Using Polygonal Data Clusters to Investigate LIME

[1]Jesse He, [2]Subhasish Mazumdar
[1]*The Ohio State University Columbus, Ohio, USA*
[2]*Department of Computer Science and Engineering*
New Mexico Institute of Mining and Technology Socorro
New Mexico, USA

Abstract

While machine learning classifier models become more widely adopted, opaque "black-box" models remain mostly inscrutable for a variety of reasons. Since their applications increasingly involve decisions impacting the lives of humans, there is increasing demand that their predictions be understandable to humans. Of particular interest in eXplainable AI (XAI) is the interpretability of explanations, i.e., that a model's prediction should be understandable in terms of the input features. One pop- ular approach is LIME, which offers a model-agnostic frameworkfor explaining any classifier. However, questions remain about the limitations and vulnerabilities of such post-hoc explainers. We have built a tool for generating synthetic tabular data sets which enables us to probe the explanation system opportunistically based on its architecture. In this paper, we report on our success in revealing a scenario where LIME's explanation violates local faithfulness.

1. Introduction

The increasing success of machine learning in analyzing data and the rising potential for its applications in diverseareas has made the necessity for understandable explanations of machine learning predictions readily apparent. However, while these models have displayed impressive classification accuracy, the reasons for their predictions remain inscrutable in many cases. As these models are increasingly being deployed to inform decisions impacting the careers, lives, and freedom of people, there is an urgent need for explanation frameworks. Each of the models used need to be interpretable; their results need to be explained in terms of their inputs. Researchers have started to address this need [7, 9, 10] and offer explanation frameworks, but this in turn raises concerns about the limitations of such frameworks. Towards that end, we have built and deployed a tool for generating synthetic tabular data sets.

Our insight is that a key tool in evaluating such explanations, and in machine learning more generally, is the ability to generate synthetic data sets that we can intuitively grasp and that enables visualization. The family of classificationproblems we have chosen contain n-dimensional data for which the boundaries of identifying classes can be described geometrically using polyhedrons, with an emphasis on the 2-dimensional case. Below, we will use the term "polygonal cluster" to refer to such a class of data points which is definable with a polygonal boundary, and the term "polygonalclustering" to refer to the task of classifying such data. Wewill also allow a polygonal cluster to have a certain fractionof outliers which lie outside the polygonal boundary. This presents an interesting class of classification problems that currently lacks a robust synthetic data tool.

There are a number of efforts in the realm of explainable AI, particularly regarding "opaque" or "black-box" models [2]. One popular approach is the use of Local Interpretable Model-agnostic Explanation (LIME) [9], which attempts to identify the contribution of individual features towards a classifier's prediction by observing the behavior of the classifier around perturbations of the original data point. An intuitive illustration from Ribeiro, Singh, and Guestrin [9] is given in Figure 1.

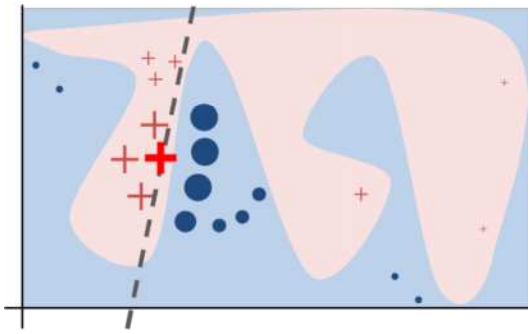

Figure 1. Illustration to build intuition for LIME
Ribeiro, Singh, and Guestrin [9]

The black-box model's prediction for the red cross is based ona complex decision boundary, illustrated by the blue/pink background. LIME samples nearby instances, gets the model's predictions, and attempts to create an explanation that is locally faithful.

LIME has seen adoption in image and text analysis, but its application to tabular data has certain limitations. Different methods of perturbing or identifying a local neighborhood for a data point can create different explanations for the same classifier on the same dataset [3]. Because of the prevalence of tabular data in machine learning applications, it is important to investigate LIME's ability to explain tabular classification models.

We are interested in data which can be described with a polygonal model, which attempts to create a suitable polygon to represent the shape of a data cluster, often employed with spatial data [1]. Of particular interest are clusters bound by polygons which are *non-convex*, since convex hulls can create large empty areas that do not tightly fit the structure of a cluster. Although sample datasets exist that can be used for such problems, there is no robust way to randomly generate and customize new datasets with polygonal clusters.

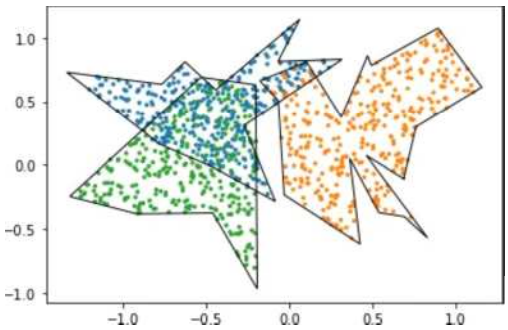

Figure 2. Three uniform balanced polygonal clusters.

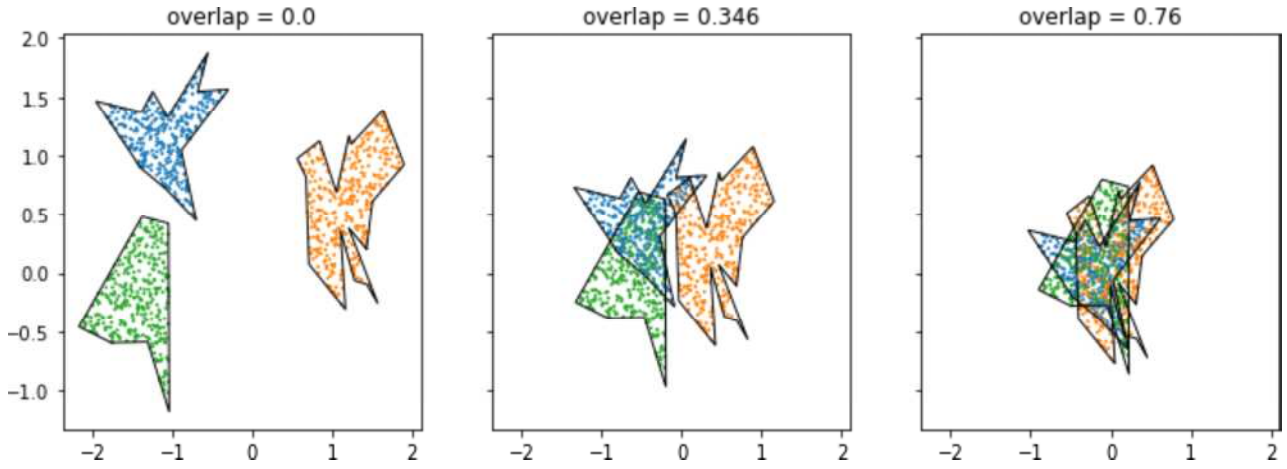

Figure 3. Dataset from Figure 2 with clusters shifted to create different overlaps. From left to right: dataset with no overlap, original dataset, dataset with high overlap.

In this paper, we describe a software tool which uses a simple but effective method of generating such classification problems, and we demonstrate its utility in investigating the behavior of LIME on a black box classifier by producing examples where LIME's explanations fail to be locally faithful to the classifier's behavior.

In the next section of this paper, we review related work. Next, we outline our scheme for synthetic data generation. In the following section, we show how a problematic behavior of LIME is revealed by our approach. Finally, we offer concluding remarks.

2. Related Work

Given any classifier and a set of inputs along with their classifications, LIME [9] takes a sample and creates a linear approximation of the classifier's decision boundary in its neighborhood by repeatedly perturbing the sample and checking the classifier's decision on the perturbed input. Such perturbation-based methods are popular for post-hoc explanation [2], which provides data scientists with insight, and consequently, a powerful tool that enables comparison among machine learning models. Accuracy notwithstanding, a model that relies on features that are relevant to humans is viewed as trustworthy, while one that relies on accidentally correlated features is considered dangerous, even when both display statistically equivalent accuracy. In addition, such post-hoc explanations can potentially uncover bias that could be implicit in classifiers. Lundberg and Lee's SHAP [7], which stands for *SHapley Additive ExPlanations*, computes an importance metric for each feature for a given prediction. Their technique is to consider all relevant subsets of features that contain a given feature to assess its contribution.

However, Slack et al. [11] demonstrated that such explainers can themselves be fooled. They demonstrated a scenario in which an adversarial entity could demonstrate an explanation of their choosing, for example, an explanation that lets a biased classifier go undetected. With their *scaffolding* technique, they could scaffold a biased classifier such that the latter would continue to churn out biased classifications without such bias being detected in the post-hoc explanations. In addition, they were able to show that both LIME and SHAP were vulnerable to their scheme.

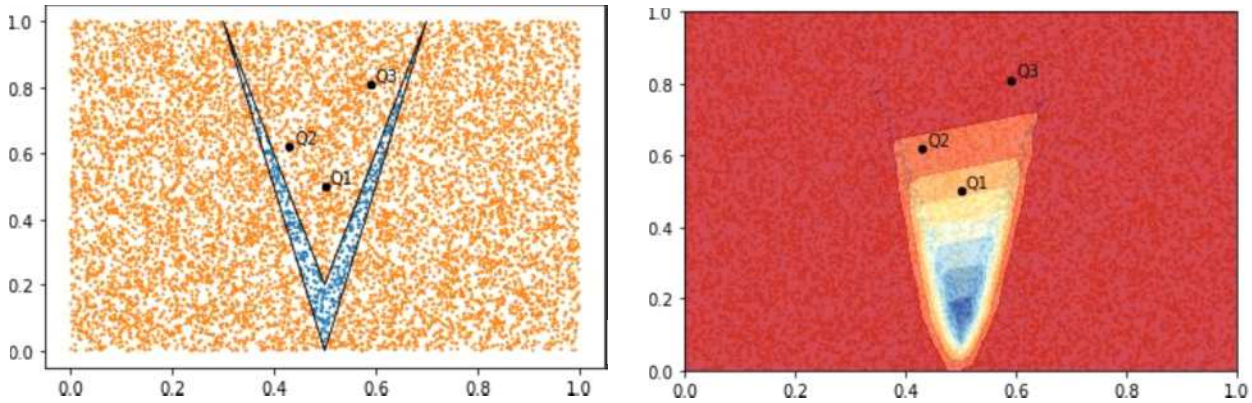

Figure 4. A classification problem featuring a "V"-like polygon with sample points Q_1 = (0.5, 0.5), Q_2 = (.43, .62), Q_3 = (.59, .81) (left), the classifier's decision function (right).

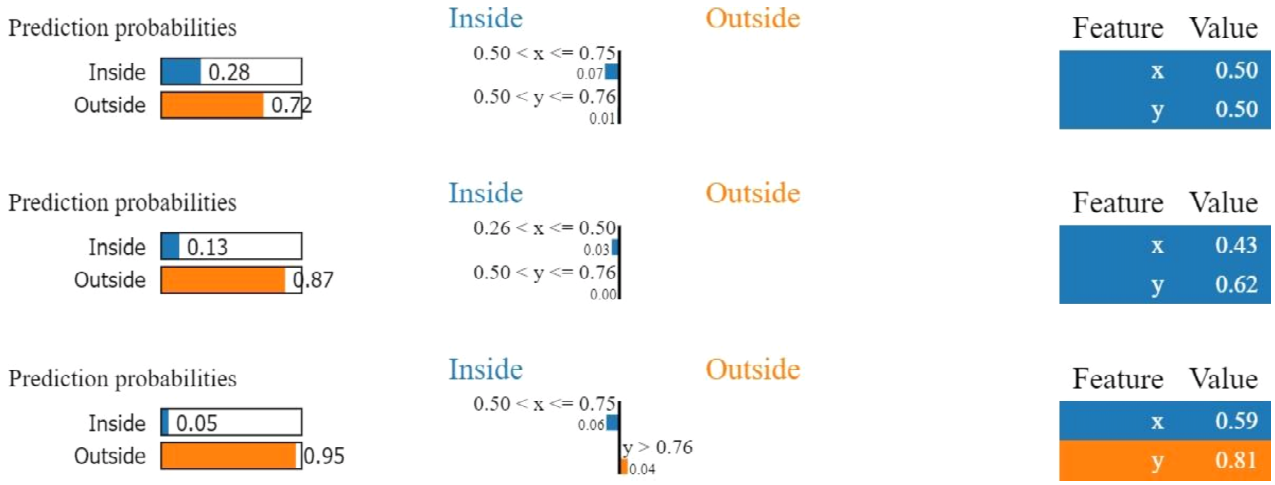

Figure 5. LIME's explanations at Q_1, Q_2, and Q_3.

Other synthetic data generation software includes the datasets module of the scikit-learn Python library [8], which provides a number of methods to generate synthetic machine learning problems. The use of polygonal models for data mining is investigated by [1], demonstrating the utility of a synthetic data generation tool which can create tabular data sets for which polygonal models are well-suited.

3. Generating Clusters with Polygonal Boundaries

We can specify an n-gon p as an ordered list $p = (v_1, v_2, \ldots, v_n)$ of its vertices, which we generate using a star-like approach: given a specified or randomly generated "cen- ter" we draw a radius r uniformly at random from some range $[r_m, r_M)$, where $0 \leq r_m < r_M$, and an angle θ uniformlyat random from $[0, 2\pi)$, giving us the polar coordinates for each vertex. To ensure that the polygon contains the central point about which it is generated, we check that no consecutive angles θ_1, θ_2 have a difference of more than π radians counter-clockwise. We then connect these vertices counter-clockwise to produce the polygon. Once we have created a polygon p,we use rejection sampling to generate points that lie inside p. Of course, p need not be generated randomly by the above process; the user may specify a polygon by enumerating its vertices.

Our software uses Matplotlib's path library [6] to represent polygonal paths and test if a polygon contains a given point. This also allows for easy visualization in plots made using Matplotlib and for integration with other Python libraries, including efficient vectorized operations with NumPy [5]. An example is shown in Figure 2.

A. Controlling Overlap

One additional task in generating new classification prob- lems is to customize the difficulty of the problem. The primary way we achieve this is by moving polygonal clusters closer or farther away from each other, and in particular manipulating

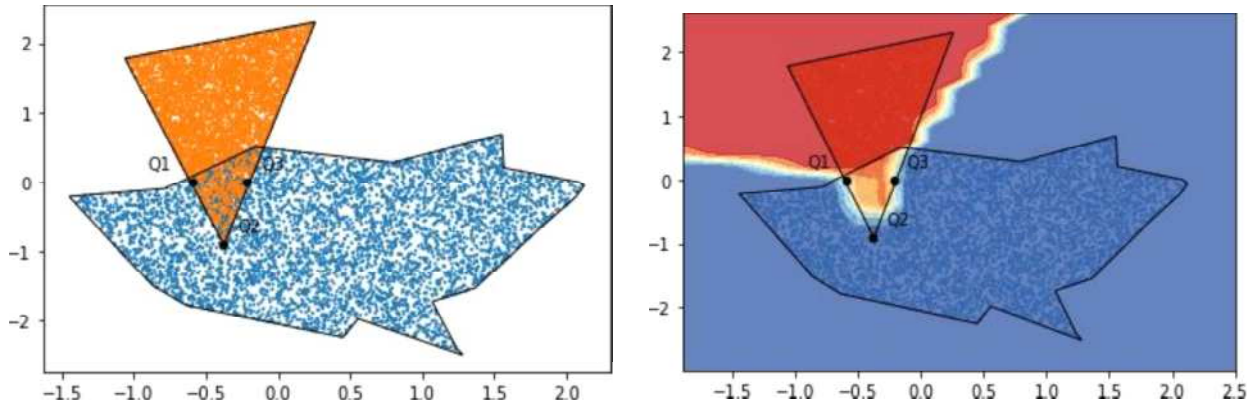

Figure 6. A binary polygonal classification problem. A plot of the dataset and polygons (left) next to the classifier's decision boundary (right) with marked points $Q_1 = (-0.6, 0)$, $Q_2 = (-0.38, -0.9)$, and $Q_3 = (-0.22, 0)$.

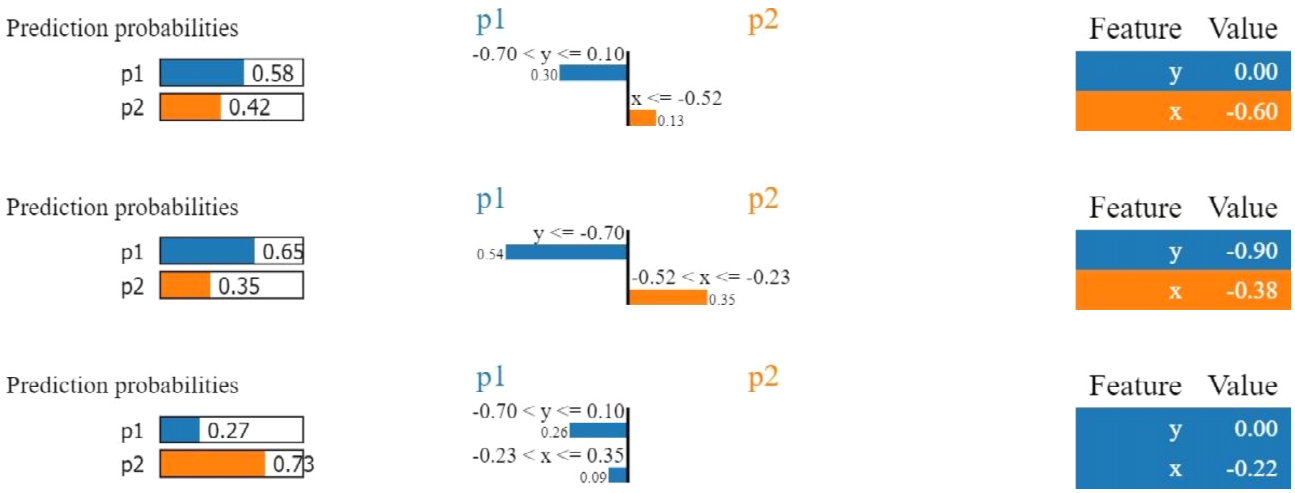

Figure 7. LIME's explanations for the classifier's predictions at these points.

the *overlap* between clusters. Given a set $X \subseteq \mathbf{R}^2$ of points and regions P_1, P_2 bounded by polygons p_1 and p_2, respectively, we define the overlap between these clusters by

$$\text{overlap}(X, p_1, p_2) = \frac{|X \cap P_1 \cap P_2|}{|X|}.$$

For problems with more than two polygons, we extend this definition by counting the number of points of X that lie in the intersection of any two polygons:

$$\text{overlap}(X, p_1, \ldots, p_k) = \frac{\left|X \cap \left(\bigcup_{1 \le i < j \le k} (P_i \cap P_j)\right)\right|}{|X|}.$$

Then shifting clusters gives us control over the overlap of a classification problem, as seen in Figure 3. The method by which we control the overlap is given in Appendix A. By manipulating the overlap of our polygonal point sets, we can effectively scale the difficulty of the classification problem: When there is no overlap, a simple linear model may suffice to accurately classify new test points. Introducing overlap between clusters can give us insight into how classifiers create their decision boundaries in the presence of ambiguous data.

4. Explaining Classifier Behavior on Polygonal Clusters

We first demonstrate the use of polygonal boundaries to specify challenging classification problems by creating a polygon $p = ((.5, 0), (.7, 1), (.5, .2), (.3, 1), (.5, 0))$ and training an off-the-shelf multilayer perceptron (MLP) classifier to classify points as inside or outside it. We then select predictions in challenging areas and ask LIME to explain these predictions.

As we can see in Figure 4, the inner triangle which is in the convex hull of p but not within its boundaries is challenging for LIME to explain. Because LIME looks at individual features, its attempted explanations for the specified points contradicts the classifier's prediction: the classifier correctly identifies that each point lies outside p, but LIME's explanations suggest that

each point *should* be classified as inside the polygon based on their x values, as seen in Figure 5.

To investigate the efficacy of LIME in evaluating tabular data in 2 dimensions with the overlap we have defined, we generate a binary classification problem with two polygonal clusters $p1$ and $p2$ whose overlap is between 0.1 and 0.2, and we train an MLP on a subset of the data. We then probe the classifier's prediction of a particular point using LIME and compare it to the classifier's actual decision function as shown in Figure 6, with LIME's explanations in Figure 7.

All three points were chosen to be difficult for the classifier. Each lies within the boundaries of both $p1$ and $p2$ but is very close to the boundary of $p2$, and this difficulty is reflected when examining the classifier's decision function. LIME's explanations at Q_1 and Q_2 reflect the classifier's uncertainty at those points, but its behavior at Q_3 is unintuitive: the MLP classifies Q_3 in $p2$ with a probability of 0.73, but LIME's explanation for this prediction asserts that the x and y values are both in ranges correlated with $p1$. This explanation contradicts the actual prediction of the classifier at Q_3, violating LIME's local faithfulness.

5. Conclusion and Future Work

The method of generating and manipulating polygonal tabular data we have presented is effective at generating simple example datasets for clustering problems. The generated datasets are easily visualized and can be customized to fit different geometric and statistical structures. Using polygons to specify the geometric structure allows for customization of cluster shape, and using overlap allows for customization of the difficulty of a generated classification problem.

In particular, our simple definition and manipulation of over- lap for polygonal clustering problems allows us to demonstratesurprising behavior with LIME. Although overlap is currently only defined for polygonal clusters in two dimensions, the def-inition generalizes easily to other polytopes and a more refinedcomputational approach may allow us to further investigate thebehavior of LIME.

Future work could continue to develop this method of generating and manipulating tabular data with more features and adding more robust support for higher-dimensional polytopes. In addition, these methods could be used to investigate the behavior of other post-hoc black-box explainers like SHAP [2], as well as other classification methods including polygonal modelling [1] or dimensionality reduction.

Appendix A

Implementation and the Makeoverlap Algorithm

Algorithm 1 and Algorithm 2 contain the pseudocode for our method of manipulating the overlap of a polygonal clustering problem. Note that the current implementation supports additional features not described here.

The data generation methods described in this paper and code for this work can be found at github.com/he-jesse/polydata. Interface design was based in part on the datasets module of the sci-kit learn project [4] and the implementation uses a number of well-known Python libraries [5, 6, 8].

Algorithm 1: MakeOverlap

Data: A point set X, a polygon set P, a function $f : X \to P$, a range (o_m, o_M) of overlap values
Result: A point set X^I and polygon set P^I such that $o_m <$ overlap $< o_M$

```
begin
    X^I ← X  P^I ← P  s_m ← −1
    s_M ← 1
    s ← (s_m + s_M)/2
    while o_m >overlap(X^I, P^I)  or overlap(X^I, P^I)> o_M  do
        if overlap(X^I, P^I) < o_m then
            s_M ← s
        end
        else if overlap(X^I, P^I) > o_M then
            /* ε is a small threshold   */
            if s_M − s_m < ε then
                s_M ← s_M + 1
            end
            s_m = s
        end
        s ← (s_m + s_M)/2
        for x ∈ X^I do
            x ← x + s f(x).centroid())
        end
        for p ∈ P^I do
            p ← p + s p.centroid()
        end
    end
    return X', P'
end
```

Algorithm 2: overlap

Data: A point set X, a polygon set P
Result: The proportion c of overlap

```
begin
    Y ← ∅
    for {p1, p2} ∈ P^(2) do
        for x ∈ X do
            if p1.contains(x) and p2.contains(x) then
                Y ← Y ∪ {x}
            end
        end
    end
    return |Y|/|X|
end
```

Acknowledgment

This work is supported by the National Science Foundation under grant number CNS-1757945 and was inspired by work

performed by the second author under Subcontract number 612048 from the Los Alamos National Laboratory.

6. References

[1] Akdag, F., Eick, C. F., and Chen, G. (2014). Creating polygon models for spatial clusters. In T. Andreasen, H. Christiansen, J.-C. Cubero, and Z. W. Ras´ (Eds.), *Foundations of intelligent systems* (pp. 493–499). Springer International Publishing.

[2] Belle, V., and Papantonis, I. (2021). Principles and practice of explainable machine learning. *Frontiers in Big Data*, *4*, 39. https://doi.org/10.3389/fdata.2021.688969. (Access Date: 27July 2021).

[3] Biecek, P., and Burzykowski, T. (2021). *Explanatory Model Analysis*. Chapman; Hall/CRC. https://pbiecek.github.io/ema/. (Access Date: 27 July 2021).

[4] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller,A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A.,Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B.,and Varoquaux, G. (2013). API design for machine learning software: Experiences from the scikit-learn project. *ECMLPKDD Workshop: Languages for Data Mining and Machine Learning*, 108–122.

[5] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del R´ıo, J. F., Wiebe, M., Peterson, P., .Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. DOI: 10.1038/s41586-020-2649-2.

[6] Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science and Engineering*, *9*(3), 90–95. DOI: 10. 1109/MCSE.2007.55.

[7] Lundberg, S., and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *CoRR*, *abs/1705.07874*. http://arxiv.org/abs/1705.07874. (Access Date: 27 July 2021).

[8] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit- learn: Machine learning in Python. *Journal of Machine Learn-ing Research*, *12*, 2825–2830.

[9] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why Should I Trust You?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD Interna- tional Conference on Knowledge Discovery and Data Mining*, 1135–1144. https://doi.org/10.1145/2939672.2939778.

[10] Shrikumar, A., Greenside, P., Shcherbina, A., and Kun- daje, A. (2016). Not just a black box: Learning important features through propagating activation differences. *CoRR*, *abs/1605.01713*. http : / / arxiv. org / abs / 1605 . 01713. (Access Date: 12 September 2021).

[11] Slack, D., Hilgard, S., Jia, E., Singh, S., and Lakkaraju, H. (2020). Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 180–186. https://doi. org/10.1145/3375627.3375830.