicable in any case where predictions can be generated for input data, even without direct access to the model. The framework works well for lower-dimensional problems but becomes unpractical due to high computation times at around 12 to 15 dimensions.**

***Index Terms*—XAI, Counterfactual Explanations, Global Explanations.**

I. Introduction

Machine learning and Artificial Intelligence have become a hot topic in recent years. Learning models have achieved unprecedented results in many tasks such as image recognition, language translation and a variety of classification tasks. However there is still some hesitance in the adoption of learning systems, especially in critical areas such as medicine or finance. One reason that hinders the deployment of machine learning models in such areas is a lack of trust. This distrust is caused in part by missing transparency of learning models. Because of their complexity most models, especially deep neural networks, can not explain their decisions and even experts are unable to understand the model's inner workings. The newly emerging field of Explainable Artificial Intelligence or XAI[1], tries to overcome this problem by making learned models more understandable. A variety of different approaches for explaining learned decisions have been proposed. Some try to explain the model as a whole or completely replace it with an inherently understandable model, like for example a decision tree[2]. Other approaches try to steer the model during the learning process to a more explainable state or focus on explaining single predictions for example by highlighting important features[3] or contrasting them to other decisions[4]. One of the latter type of explanation are counterfactuals.

In psychology counterfactual thinking refers to the imagination of an alternative reality that differs to the actual reality in one or multiple specific aspects[5]. In machine learning counterfactuals explain a models decision for a specific instance, the factual, by providing a similar instance which leads to a different decision, the counterfactual. By comparing the factual and counterfactual a user can see the differences that caused the model to make another decision. The factual and counterfactual should ideally lie close to each other in the feature space. In this case the counterfactual represents the minimum change required to alter the models prediction. Counterfactual explanations have many advantages. They can be generated without access to training data and without any information of the inner working of the models they explain. The only information needed are the inputs and outputs of the model. This makes them applicable to any model, even non-learning ones. Counterfactual explanations are also intuitively understandable and can be expressed in natural language. Because no previous knowledge is required they are ideal for end users that are unfamiliar with computer science and machine learning. Counterfactuals have additional advantages in terms of privacy. The explanation reveals no additional information, it consists only of an instance and its prediction. And because no direct access to the model is required no information about its inner workings are revealed, which may be a trade secret. However there are also some disadvantages to counterfactuals. For any instance that should be explained there usually exist multiple counterfactual explanations which requires additional work by developers to deal with the ambiguity. A counterfactual explanation is also local which means that it only explains the prediction of a single instance and can usually not be generalized to other instances.

To tackle the issue of locality we propose a new approach called the global explanation framework which tries to give counterfactuals a more global scope. It would be desirable to make counterfactual explanations more global, meaning that they explain multiple instances or ideally the whole feature space. To achieve this the global explanation framework maps the feature space and builds a representation of it in form of two graphs. In these graphs counterfactuals can be found more quickly and can be augmented with global explanations of the feature space. These global explanations are visualizations of the feature space and instances that are representative of the models decisions generated with submodular pick[3].

This paper is structured as follows: Section II explains the basics of XAI and counterfactual explanations followed by some related work in section V. Afterwards, section III covers the global explanation framework. Section IV presents the insights gained by comparing the new approach to existing ones. The last section VI summarizes everything as well as giving an outlook on possible future work.

II. Counterfactual Explanations

The word *counterfactual* consists of two parts: *counter* and *factual* so the literal meaning is something contrary to the facts. Humans engage in counterfactual thinking in their everyday life when they think about things that could have happened in the past that would have lead to a different present or future. For example: If the weather had been sunny instead of rainy, I would have gone for a walk instead of staying at home. Counterfactuals can be expressed in a more formal form as follows: If *X* had been *X'*, *Y* would have been *Y'*. The factual *X* has the consequence *Y*, however if *X* changes to the counterfactual *X'* the consequence changes to *Y'*. In the example above *X* is the rainy and *X'* the sunny weather. The rain (*X*) lead to the present state of staying at home (*Y*) whereas the sun (*X'*) would have resulted in going for a walk (*Y'*). Some psychologists and philosophers like Lipton[6] argue that, when people ask why a certain event happened they actually ask why this event happened and not some other one. In other words: If somebody asks "Why *Y*?" the implicit question they actually want to get answered is "Why *Y* and not *Y'*?". This kind of question asks for a counterfactual explanation.

If counterfactuals are used as an explanation for machine learning systems *X* and *X'* are instances in the input space of a model while *Y* and *Y'* are outputs or predictions made by the model. The instance that should be explained *X* was classified as belonging to the class *Y*. The counterfactual explanation *X'* is an instance that is classified as belonging to the desired class *Y'*. Thus the problem of finding a counterfactual explanation turns into a search problem in the feature space with the goal of finding an instance in close proximity that leads to the prediction of the desired class. The difference between the two instances *X* and *X'* explains which requirements were not met by *X* to achieve the desired prediction. A counterfactual explanation can be presented either by itself, or as the difference from its factual to highlight the changes responsible for the different classification. This can be done in either a tabular or graphical form to quickly see the differences or in natural language to make the explanations accessible to laypeople. Counterfactual explanations can also be used for image classification tasks[7] in this case the presentation of the results can also be pictorial.

A counterfactual explanation is only valid for a single instance, the factual, and can not be generalized to other instances. This makes counterfactuals local explanations in contrast to global explanations which explain multiple instances or the whole feature space. Because the explanation comes in form of another instance from the feature space counterfactuals are also example based explanations. Another property of counterfactuals is that they are model agnostic which means that there is no knowledge of the decision process required, the only information needed to generate the explanations are the inputs and corresponding outputs of a model. The fact that they can be found with only access to the model without any further information is a strong advantage of counterfactuals.

For any instance that should be explained there are often multiple possible counterfactual explanations. Every point in the feature space that is classified differently could theoretically be used as a counterfactual explanation. However to be a viable explanation a counterfactual should be relatively close to the instance it tries to explain. A closer counterfactual requires less changes to the original instance and is therefore briefer, easier to understand and an overall better explanation. Another important criterion is the actionability. If for example a credit application gets denied and the explanation states that the applicant has to become 5 years younger in order to receive a credit it would be a rather unpractical explanation. Counterfactual explanations should take into account which features are mutable, like a credit amount or duration, and which features are immutable, like an applicants sex, age or race. But even if the distance between factual and counterfactual and its actionability are taken into account it is still possible and often likely to find multiple equally suited counterfactuals. This poses a challenge for developers trying to utilize counterfactual explanations. Should they present all the counterfactuals that are found, which can lead to a better explanation but can also confuse the user, or should they only pick one counterfactual which would result in a more precise explanation but also requires some kind of rating. This is an open research question[8].

Counterfactuals have many advantages as explanations for a machine learning system. Because counterfactual thinking is intuitive to humans and counterfactuals can be expressed formally as well as in natural language they are especially useful as explanations for end users. Counterfactuals are also rather easy to implement because they are model-agnostic which means that they can be used on top of any prediction model. The models do not have to be learned, it can also be a rule based system[9]. This makes them for example useful for prototyping or any modular situation where the underlying model may be switched out. Because they are model-agnostic counterfactuals have the additional advantage that they do not reveal any information about the underlying model and data which is not accessible through the models predictions. So counterfactuals can for example be used by a company interested in protecting trade secrets or customer data[9]. Counterfactuals also satisfy the GDPRs[10] right to explanation. This combined with the previously mentioned advantages makes them very useful to companies that need to satisfy the GDPR.

However counterfactuals also have some disadvantages. The results are usually ambiguous, meaning that there are often multiple counterfactuals for any given data instance. This raises additional challenges that need to be addressed.

All the found counterfactuals can be presented to the user to be as transparent as possible and leave the choice to the user. As shown by Sokol et al.[11] this option should only be regarded with caution. Multiple counterfactuals reveal more information which may enable malicious users to gain unwanted knowledge about a system. The other option is to provide only chosen counterfactuals. This means that the found counterfactuals have to be evaluated or scored by some kind of metric and may result in worse explanations. On the other hand it might be impossible in some cases to find a counterfactual for a given instance that satisfies all desired constraints. In this case some constraints like the distance have to be reduced or another form of explanation has to be considered.

III. Global Explanation Framework

Counterfactual explanations are local explanatnios, which means, that they only explain a single instance in the feature space and can not be generalized to other instances. The approach described in this section tries to tackle this problem by giving counterfactual explanations a more global scope.

The global explanation framework creates a data structure over an interesting part of the feature space, which can encompass the whole feature space, with the goal of getting a representation of that region and speeding up the process of finding counterfactuals. The whole approach is depicted in figure 1. It starts with a feature space given in the form of a classifier. The process has three main steps: division, labeling and building a search graph. The first two steps: division and labeling are iterated in order to map the feature space and create a representation of it. Afterwards a search graph is created on this representation.

The algorithms for all three phases are implemented in interfaces and can therefore be easily changed. The implemented division algorithm splits every region in half along each dimension. For labeling an algorithm that samples a number of random points from a region and passes them to the classifier was implemented. If all the points belong to the same class the region can be labeled as belonging to this class, otherwise it is labeled as “uncertain” and divided further in the next iteration. The algorithm that creates the search graph connects each node to all the nodes it shares a border with. These three implementations are presented in more detail in the following.

A. Division and Labeling

The goal of the first two steps which are depicted in figure 1 is to hierarchically divide the feature space into areas belonging to the same class. Alongside the division a hierarchy tree is created to keep track of all the divisions. The hierarchy tree represents a parent child relation of all the regions in the feature space. The trees root node represents the chosen region in the feature space which can be the entire feature space. If a region gets divided, new children are added to the respective node. The sum of all children always adds up to the same region as the parent. This process is depicted in detail in figure 2.

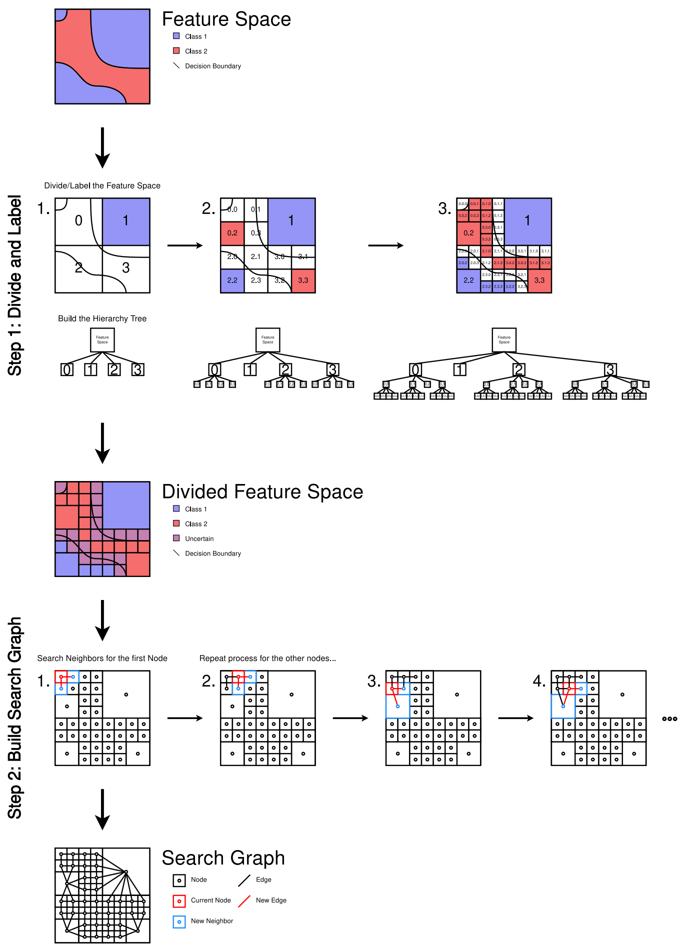

Fig. 1. The global explanation framework. First the feature space is divided and labeled afterwards a search graph is created.

The divide algorithm implemented here splits regions in half along each dimension. At the beginning the feature space gets divided by splitting it into two intervals of equal size along each dimension. The initial division is shown in detail in figure 3 step 1. After every dimension is split the feature space is implicitly split into orthotopes. An orthotope is the Cartesian product of intervals so the higher-dimensional analogue of a rectangle or cuboid. Because every dimension is split in half there are 2^D orthotopes created with every split where D is the dimensionality of the feature space. In a two-dimensional case the feature space is a two-dimensional orthotope, which is a rectangle. So the feature space is divided into smaller and smaller rectangles, as shown in figures 2 and 3

Now the orthotopes have to be labeled. The implemented algorithm labels an orthotopes by randomly sampling multiple data points in each orthotope and passing them to the black-box classifier for classification. Another way of labeling could be to deterministically get the samples from some kind of grid. Figure 3 shows a random sampling using six points in step 2. The sampled points are then classified using the black-box classifier as shown in step 3. With the labeled points the labels for the regions are inferred in step 4. If all the sampled data points inside an orthotope belong to the same class (Figure 3, region 1) it can be assumed that there are

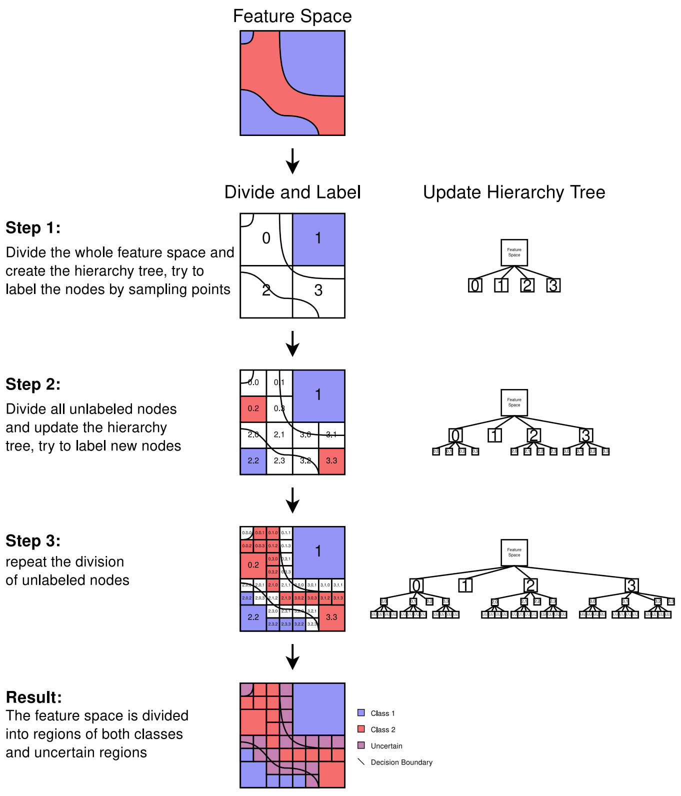
Feature Space
Divide and Label
Update Hierarchy Tree
Step 1:
Divide the whole feature space and create the hierarchy tree, try to label the nodes by sampling points
Step 2:
Divide all unlabeled nodes and update the hierarchy tree, try to label new nodes
Step 3:
repeat the division of unlabeled nodes
Result:
The feature space is divided into regions of both classes and uncertain regions
Class 1
Class 2
Uncertain
Decision Boundary

Fig. 2. The feature space is divided hierarchically and a tree is used to keep track of the created regions.

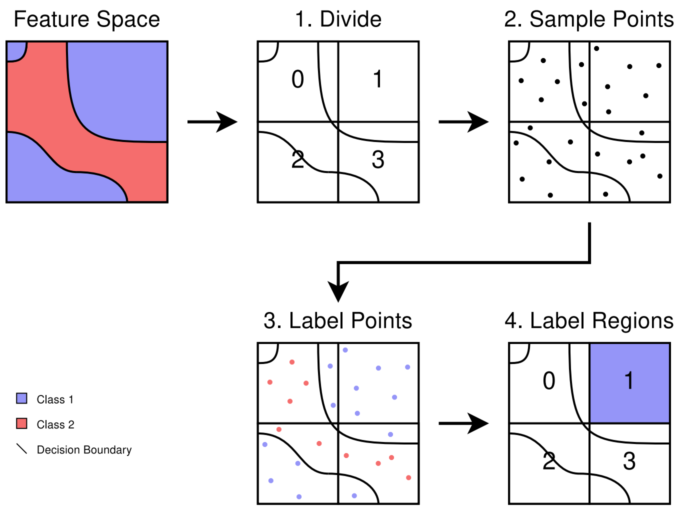

Fig. 3. The four steps of dividing and labeling a region.

no decision boundaries in the orthotope and therefore all the data points in the orthotope belong to the same class. In this case the orthotope is labeled as belonging to the respective class. If there are data points from different classes inside the orthotope (Figure 3, regions 0, 2, 3) then there must be at least one decision boundary passing through the orthotope. In this case the orthotope is labeled as “uncertain” and the process is repeated. With every subdivision a depth counter is increased. The subdivision and labeling steps are repeated until all orthotopes can be labeled or the depth counter reaches a predefined maximal depth.

During each division step all the created regions are added to a tree called the hierarchy tree. The hierarchy tree is defined by three things:

- The maximal depth of the tree
- Dimensionality of the feature space: Determines how much the tree branches
- Structure of the feature space: Determines the structure of the tree

Every node in the tree represents a region in the feature space. Whenever a node is split 2^D children are created, each representing one orthotope in the feature space. These 2^D new children are added to the respective node in the tree. The orthotopes of all the direct children of a node always combine to the nodes orthotope. The hierarchy tree starts with the root node which represents the whole feature space. Internal nodes of the tree are orthotopes that are divided further into smaller orthotopes, leafs are final orthotopes that aren’t divided anymore. Between these leafs the search graph is spanned in the next step. Each node is numbered by a number between 0 and 2^D-1. The numbers are assigned in such a way that their binary representation represent the corners of a D-dimensional hypercube. This binary representation is important for the creation of the search graph in section III-B.

This approach can be naturally adapted to only a region of the feature space. The only thing that has to change is the root node. If the orthotope of the root node only comprises a part of the feature space than only this part is mapped.

B. *Building the Search Graph*

In order to keep track of all orthotopes and search for counterfactuals two graphs are created. The first one is the already mentioned hierarchy tree which is created during the division and labeling steps. The second graph is the search graph which spans over all the leafs of the hierarchy tree. It connects orthotopes to enable the search for counterfactuals in neighboring orthotopes. The algorithm implemented here connects all orthotopes with a common border. Some steps of the creation of the search graph for the division created in the previous example can be seen in Figure 4

We are only interested in the neighbors of the leafs of the hierarchy tree because higher nodes only a summarization of the leafs. If, like before, the feature space has a dimensionality of D our orthotope has at least $2D$ neighbors, two in each direction, except for orthotopes on the border of the feature space. To be more efficient we only look for the neighbors in the positive direction of each dimension and add edges in both ways. During the division steps all regions are numbered with binary numbers These numbers of the nodes in the hierarchy tree are used to calculate neighboring nodes. The binary representation of all the direct children of a node can be seen as the corners of a D-dimensional hyper-cube. In the feature space each corner of the hyper-cube is analog to an orthotope. A property of such a hyper-cube also known as a hamming cube is that adjacent corners have a hamming distance of 1. Therefore adjacent orthotopes in the feature space can be

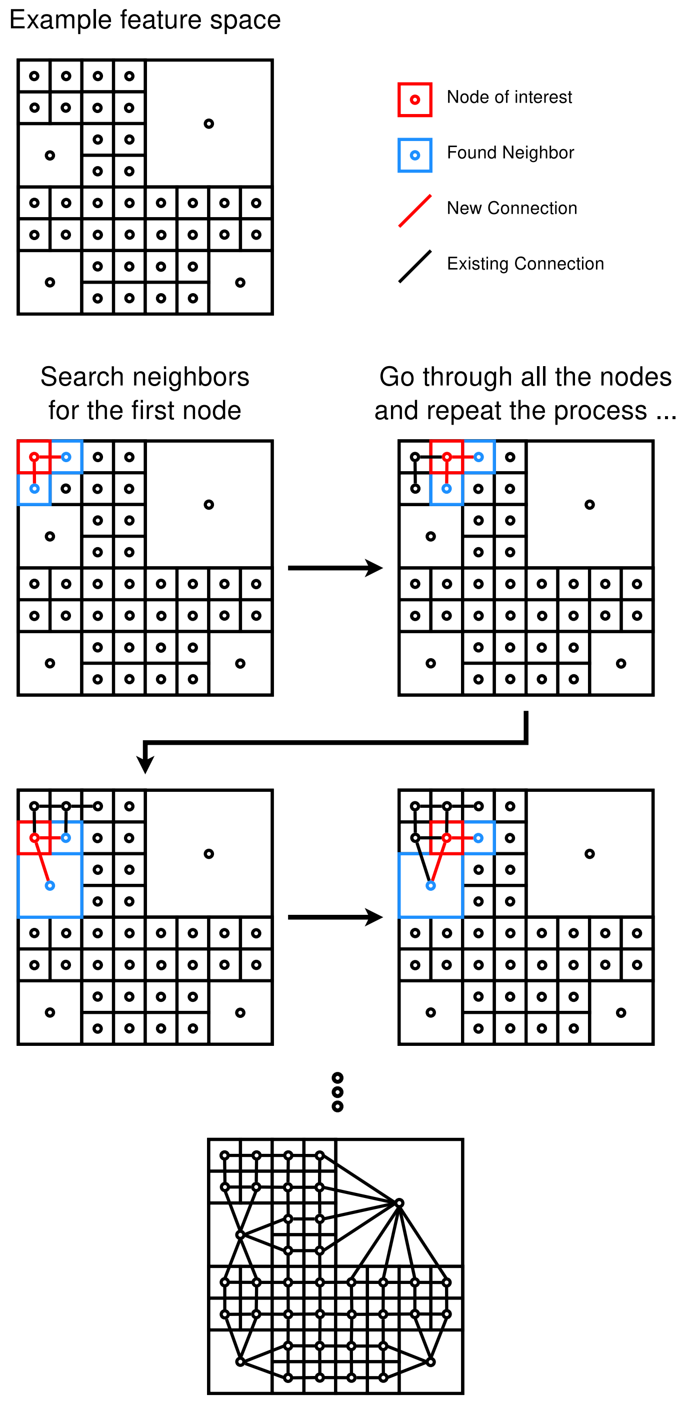
Example feature space
Node of interest
Found Neighbor
New Connection
Existing Connection
Search neighbors for the first node
Go through all the nodes and repeat the process ...

Fig. 4. The creation of the search graph

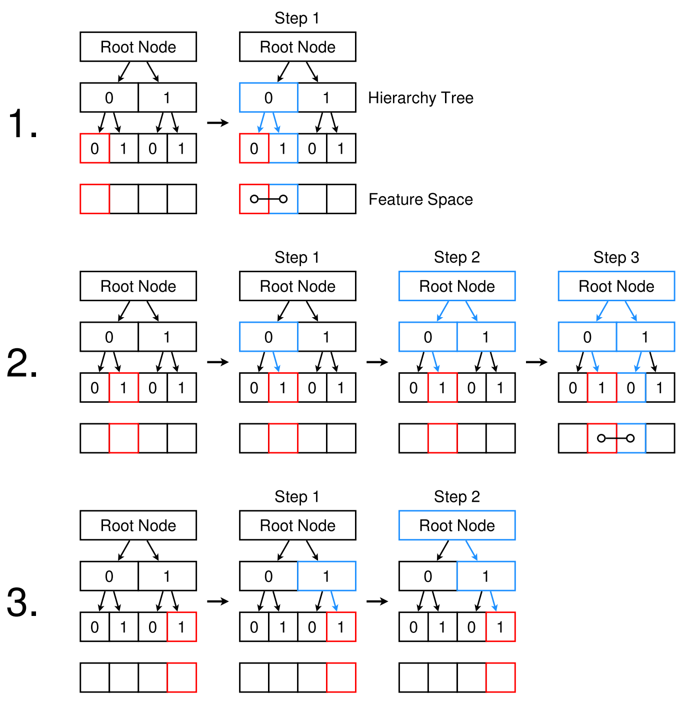

Fig. 5. The three cases in the search for neighbors

found by their hamming distance. This is important for the calculation of the edges in the search graph.

For each node the search for neighbors is conducted separately for each dimension of the feature space. In the following we name the currently searched dimension d. There are three possibilities for the border between the current node and its neighbors in each dimension: the border can lie inside a common parent, it can be outside of the current nodes parent or it can be the border of the feature space. An example for every case is depicted in figure 5. For each case the figure shows a one-dimensional feature space on the bottom and the corresponding hierarchy tree above. Red marks the currently regarded node and blue the path the algorithm takes through the tree. The algorithm for the search for neighbors is outlined in algorithm 1. This function $search\_neighbors()$ is executed for every node and every dimension. Case 1 is rather trivial to check, which is done in line 3. Due to the binary numbering the neighbor in this case is the node in the same parent with the binary digit of the currently regarded dimension flipped. In the other two cases higher parents have to be traversed which is done in function $neighbor\_in\_higher\_parent()$. In any case if a neighboring node was found it doesn't have to be a leaf. In this case the algorithm has to find the children of the found node that are on the border of interest. This is done by the function $nodes\_on\_border()$. The $generate\_candidates()$ function in line 41 generates these children which are of the form:

$$[0.1]^{D-dim-1}0[0,1]^{dim}$$

Where D is the dimensionality of the feature space and dim the currently regarded dimension.

As described earlier this search is only conducted in one direction of every dimension. This means that all edges have to be added bidirectionally. The process is repeated for every dimension and every leaf of the hierarchy tree to build the complete search graph.

C. Counterfactual Explanations

To get a counterfactual explanation for a data point the hierarchy tree is used to find the orthotope in which the data point lies and the search graph is used to find one or multiple closest orthotopes with a different label. If a orthorope has a different label it has to contain data points with different labels

Algorithm 1 Algorithm for searching neighboring nodes

```
function SEARCH_NEIGHBORS(node, dim)
    binary = binary_digit(node, dim)
    if binary == 0 then                                  ▷ case 1
        n = node_across_border(node, dim)
    else if binary == 1 then                             ▷ case 2 or 3
        n = NEIGHBOR_IN_HIGHER_PARENT(node, dim)
    end if
    neighbors = NODES_ON_BORDER(n, dim)    ▷ find all the nodes on the border
    return neighbors
end function

▷ find the neighbors if they are outside the current parent
function NEIGHBOR_IN_HIGHER_PARENT(node, dim)
    current = node.parent
    path = [node]
        ▷ go up the tree until current and its neighbors lie inside a common parent or the root is reached
    if current ≠ None then
        if current.is_root then return []                ▷ case 3
        else if binary_digit(current, dim) == 0 then
            current = node_across_border(current, dim)
            break                                        ▷ case 2
        end if
        path.append(current)
        current = current.parent
    end if
        ▷ reverse the steps on the other side of the border
    path.reverse()
    for node in path do
        if current.is_leaf then return [current]
        else
            current = node_across_border(node, dim)
        end if
    end for
    return current
end function

▷ returns the node if it is a leaf or all the children of the node that lie on the "left" border of the specified dim
function NODES_ON_BORDER(node, dim)
    if node.is_leaf then return [node]
    else
        nodes = []
        for n in generate_candidates(dim) do
            nodes.add(NODES_ON_BORDER(n, dim))
        end for
    end if
    return nodes
end function
```

which are by definition counterfactuals. To find the orthotope in which the data point lies the hierarchy tree is traversed. Starting at the root node the child containing the data point is calculated. To calculate the containing child the following formula can be used:

$$\sum_{i=0}^{d-1} 2^i \cdot \frac{sign(\frac{x_i - width_i}{2}) + 1}{2}$$

Where *x_i* is the coordinate of the data point and *width_i* the width of the current orthotope in the *i*-th dimension. This returns the binary number of the child in which the current data point lies. Once the correct child node if found it is entered and the same process is repeated until a leaf is reached. The orthotope represented by this leaf contains the data point. Once the correct orthotope is found the class of the data point is compared to the label of the orthotope. The classes can be different because the label of an orthotope is determined only by a sample of points inside. So the label of the orthotope can be a different class or "uncertain" because the maximal depth of the tree was reached during the division process. In both cases there is a counterfactual in this orthotope.

If the label of the data point and the orthotope are the same the search graph is used to find the closest orthotopes with different labels. A search algorithm similar to Dijkstra's algorithm is used for the search. Starting at the initial orthotope all its neighbors in the search graph are added to a list This list is then sorted by the distance of the data point to the respective orthotopes. The list is expanded by the neighbors of the closest orthotope until the first orthotope in the list has a different label. It is important to use a distance metric in the feature space to measure the distance between the data point and orthotopes as the distance in the search graph is not very representative of the real distance in the feature space. The framework is agnostic to the used distance metric. For all the experiments conducted in later sections the euclidean distance over normalized features was used. A better option may be the weighted euclidean distance to represent the different difficulty of changing different features this is however out of the scope of this work.

Now counterfactuals have to be found in the returned orthotopes. The simplest way is to use the points used earlier to label the orthotopes. Because the label of the orthotopes is different than the label of the factual there is at least one point from a different class in each orthotope that can be used as a counterfactual. Afterwards a binary search between the factual and counterfactual can be conducted to find a closer counterfactual. The binary search searches on the line between the factual and counterfactual to find the counterfactual with the smallest distance to the factual.

The search algorithm can also be used to find multiple diverse explanations. In this case the search is continued until a desired number of orthotopes is found. From each orthotope one or multiple counterfactuals can be chosen as before. Using multiple orthotope results in more diverse explanations. Alternatively any literature procedure, like the optimization approach by Wachter et al. [4] can be used and constrained to the found region.

D. Global Explanations

In addition to counterfactual explanations the global explanation framework can also provide global explanations for the machine learning model. Two forms of global explanations can be generated. The first form of global explanations are created with submodular pick[3] which chooses instances from the feature space that best explain the model. These instances are presented together with a counterfactual explanation as the first form of the global explanation. Additionally visualizations of the feature space can be used as global explanations. They are created by visualizing the regions in the feature space created during the division steps of the global explanation framework. These visualizations show a two-dimensional cut through the feature space where the decisions of the model in dependence of two variables can be seen. Regions from different classes are shown in different colors to let the user see where the model makes which decisions. An example can be seen in Figure 6.

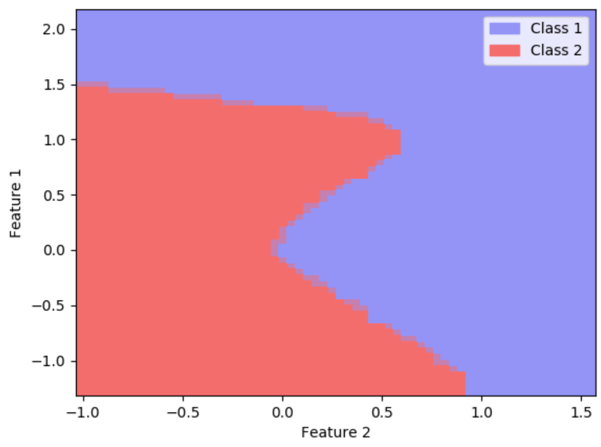

Fig. 6. A 2D Visualization of a feature space

E. Presenting Results and Explanations in Natural Language

Once explanations are generated they have to be displayed to the user. Counterfactuals can be displayed in tabular form or in natural language. Additionally different two-dimensional representations of the feature space with the labeled regions from the global approach can be shown. In the two-dimensional representations the different orthotopes can be shown in different colors to see what decisions the model makes in which regions.

One big advantage of counterfactual explanations is that they can be expressed in natural language which makes them easily understandable even for laypersons. The global framework can create such sentences in natural language for the found counterfactuals. Explanations presented in this way are easy to generate and can be understood by everyone. The sentences can be created using simple string substitution. This makes the creation of the sentences very easy and extensible. Here are some examples generated with predefined templates:

- The prediction for the data point $income : 5000, age : 25, credit : 11000$ is $rejected$. If the data point had been $income : 6000, age : 25, credit : 10000$ the prediction would have been $accepted$.
- In order to change the prediction from $rejected$ to $accepted$ you have to apply these changes: $income + 1000, credit - 1000$.
- To achieve the desired prediction of $accepted$ the features $income + 1000, credit - 1000$ have to change.
- Your request has been classified as $rejected$. To change the prediction to $accepted$ these values have to change: $income + 1000, credit - 1000$.

These generated sentences can be used for local counterfactual explanations and the first form of global explanations described in section III-D. In addition to this visualizations of the mapped feature space can be shown to the user. The visualizations are easy to understand and show the user two complete dimensions of the feature space. For feature spaces with more than two dimensions only a cut through the feature space can be shown. This means that some values have to be fixed. These values are then listed in the title of the visualization. These visualizations can serve different purposes. They can be used in a more local form by visualizing the surroundings of a specific instance to better understand the models behavior around this instance. To serve as a global explanation visualizations at different points can be used to give the user a representation of the feature space as a whole. The visualizations are generated by showing the regions generated in the divide and label steps of the global explanation framework in a color representing their label. An example can be seen in figure 6. It shows the feature space of a two-dimensional two-class classification problem. Each leaf of the hierarchy tree is shown in the plot and colored according to its label. The unlabeled regions, in which instances from both classes were found, are colored purple to make a visually smooth transition.

TABLE I
DATASETS

Dataset	Reference	Dimensionality
Adult	[12]	14
Moons	[13]	3
Titanic	[14]	7
Transfusion	[15]	5
Wine	[16]	13

IV. Evaluation

In this chapter the work presented in the previous sections is evaluated. The datasets used in the experiments are listed in table I. A model was trained for each dataset and explained using different methods.

The method of finding counterfactuals with the global explanation framework is compared to other methods for finding counterfactuals. In section IV-A the different methods are compared in terms of the distances from the found counterfactuals to their factuals. In section IV-B the methods are compared in terms of the calculation time needed in order to find a counterfactual.

TABLE II
GLOBAL EXPLANATION FRAMEWORK (GEF) VS. WACHTER: DISTANCE (SAMPLING POINTS: 10)

Dataset	GEF		Wachter
	Depth	Distance	
Adult	2	2.19	0.59
	3	-	
	4	-	
Moons	2	0.81	0.54
	3	0.71	
	4	0.62	
Titanic	2	1.69	0.75
	3	1.54	
	4	1.22	
Transfusion	2	1.36	1.23
	3	1.28	
	4	1.20	
Wine	2	2.38	1.81
	3	-	
	4	-	

A. Distance

Here counterfactuals from the global explanation framework are compared to counterfactuals found using other methods. The methods are compared in terms of the distance between the factuals the their respective counterfactuals. The euclidean distance over normalized features is used as the norm for determining the distance. For each dataset multiple hierarchy trees with different maximal depths are created and compared. In all the tests 10 randomly sampled points are used to determine a regions label. Some rows in the table are empty because the calculation time needed to set up the graphs exceeded one hour. The distances are compared to the optimization approach by Wachter et al.[4]. The results can be seen in table II.

The distances decrease with an increasing depth of the hierarchy tree. This was expected because a deeper hierarchy tree contains more regions which results in a finer division. This means that the feature space can be mapped in more detail, which in turn results in a more accurate representation in which better counterfactuals can be found.

It can also be seen that the counterfactuals found by the global explanation framework mostly have greater distances from their factuals than the ones found by the other approach. The distance between the factual and its counterfactual explanation is a measure of how much an instance has to be changed to achieve the desired prediction. Explanations with a greater distance are still valid explanations they may however be less relevant because they require more change to reach the desired result.

B. Performance

The next metric under which the counterfactual search of the global explanation frameworks is evaluated is performance. The performance of different models is measured by the calculation time needed to find counterfactuals. For the graph search multiple measurements are again conducted with different maximal depths of the hierarchy tree. Some rows in the table are again empty because the calculation time needed to set up the graphs exceeded one hour.

The results show that the setup time increases very rapidly with the depth of the hierarchy tree and even more rapidly with the dimensionality of the feature space. The reason is that the depth of the hierarchy tree as well as the dimensionality, increase the number of created nodes exponentially. This makes the global explanation framework less useful for problems with more than about 15 features.

The calculation time for counterfactuals on the other hand is much lower than the time needed by the other approach. The setup has to be done only once for a specific feature space. Afterwards the graphs can be stored and any number of counterfactuals can be calculated. This means that the global explanation framework could be used in cases where the calculation time during an execution is a bigger concern than the initial setup time.

C. Results

The tests done in this section show that the global explanation framework has problems with higher-dimensional feature spaces. The time needed to build the graphs goes up exponentially with the number of dimensions because the number of nodes in the graphs grows exponentially. Section VI-A shows some possible solutions to this problem. However the setup has to be executed only once to create the graphs. Afterwards the created search graph can be used to find as many counterfactuals as needed. The time needed to search counterfactuals in the graph is much smaller than the time needed by the approach from Wachter et al. The presented framework makes a trade-off between a longer setup time and a much smaller calculation time per counterfactual.

The distances between factuals and counterfactuals decreases with increasing depth of the hierarchy tree. This is because a deeper hierarchy tree means more divisions which in turn means a more precise mapping of the feature space.

V. RELATED WORK

Burkart et al.[17] propose a framework to generate decision boundary centered explanations which come in form of surrogate models and counterfactuals. The goal is to find the local decision boundary for a given point and create a representation of the boundary with a simpler model. The simpler model enables the user to understand the decision made by the model in this area. The framework consists of five phases. Suggested implementations are presented for every phase however the framework itself is agnostic to changes in every phase. In the first phase an initial counterfactual has to be found. This can be done with any method found in the literature, the method described in the paper is an adaption of the optimization approach by Wachter et al.[4]. The goal of the second phase is to find support points which are other counterfactuals that are close to the same decision boundary. Afterwards an implicit representation of the decision boundary is created. The points found in the previous phase are all counterfactuals so the decision boundary passes between each of them and the factual. For each point a binary search is conducted on the line between it and the factual to find the respective border touchpoint which, as the name suggests, is a point that lies on the decision boundary. In phase 4 the surrogate model is trained. To do so more points around

TABLE III
GLOBAL EXPLANATION FRAMEWORK VS. WACHTER: CALCULATION TIME (SAMPLING POINTS: 10)

Dataset	Dimensions	Global Explanation Framework			Wachter
		Depth	Setup Time	Calculation Time	
Adult	14	2	1001	0.0038	6.91
		3	-	-	
		4	-	-	
Moons	2	2	0.042	0.0007	0.52
		3	0.064	0.0009	
		4	0.102	0.0008	
Titanic	6	2	9.97	0.0014	3.78
		3	209	0.0040	
		4	11934	0.0211	
Transfusion	4	2	0.305	0.0008	2.90
		3	1.21	0.0026	
		4	11.26	0.0056	
Wine	13	2	386	0.0904	6.96
		3	-	-	
		4	-	-	

the border touchpoints are sampled. With these points the surrogate model can be trained. The surrogate model is chosen from a class of models by the loss to the original model. It is deliberately kept simple by constraining its complexity. In the last phase the results are presented. There is a special focus on three main aspects to make the approach as accessible as possible: understandability, actionability and simulatability. For understandability the reasons for the classification of the instance should be made clear. Feature importance, using for example LIME[3], is a good way to convey understandability. For actionability the user has to receive advice that can be implemented. This advice is given by the counterfactuals as they show changes to the instances that can be applied to change the classification. The last aspect of the explanation is simulatability which means that a user gets information that allows him to understand or simulate the decision process. This information is given in the form of the surrogate model. The constraints put on the model keep it simple and therefore make it understandable. Decision trees are especially useful for this task. The framework presented by Burkart et al. is similar to this work in so far as it also builds on counterfactuals and augments them with other forms of explanations. The global explanation framework proposed in this paper accelerates the search for counterfactuals and can generate visualization of the whole feature space.

Goyal et al.[7] developed a system to generate counterfactual visual explanations which apply the concept of counterfactuals to images. In an image *I* that is classified as class *c* they try to find a minimal number of regions that need to change in order to make the model predict a desired class *c'*. The regions to change are swapped from a distractor image *I'* that is already classified as *c'*. Spacial feature maps from a convolutional neural network are used as the exchange regions to make this approach feasible. The authors emphasize the usefulness of visual explanations for machine teaching. They claim that visual explanations can aid humans in learning image classification. To demonstrate this they conducted a user study in which humans learn to distinguish different bird species and are shown important regions for the classification if they make a mistake. They find that visual explanations help the subjects in learning and improve the test accuracy. The global explanation framework also generates visual explanations which are however separate from the generated counterfactual explanations.

Grath et al.[18] contributed two new concepts to the state of the art, namely positive counterfactuals and weighted counterfactuals. Positive counterfactuals are counterfactual explanations that should explain a desired outcome like an accepted credit application in contrast to explaining only negative predictions. Because they are counterfactuals they can be generated the same way. The authors claim that users can gain valuable information from these explanations. They are shown to the users as tolerances that tell them, assuming that all the other values stay the same, how much each value can change in order to keep the desired prediction. This could for example let the user make informed financial decisions in the future. Positive counterfactuals can also be generated with the global explanation framework. Weighted counterfactuals assign a weight to each feature that should represent the features ability to change. The idea has similarities to the approach by Fernandez et al.[19] but utilizes the approach from Wachter et al.[4] to generate the counterfactuals. They adapt the distance function as follows to incorporate the weight:

$$d_2(x, x') = \sum_{j=1}^{p} \frac{|x_j - x'_j|}{\mathrm{MAD}_j} \theta_j \tag{1}$$

The value of θ_j determines how much it costs to change the j-th feature. A bigger θ_j has the effect that other instances in the j-th direction get assigned a greater distance. This penalizes the change of the feature j and therefore makes it more likely to retain the current value. If θ_j is smaller the effect is reversed. The framework presented in this paper can generate counterfactuals with different distance metrics so it would be possible to generate weighted counterfactuals with the metric from equation 1.

An additional challenge of counterfactuals to keep in mind is that they may explain unwanted artifacts of the classifier that are not part of the training data. Laugel et al.[20] define what it means for a counterfactual to be justified and provide an algorithm that checks the justification of counterfactuals. They define that a counterfactual is justified if there exists a continuous path between the counterfactual and an instance from the training data that is classified in the same class. Every point on the path that connects the two instances also has to be classified as the same class. This should ensure that the counterfactual is not part of an artifact of the classifier that is represented in the training data. The global explanation framework searches counterfactuals in an approximation of the feature space. This could help to reduce unjustified counterfactuals because small artifacts will not be represented in the approximation.

VI. Conclusion

This paper presented a new approaches that tries to overcome a big problem of counterfactual explanations, their locality. Three types of explanations can be generated by the presented framework: First instances picked by submodular pick[3] and their counterfactual explanations are used as global explanations to give an first overview of the models decisions. Secondly visualizations of two-dimensional cuts through the feature space can be created which serve as an explanation of the models decisions in dependence of two features. Lastly the framework allows a quick search for counterfactual explanations for individual instances. These three parts give the user an understanding of the feature space as a whole as well as giving the ability to examine specific instances or regions more deeply.

The counterfactual explanations generated by the global explanation framework were compared to counterfactuals generated using an optimization approach by Wachter et al.[4]. The global explanation framework needs a longer setup time to create the graphs used to generate the explanations but once this is done the search for counterfactuals is much faster than existing approaches. The approach by Wachter et al. however generates in most cases counterfactuals with a smaller distance to their factual.

A. Future Work

The global explanation framework was developed as a framework with the ability to extend it in mind. The processing of the feature space consists of three major steps which are all implemented as interfaces. The feature space is hierarchically divided into regions of instances which ideally all belong to the same class. The regions are labeled during the division process and at the end a search graph is build over these regions that is used to generate explanations. Because the three steps are implemented as interfaces the algorithms presented here can be exchanged with other ones. This leaves room for improvements of the framework.

The evaluation discovered that a high number of dimensions poses a challenge for the global explanation framework. The reason is that the proposed division algorithm splits each region once along each dimension. This results in an exponential growth in operations to create the graphs. One solution for this problem could be to be more selective about the dimensions along which splits are performed. Dimensions with protected attributes could be omitted by the division algorithm which would have the added benefit of the respective features being unchanged in the generated counterfactuals. Alternatively feature importance could be calculated and less important features could be divided less or omitted completely. A reduction of the dimensions would most likely result in better, more relevant explanations and would reduce the computation time because less operations are performed. Another way could be to parallelize the division and labeling steps. This should be easily achievable because once split the different regions are completely independent from each other.

It could also be possible to improve the counterfactuals found by the framework. The division algorithm presented here splits the regions exactly in half along each dimension. In addition to omitting unchangeable or less important dimensions the splits could be performed at more strategic positions. Divisions closer to a decision boundary could for example result in a more precise mapping of the feature space. Another improvement could be weights on the edges of the search graph or a different distance metric in the search algorithm. For example the metric by Grath et al.[18] in equation 1 or the weighted euclidean distance. This could incentivise the search algorithm to search preferably along certain dimensions. Harder to change features could get a higher cost making them less attractive in the search for counterfactuals. This could be used to reflect user preferences or keep some features unchanged by giving them an infinite cost. These changes would make the generated explanations more relevant to the user or could speed up the search.

Acknowledgment

This work was supported by the Competence Center Karlsruhe for AI Systems Engineering (CC-KING, https://www.ai-engineering.eu) sponsored by the Ministry of Economic Affairs, Labour and Tourism Baden-Württemberg.

References

[1] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. 2017.

[2] Alex A Freitas. Comprehensible classification models: a position paper. *ACM SIGKDD explorations newsletter*, 15(1):1–10, 2014.

[3] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " 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*, pages 1135–1144, 2016.

[4] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

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

[6] D. Knowles, Royal Institute of Philosophy, Royal Institute of Philosophy Conference on Explanation, University of Glasgow) its Limits (1989, and Royal Institute of Philosophy. Conference. *Explanation and Its Limits*. Landmarks of World Literature. Cambridge University Press, 1990.

[7] Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. Counterfactual visual explanations. In *International Conference on Machine Learning*, pages 2376–2384. PMLR, 2019.

[8] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

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

[10] Council of European Union. Regulation on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (data protection directive), 2016.
https://gdpr-info.eu/.

[11] Kacper Sokol and Peter A Flach. Counterfactual explanations of machine learning predictions: opportunities and challenges for ai safety. In *SafeAI@ AAAI*, 2019.

[12] Dheeru Dua and Casey Graff. UCI machine learning repository: Adult dataset, 1996. https://archive.ics.uci.edu/ml/datasets/Adult.

[13] scikit learn. Moons dataset. https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html.

[14] Ben Hamner Anthony Goldbloom. Kaggle: Titanic dataset. https://www.kaggle.com/c/titanic.

[15] Dheeru Dua and Casey Graff. UCI machine learning repository: Blood transfusion service center dataset, 2008. https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center.

[16] Dheeru Dua and Casey Graff. UCI machine learning repository: Wine dataset, 1991. https://archive.ics.uci.edu/ml/datasets/wine.

[17] Nadia Burkart, Maximilian Franz, and Marco F Huber. Explanation framework for intrusion detection. In *Machine Learning for Cyber Physical Systems*, pages 83–91. Springer Vieweg, Berlin, Heidelberg, 2021.

[18] Rory Mc Grath, Luca Costabello, Chan Le Van, Paul Sweeney, Farbod Kamiab, Zhao Shen, and Freddy Lecue. Interpretable credit application predictions with counterfactual explanations. In *NIPS 2018-Workshop on Challenges and Opportunities for AI in Financial Services: the Impact of Fairness, Explainability, Accuracy, and Privacy*, 2018.

[19] Carlos Fernandez, Foster Provost, and Xintian Han. Counterfactual explanations for data-driven decisions. 2019.

[20] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. In *Twenty-Eighth International Joint Conference on Artificial Intelligence {IJCAI-19}*, pages 2801–2807. International Joint Conferences on Artificial Intelligence Organization, 2019.