Comparing LIME and SHAP using Synthetic Polygonal Data Clusters

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

Abstract

As machine learning classifiers continue to enjoy tremendous popularity and expand their domains of ap-plication, their reach has extended to many facets of our lives and are impacting society at large. However, the most powerful models remain inscrutable: in spite of high accuracy, it is difficult, if not impossible, to ex-plain the classification of particular instances. In many situations, e.g., a medical diagnosis or the arrest of a citizen that involve a potential loss of life or freedom, explanations of the predictions are crucial. This has led to research interest in eXplainable AI (XAI) which seeks the interpretability of explanations, i.e., that a prediction made by a classifier should be understand-able in terms of the inputs it was fed. Two popular approaches are LIME and SHAP, both of which offer a model-agnostic framework for explaining almost any classifier. However, this raises the question of the ex-plainers' trustworthiness. In this paper, we probe the limitations and vulnerabilities of these two post-hoc ex-plainers. We have built a tool that generates synthetic tabular data sets which enables us to probe the expla-nation system opportunistically based on its architec-ture. Here, we reveal a scenario where LIME's expla-nation violates local faithfulness, but where SHAP ef-fectively rectifies this disparity.

1. Introduction

Since machine learning tools are both widely avail-able and easy to use, they can be deployed wherever there is a collection of data and an interest in its analy-sis. The most powerful machine learning methods are indeed impressive as far as classification accuracy is concerned, but offer an opaque *black-box* interface, de-fying scrutiny and offering little to no explanation. Yet, when they are involved in decisions involving hiring and firing, arrest, or irreversible medical procedures, this lack of explanation amounts to belittling the lives, careers, and freedom of citizens; this can shake the foundations of society and sow distrust in technology as a whole. Thus, there is an urgent need for explana-tion frameworks.

Consequently, researchers have articulated the need for transforming Artificial Intelligence (AI) into *eX-plainable AI* (XAI) which seeks the interpretability of explanations.

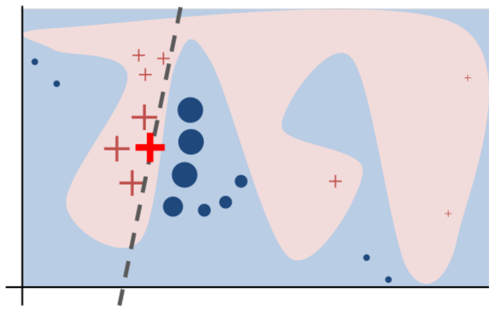


Figure 1. Illustration to build intuition for LIME from Ribeiro, Singh, and Guestrin [10]. The black-box model's prediction for the red cross is based on a com-plex decision boundary, illustrated by the blue/pink background. LIME samples nearby instances, gets the model's predictions, and attempts to create an explana-tion that is locally faithful

That is, XAI seeks to ensure that the re-sults of learning models are explainable in terms of the inputs that were provided. Since humans are familiar with the semantics of these inputs, it is presumed that this will provide satisfactory explanations. This has led to promising recent approaches [8, 10, 11], but this in turn raises questions about the trustworthiness of the explainers and motivates us to probe their limitations and vulnerabilities.

Towards that end, we have built and deployed a tool for generating synthetic tabular data sets. Our insight is that to evaluate such explanations, and more gener-ally, to conduct research in machine learning, a key tool needed is the ability to generate synthetic data sets that we can manipulate, intuitively grasp, and that enables visualization.

We have chosen the family of classification problems involving n-dimensional datasets for which the bound-aries of identifying classes can be described geomet-rically using polyhedrons; with a current emphasis on the 2-dimensional case for ease of development and vi-sualization. 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 "polygonal clustering" to refer to the task of classify-ing such data. We will also allow a polygonal cluster to have a certain fraction of 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) [10], which attempts to identify the contribu-tion of individual features towards a classifier’s predic-tion by observing the behavior of the classifier around perturbations of the original data point. An intuitive illustration from Ribeiro, Singh, and Guestrin [10] is given in Figure 1.

LIME has seen adoption in image and text analy-sis, but its application to tabular data has certain lim-itations arising from the inapplicability of its key con-cept of a *superpixel* in the context of general tabular data. Different methods of perturbing or identifying a local neighborhood for a data point can create dif-ferent 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 inves-tigate LIME’s ability to explain tabular classification models.

Earlier, we have reported on our use of a syn-thetic dataset that exposed an important vulnerability in LIME [6]. Here we extend our investigation to include SHAP, another promising explanation framework.

Lundberg and Lee’s SHAP, or *SHapley Additive Ex-Planations*, seeks to unify various explainer methods, and their Kernel SHAP in particular seeks to rectify potential violations of local accuracy in LIME’s expla-nations [8]. Because LIME’s method of local explana-tion heuristically performs additive feature attribution locally, local accuracy is not guaranteed. SHAP, how-ever, avoids the heuristic approach and instead com-putes a *Shapely kernel*, which recovers the *Shapely val-ues* so the resulting additive feature attribution matches the classifier’s behavior.

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 clus-ter, 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 data sets exist that can be used for such prob-lems, there is no robust way to randomly generate and customize new data sets with polygonal clusters.

In this paper, we describe a software tool which uses a simple but effective method of generating such classi-fication problems, and we demonstrate its utility in in-vestigating the behavior of LIME on a black box clas-sifier by producing examples where LIME’s explana-tions fail to be locally faithful to the classifier’s behav-ior, comparing it to SHAP. 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 [10] takes a sample and cre-ates 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 power-ful tool that enables comparison among machine learn-ing models. Accuracy notwithstanding, a model that relies on features that are relevant to humans is viewed as trustworthy, while one that relies on accidentally cor-related features is considered dangerous, even when both display statistically equivalent accuracy. In ad-dition, such post-hoc explanations can potentially un-cover bias that could be implicit in classifiers. Lund-berg and Lee’s SHAP [8] computes an importance met-ric for each feature for a given prediction. Their tech-nique is to consider all relevant subsets of features that contain a given feature to assess its contribution.

However, Slack et al. [12] demonstrated that such explainers can themselves be fooled. They demon-strated a scenario in which an adversarial entity could demonstrate an explanation of their choosing, for ex-ample, an explanation that lets a biased classifier go un-detected. 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.

Other synthetic data generation software includes the datasets module of the scikit-learn Python library [9], 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 “center” 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 uniformly at random from $[0, 2\pi)$, giving us the polar coordinates for each vertex. To en-sure that the polygon contains the central point about

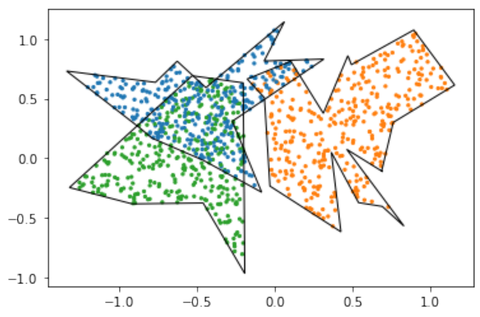


Figure 2. Three uniform balanced polygonal clusters.

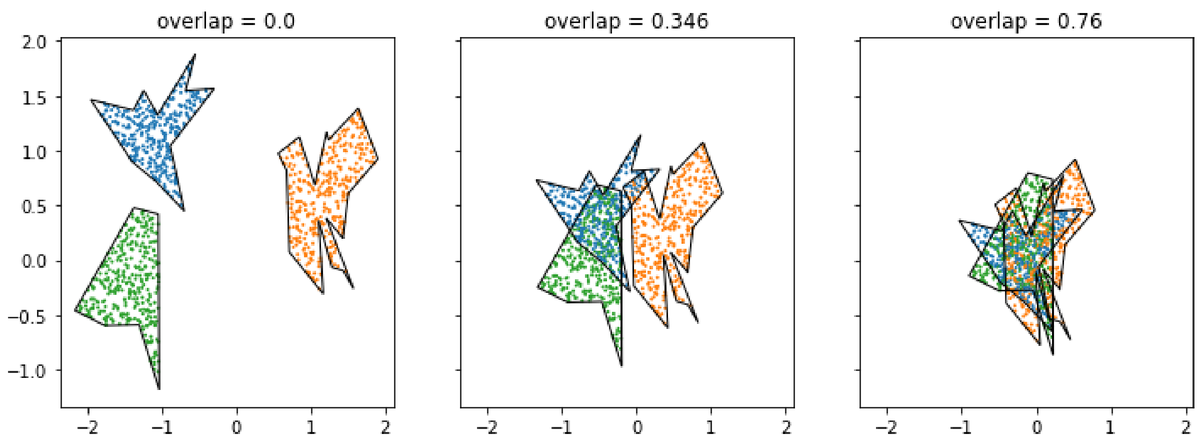


Figure 3. Dataset from Figure 2 with clusters shifted to create different overlaps. From left to right: dataset with no overlap, original dataset, dataset with high overlap

which it is generated, we check that no consecutive angles θ_1, θ_2 have a difference of more than π radi-ans 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 [7] to represent polygonal paths and test if a polygon con-tains a given point. This also allows for easy visualiza-tion in plots made using Matplotlib and for integration with other Python libraries, including efficient vector-ized operations with NumPy [5]. An example is shown in Figure 2.

3.1. Controlling Overlap

One additional task in generating new classification problems 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 the *overlap* between clusters. Given a set $X \subseteq \mathbb{R}^2$ 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:

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

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 5. 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.

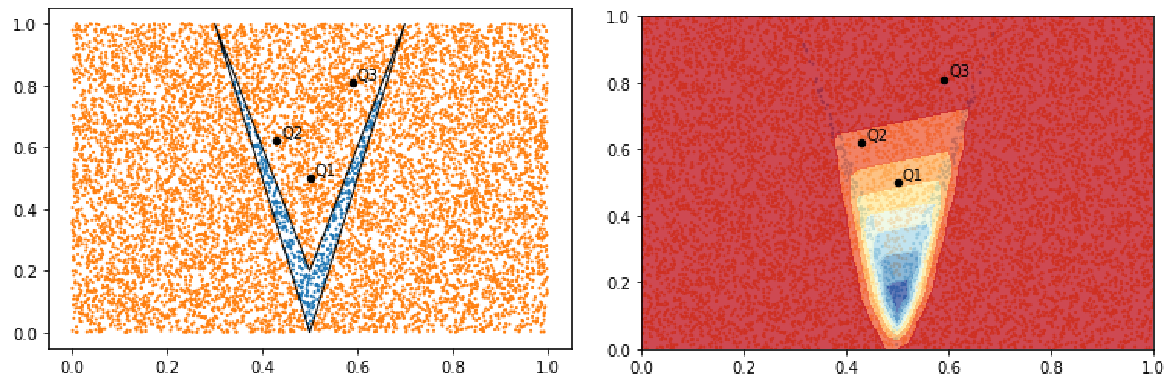
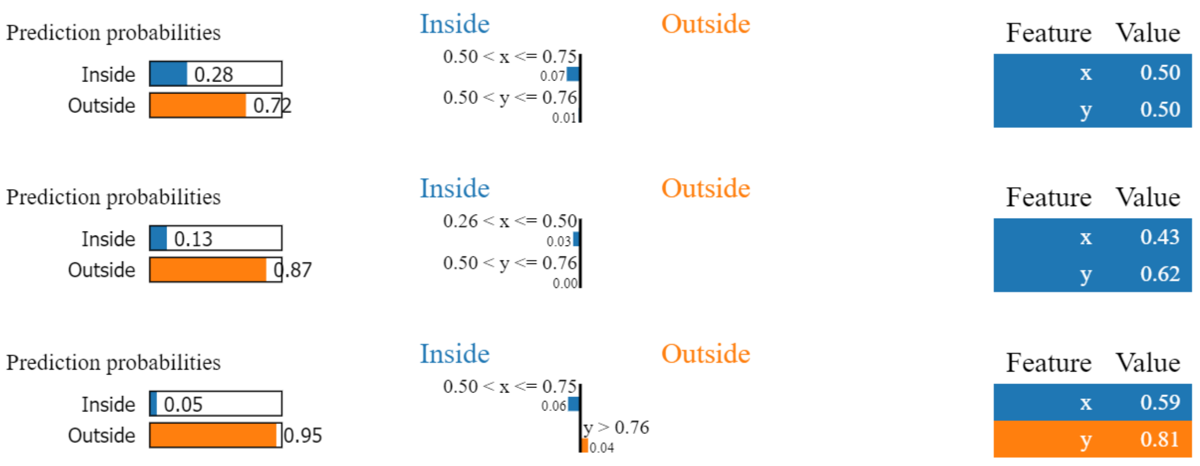


Figure 4. A classification problem featuring a "V"-like polygon with sample points $Q_1 = (0.5, 0.5), Q_2 = (.43, .62), Q_3 = (.59, .81)$ (left), the classifier's decision function (right)



Feature	Value
x	0.50
y	0.50

Feature	Value
x	0.43
y	0.62

Feature	Value
x	0.59
y	0.81

Figure 5. Left: classifier predictions for Q_1, Q_2 , and Q_3. Rest: LIME's corresponding explanations for these predictions. Blue features are ones which LIME identifies as increasing the classifier's predicted probability of being inside the polygon while orange features are ones which LIME identifies as increasing the classifier's predicted probability for being outside the polygon

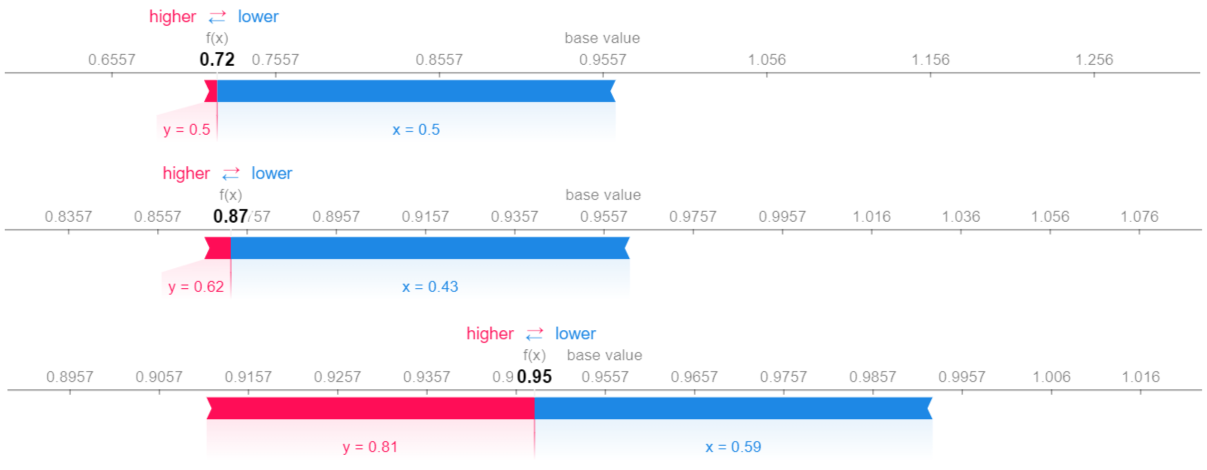


Figure 6. Force plots of SHAP's explanations at Q_1, Q_2 , and Q_3. The base value of 0.9557 represents the classifier's average prediction that 0.9557 of all points lie outside the polygon. The blue bar indicates the strength of features which push the classifier towards a classification of inside while the red bar indicates the strength of features which push the classifier farther towards a classification of outside

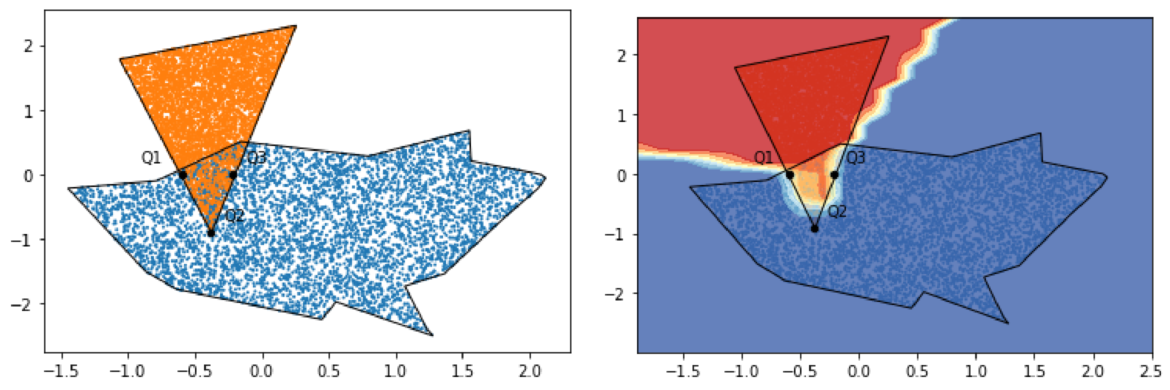


Figure 7. A binary polygonal classification problem with an overlap of between 0.1 and 0.2. A plot of the dataset and polygons (left) next to the classifier's decision boundary (right) with marked points $Q_1 = (-0.6, 0)$, $Q_2 = (-0.38, -0.9)$, and $Q_3 = (-0.22, 0)$

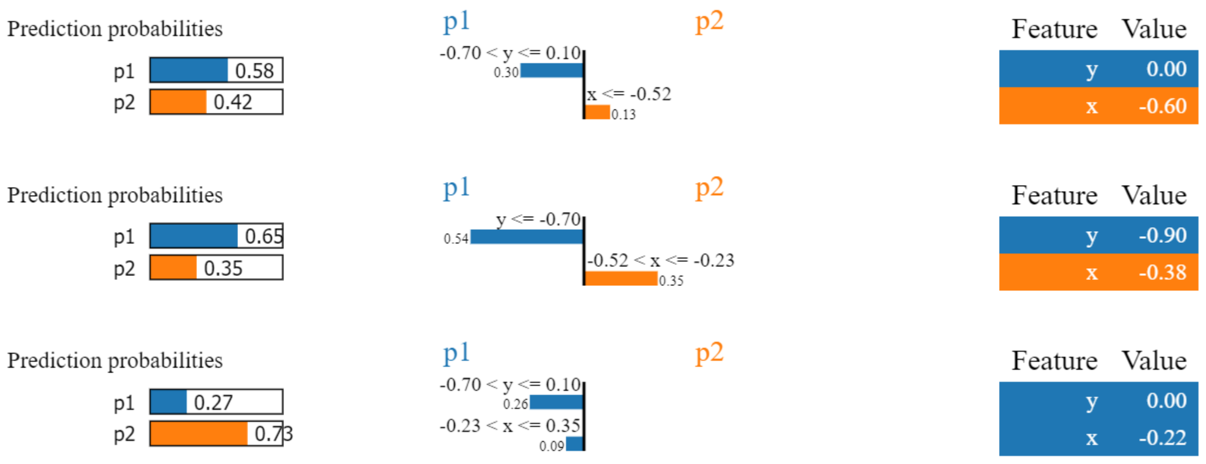


Figure 8. Left: classifier predictions for Q_1, Q_2 , and Q_3. Rest: LIME's corresponding explanations for these predictions. Blue features correspond to p_1 while orange features correspond to p_2

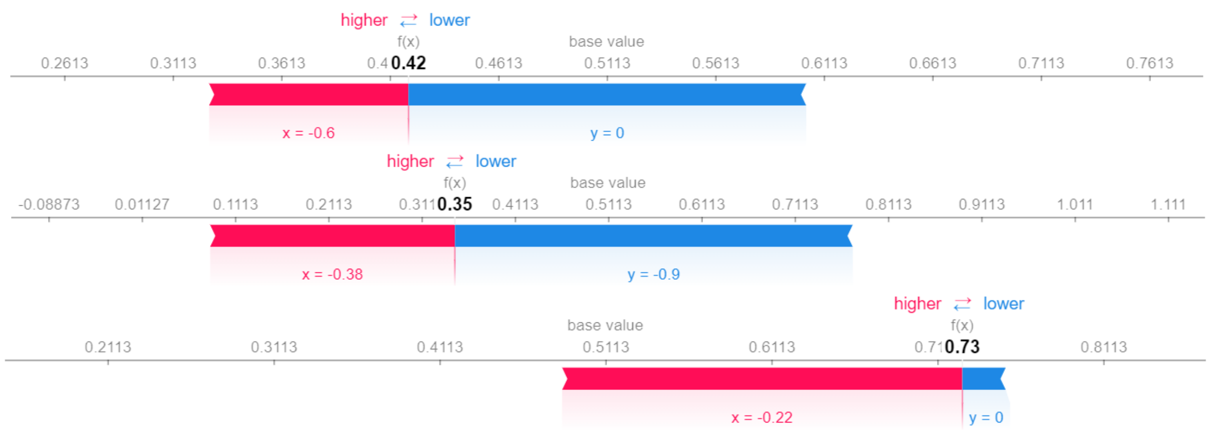


Figure 9. Force plots of SHAP's explanations for the classifier's probabilities of Q_1, Q_2, and Q_3 lying in p_2. Features which contribute to a prediction of p_1 are blue and those which contribute to a prediction of p_2 are red

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 points in challenging areas and ask both LIME and Kernel SHAP 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 may contradict the classifier's prediction: at Q_1 and Q_2,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. For example, at Q_1, Figure 5 shows that LIME's explanation for the classifier's prediction of outside, with a probability of 0.72, is based on the x and y values for Q_1 locally being individually correlated with an increased in the classifier's predicted probability of outside. However, because SHAP values are computed specifically to recover additive feature attribution values which match local predictions, the SHAP explanations accurately reflect the model's output.

Next, we create a classification problem and ensure it has nonzero overlap according to our description in the previous section. We generate a binary classification problem with two polygonal clusters p_1 and p_2 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 pre-diction of a particular point using LIME and compare it to the classifier's actual decision function as shown in Figure 7, with LIME's explanations in Figure 8 and SHAP's in Figure 9. The SHAP explanations show the additive feature attribution for the probability that each point is in p_1. All three points were chosen to be difficult for the classifier. Each lies within the bound-aries of both p_1 and p_2 but is very close to the bound-ary of p_2, and this difficulty is reflected when examin-ing the classifier's decision function. Both LIME's and SHAP's explanations at Q_1 and Q_2 reflect the classifier's uncertainty at those points, but LIME's behavior at Q_3 is unintuitive: the MLP classifies Q_3 in p_2 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 p_1. This explanation contradicts the actual prediction of the classifier at Q_3, violating LIME's local faithfulness. SHAP's explanation, however, matches the model's output.

We finally compare the behavior of LIME and SHAP by engineering a multi-dimensional data set based on the previous data set with added features: two are exact duplicates of x and y (x_{dup} and y_{dup}, respectively) then three redundant features r_1, r_2, r_3 which are random linear combinations of x and y with added Gaussian noise of standard deviations 0, 0.5, and 1.0, and an uncorrelated feature rand drawn uniformly at random from $[0, 1)$. An MLP was trained on a subset of the data and LIME and SHAP were asked to explain the classifier's behavior at the points

$$Q'_1 = (-.6, 0, -.6, 0, -.0458, .864, .811, .865)$$
$$Q'_2 = (-.38, -.9, -.38, -.9, .197, -.613, .412, .817)$$
$$Q'_3 = (-.22, 0, -.22, 0, -.017, -.468, -.789, .368)$$

representing Q_1, Q_2, and Q_3 with the generated extra features (each coordinate is, respectively, x, y, x_{dup}, y_{dup}, r_1, r_2, r_3, rand). LIME's explanations are shown in Figure 10 and SHAP's are shown in Figure 11.

LIME's explanation of the classifier's behavior at Q' $_1$ also violates local faithfulness, attributing a prediction of p_1 to the features x and x_{dup} which are correlated with p_2. SHAP also identifies x and x_{dup} as being the most prominant features correlated with p_2, but identi-fies several others as contributing to the classifer's pre-diction of p_1.

Both LIME and SHAP identify x, x_{dup}, y, and y_{dup} as being the most important features in the classifier's prediction at Q'_2, agreeing that y and y_{dup} contribute towards the ultimate prediction of p_1 over the contribution of x and x_{dup} to a prediction of p_2.

Q'_3 remains challenging for LIME, but we see that it ascribes to both y and y_{dup} similar importance, and as-signs very little to the rest of the features. Interestingly, however, LIME identifies x_{dup} as contributing to a clas-sification of p_2 while x is still identified as contributing to a classification of p_1. SHAP, as before, reflects the output of the classifier, but assigns almost no impor-tance to y and little to y_{dup}, especially in comparison to the redundant features r_1 and r_2, in contrast to LIME.

We can also combine many SHAP explanations to see which features are identified globally by the classifier. Observing Figure 12, we see that y and y_{dup} are consistently prominent features of predictions for p_2 while y_{dup}, x, and x_{dup} contribute more towards a prediction of p_1.

5. Conclusion and Future Work

The examples given show that LIME's explanations may violate local faithfuless under certain conditions, but that LIME's difficulties in its heuristic assignment of additive feature attribution are effectively rectified by SHAP.

The method of generating and manipulating polygo-nal tabular data we have presented is effective at gen-erating simple example data sets for clustering prob-lems. The generated data sets are easily visualized and can be customized to fit different geometric and statis-tical structures. Using polygons to specify the geomet-ric structure allows for customization of cluster shape,

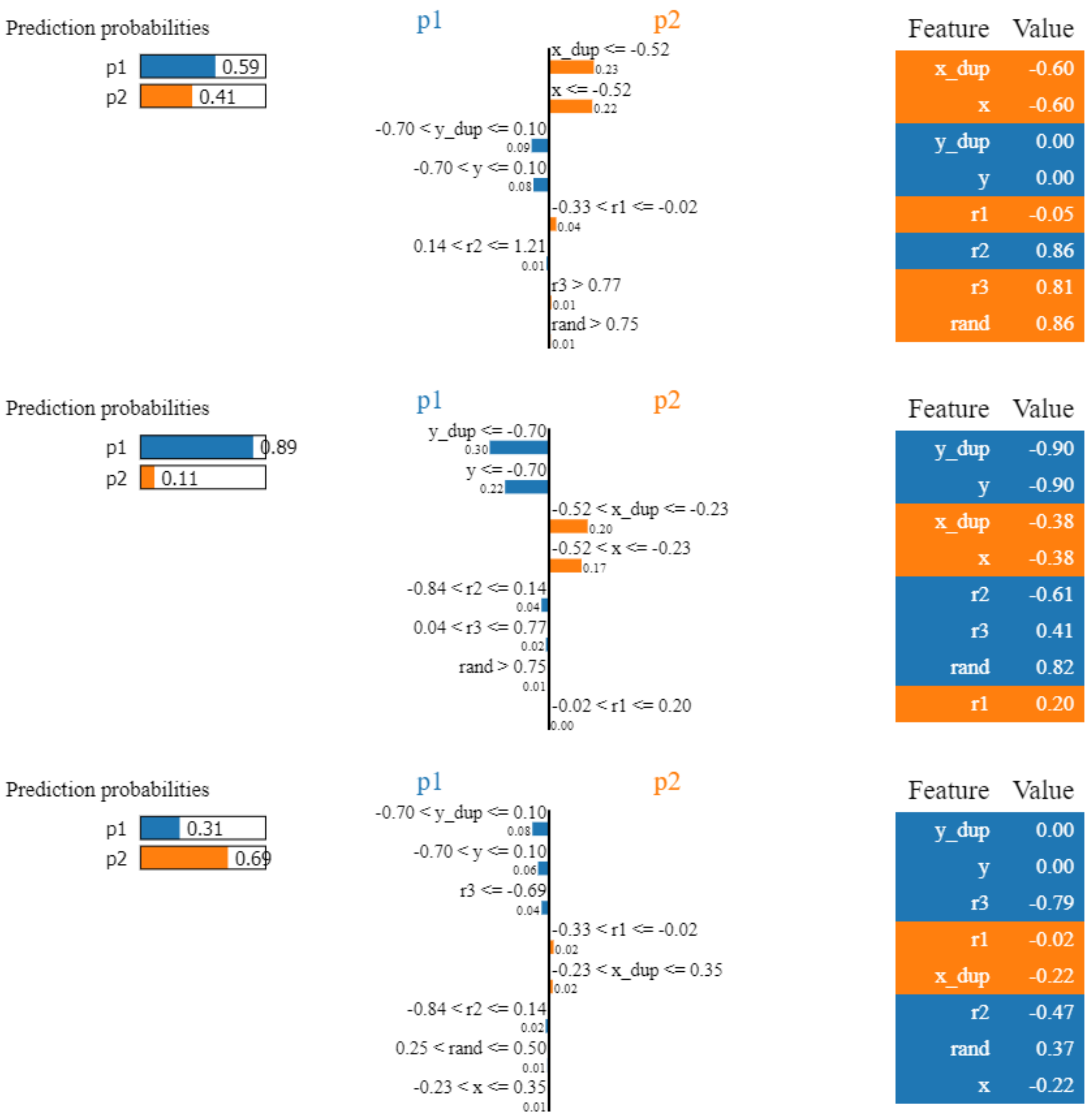


Figure 10. LIME's explanations at Q'_1, Q'_2, and Q'_3, representing Q_1, Q_2, and Q_3 with generated extra features

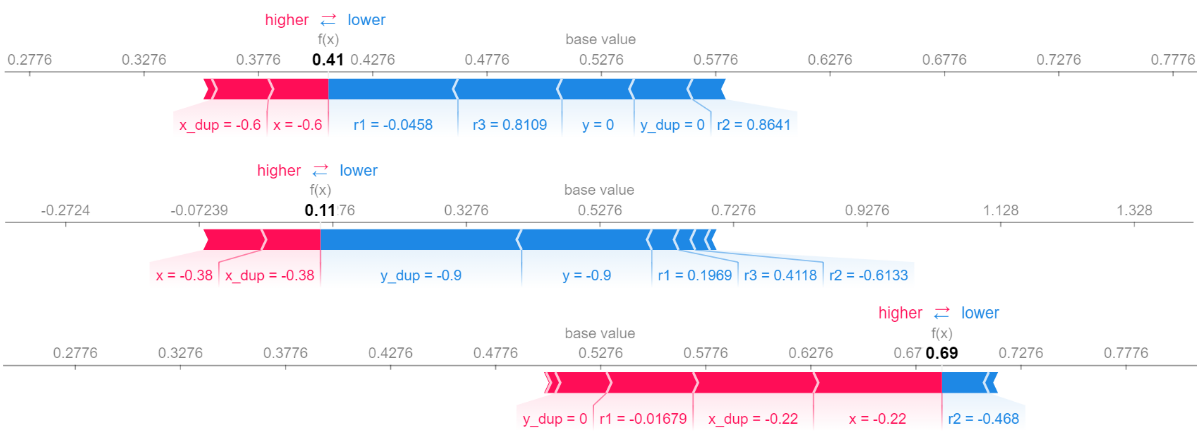


Figure 11. SHAP's explanations at Q'_1, Q'_2, and Q'_3, representing Q_1, Q_2, and Q_3 with generated extra features

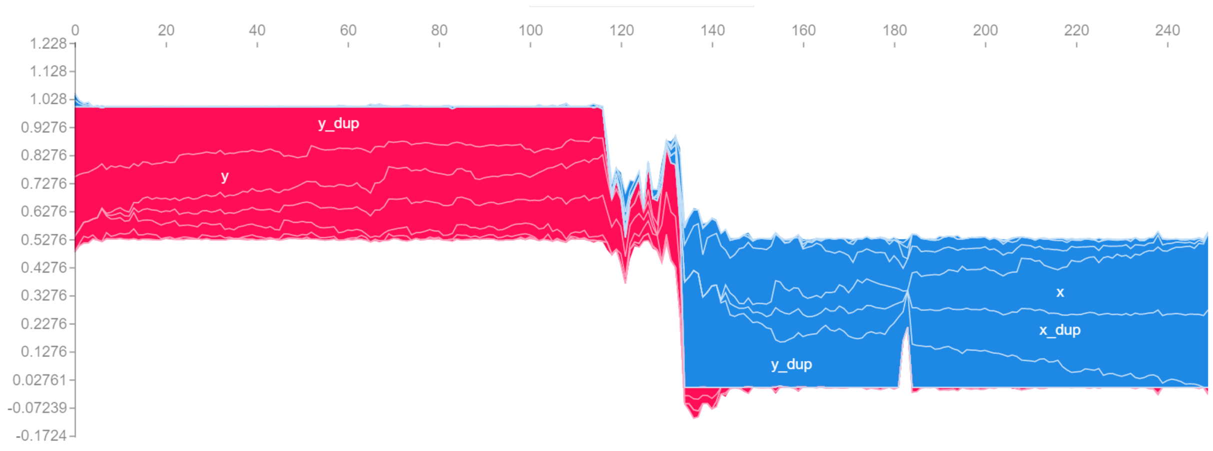


Figure 12. Combined force plot of SHAP's explanations at 250 test points for the classifier, ordered by similarity

and using overlap allows for customization of the difficulty of a generated classification problem. In particular, our simple definition and manipulation of overlap for polygonal clustering problems allows us to demonstrate surprising behavior with LIME. Although we have only presented a definition for polygonal clusters in two dimensions, the definition generalizes easily to other polytopes and a more refined computational approach may allow us to create more multidimensional data sets with which we can continue to use in probing explainers. In addition, these methods could be used to with 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, 7, 9].

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.

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 $X'$ and polygon set $P'$ such
        that $o_m <$ overlap $< o_M$
begin
    $X' \leftarrow X$ $P' \leftarrow P$ $s_m \leftarrow -1$
    $s_M \leftarrow 1$
    $s \leftarrow (s_m + s_M)/2$
    while $o_m >$overlap$(X', P')$ or
      overlap$(X', P')> o_M$ do
        if overlap$(X', P') < o_m$ then
            $s_M \leftarrow s$
        end
        else if overlap$(X', P') > o_M$ then
            /* ε is a small
            threshold                    */
            if $s_M - s_m < \varepsilon$ then
                $s_M \leftarrow s_M + 1$
            end
            $s_m = s$
        end
        $s \leftarrow (s_m + s_M)/2$
        for $x \in X'$ do
            $x \leftarrow x + s \cdot f(x)$.centroid()
        end
        for $p \in P'$ do
            $p \leftarrow p + s \cdot p$.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
    $Y \leftarrow \varnothing$
    for $\{p_1, p_2\} \in P^{(2)}$ do
        for $x \in X$ do
            if $p_1$.contains($x$) and $p_2$.contains($x$) then
                $Y \leftarrow Y \cup \{x\}$
            end
        end
    end
    return $|Y|/|X|$
end
```

References

[1] Akdag, F., Eick, C. F., and Chen, G. (2014) Creating polygon models for spatial clusters. In T. An-dreasen, H. Christiansen, J.-C. Cubero, and Z. W. Ra´s (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: 27 July 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 de-sign for machine learning software: Experiences from the scikit-learn project. ECML PKDD Workshop: Lan-guages for Data Mining and Machine Learning, 108–122.

[5] Harris, C. R., Millman, K. J., van der Walt, S. J., Gom-mers, 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. https://doi.org/10.1038/s41586-020-2649-2.

[6] He, J., and Mazumdar, S. (2021) Using polygonal data clusters to investigate lime. 2021 International Con-ference on Information Society (i-Society).

[7] Hunter, J. D. (2007) Matplotlib: A 2d graphics en-vironment. Computing in Science and Engineering, 9(3), 90–95. https://doi.org/10.1109/MCSE.2007.55.

[8] Lundberg, S., and Lee, S. (2017) A unified ap-proach to interpreting model predictions. CoRR, abs/1705.07874. http : / / arxiv. org / abs / 1705 . 07874.(Access Date: 27 July 2021).

[9] 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 Duch-esnay, E. (2011) Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

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

[11] Shrikumar, A., Greenside, P., Shcherbina, A., and Kundaje, A. (2016) Not just a black box: Learning im-portant features through propagating activation differ-ences. CoRR, abs/1605.01713. http://arxiv.org/abs/ 1605.01713. (Access Date: 12 September 2021).

[12] Slack, D., Hilgard, S., Jia, E., Singh, S., and Lakkaraju, H. (2020) Fooling LIME and SHAP: Ad-versarial 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.