Data Valuation with Shapley Values

Simon Park JUHYUNP@PRINCETON.EDU
Department of Mathematics
Princeton University

Daniel Yang DANIEL.YANG@INF.ETHZ.CH
Department of Computer Science
ETH Zurich

Abstract

The Shapley value is a concept from cooperative game theory that measures the amount of contribution from each player to the collective utility. Recent research attempts to use Shapley values in data valuation in the ML setting: evaluating the contribution of each training data point in improving the model performance. In this project, we test how robust the Shapley values are with respect to different training settings. We also devise methods to use the Shapley values to detect if the dataset contains mislabeled data or if the model is overfitting/underfitting on a specific dataset.

I pledge my honor that this paper represents my own work in accordance with University regulations. /s Simon Park, Daniel Yang

Contents

1. Introduction

Data is one of the key components of machine learning (ML), along with the training model. Whereas choosing a good training model has long been the focus of ML research, the importance of choosing a good training data has been overlooked in comparison. Data valuation addresses this problem by assigning the *importance* or *worth* of each data point for the learning.

The Shapley value is a concept from cooperative game theory, which measures the importance of each player for achieving a collective goal. In recent years, the Shapley value has been used for data valuation, where training data points are interpreted as *players* collaborating to enhance the performance of a training model (Rozemberczki et al., 2022). Previous works on applying Shapley value to data valuation show that this approach is generally better than other baseline approaches like the Leave-one-out method (Ghorbani and Zou, 2019).

But unlike other methods, Shapley values are dependent on the game formulation (i.e., exactly how the training setting is interpreted as a cooperative game). In this paper, we test how robust Shapley values are with respect to different game formulations. We additionally investigate if we can detect the existence of mislabeled data from the Shapley values.

2. Background

In this section we formally define the Shapley value, in particular for the data valuation setting. Then we present the research questions that this paper intends to answer.

2.1 Shapley Value

We first define the fundamentals of a cooperative game.

Definition 1 (Cooperative Game) *Given a finite set $\mathcal{D} = \{1, \cdots, n\}$ of n **players** and a **characteristic function** $v : 2^{\mathcal{D}} \to \mathbb{R}_{\geq 0}$ where $v(\emptyset) = 0$, any subset $S \subseteq \mathcal{D}$ is called a **coalition**; $v(S)$ is called the **valuation** of the coalition S; and the tuple $G = (\mathcal{D}, v)$ is called a **cooperative game**.*

The players are potential participants of a cooperative game. They may choose to, or choose not to participate in the game, but once they do, they collaborate with other participants to achieve a common goal. The characteristic function valuates the strength of this coalition and defines the total utility that will result from it. Then this utility is distributed among all participating players as a "dividend". We additionally want to assume that for each player, the amount of devidend that they receive is the same across all coalitions they choose to participate in. We define this amount as the **payoff** of that player.

Definition 2 (Payoff Vector) *Given a cooperative game $G = (\mathcal{D}, v)$, a vector $\mathbf{u} \in \mathbb{R}^n$ is called a **payoff vector** if for every coalition $S \subseteq \mathcal{D}$, we have $\sum_{i \in S} u_i \leq v(S)$ and the value u_i at the i-th coordinate of $\mathbf{u}$ is called the **payoff** of player i*

The condition in the Definition 2 says that the amount of payoff we distribute to the participants cannot exceed the total amount of utility generated from the game. Of course,

for each game, there are multiple ways to define the payoff vector. The Shapley value is one such way.

Definition 3 (Shapley Value: Permutation) *Given a cooperative game $G = (\mathcal{D}, v)$, let $\Pi(\mathcal{D})$ be the set of all permutations on $\mathcal{D}$. For each $i \in \mathcal{D}$ and $\pi \in \Pi(\mathcal{D})$, we define*

$$\pi_{\leq i} = \{j \in \mathcal{D} \mid \pi(j) \leq \pi(i)\} \qquad \pi_{<i} = \{j \in \mathcal{D} \mid \pi(j) < \pi(i)\}$$

*Then the **Shapley value** of player i is defined as*

$$\begin{aligned} s_i &= \mathop{\mathbb{E}}_{\pi \in \Pi(\mathcal{D})} [v(\pi_{\leq i}) - v(\pi_{<i})] \\ &= \frac{1}{|\Pi(\mathcal{D})|} \sum_{\pi \in \Pi(\mathcal{D})} (v(\pi_{\leq i}) - v(\pi_{<i})) \end{aligned}$$

Imagine that given a permutation π on $\mathcal{D}$, the players arrive in the order specified by that permutation, and everyone chooses to participate. When the newly arriving player declares to participate, the total utility of the game changes. The Shapley value of player i is defined as its marginal utility in expectation over a randomly sampled permutation, or equivalently its marginal utility averaged over all possible permutations.

Assume we have two permutations π, π' such that the set of players that arrive before i is the same (i.e., $\pi_{<i} = \pi'_{<i}$). Then notice that the summand $v(\pi_{\leq i}) - v(\pi_{<i})$ in Definition 3 is equal for both permutations. This motivates the following equivalent definition of a Shapley value.

Definition 4 (Shapley Value: Subset) *Given a cooperative game $G = (\mathcal{D}, v)$, the **Shapley value** of player i can also be defined as*

$$\begin{aligned} s_i &= \mathop{\mathbb{E}}_{k \in [n]} \left[\mathop{\mathbb{E}}_{\substack{S \subseteq \mathcal{D} \setminus \{i\} \\ |S| = k-1}} [v(S \cup \{i\}) - v(S)] \right] \\ &= \sum_{S \subseteq \mathcal{D} \setminus \{i\}} \frac{1}{n\binom{n-1}{|S|}} (v(S \cup \{i\}) - v(S)) \end{aligned}$$

Here we understand the Shapley value as the marginal utility of player i averaged over the set S it gets added to. Each summand will be weighted by the probability that it appears in a random permutation. It is easy to verify that the two definitions of Shapley values are equivalent. The Shapley value is crucial in cooperative game theory because it is known to be the unique payoff vector that satisfies three intuitive properties: efficiency, symmetry, and monotonicity. (Shapley, 1953)

2.2 Data Valuation

The training process of a ML model can be interpreted as a cooperative game. The set of players is the set of training data points. The objective of the game is to improve the model being trained. The valuation of a subset S of the training dataset is defined as some form

of metric to analyze the performance of the model when trained on S. One example would be the mean accuracy of the model when evaluated on a test dataset. Then the Shapley value for data point i is the average marginal increase or decrease in the performance of the model when i is included in the training dataset. We formally write this as follows:

Definition 5 (Training Model) *Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ be a finite set of n training data points where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. A function $M_{\mathcal{D}} : 2^{\mathcal{D}} \times \mathbb{R}^d \to \mathbb{R}$ is called a **model** on $\mathcal{D}$. For each $S \subseteq \mathcal{D}$, we define a function $M_S : \mathbb{R}^d \to \mathbb{R}$, the **model** trained on S, such that $M_S(\mathbf{x}) = M_{\mathcal{D}}(S, \mathbf{x})$ for any $\mathbf{x} \in \mathbb{R}^d$. Also, a function $v_{M_{\mathcal{D}}} : 2^{\mathcal{D}} \to \mathbb{R}$ is called the **evaluation metric** of $M_{\mathcal{D}}$.*

Example 6 *The mean accuracy of a model $M_{\mathcal{D}}$ on test dataset $\mathcal{D}_{test} = \{(\mathbf{x}_i^*, y_i^*)\}_{i=1}^k$ can be calculated as*

$$v_{M_{\mathcal{D}}, \mathcal{D}_{test}}(S) = \frac{1}{k} \sum_{i=1}^{k} \mathbb{1}\left[M_S(\mathbf{x}_i^*) = y_i^*\right]$$

and is an example of a performance metric of $M_{\mathcal{D}}$ evaluated on $\mathcal{D}_{test}$.

Definition 7 (Shapley Value) *Given a training dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, a model $M_{\mathcal{D}}$ on $\mathcal{D}$, and an evaluation metric $v_{M_{\mathcal{D}}}$, we identify $\mathcal{D}$ with $\{1, \cdots, n\}$ with the canonical bijection $f : i \mapsto (\mathbf{x}_i, y_i)$ and define $v : 2^{[n]} \to \mathbb{R}$ such that $v(S) = v_{M_{\mathcal{D}}}(f(S))$ for any $S \subseteq [n]$. We abuse notation and denote the cooperative game $G = ([n], v)$ as $G = (\mathcal{D}, M_{\mathcal{D}}, v_{M_{\mathcal{D}}})$. Also, the **Shapley value** s_i for a training point $(\mathbf{x}_i, y_i)$ in $G = (\mathcal{D}, M_{\mathcal{D}}, v_{M_{\mathcal{D}}})$ is defined as the Shapley value of player i in the cooperative game $G = ([n], v)$.*

2.3 Research Questions

In recent years, there has been an increasing number of works on data valuation with the Shapley value (Ghorbani and Zou, 2019; Rozemberczki et al., 2022), but many of them have only presented their observations about the Shapley values for a very specific training setting. However, from the definition of the Shapley value in Definition 7, we notice that the Shapley value is dependent on the choice of the training dataset $\mathcal{D}$, the model $M_{\mathcal{D}}$, and the evaluation metric $v_{M_{\mathcal{D}}}$. Therefore, it is natural to ask how robust the Shapley values are to a change in the game formulation. Otherwise, any observation about the Shapley value is only specific to the training setting it was defined on. In particular, we investigate how the Shapley value behaves on different datasets and models. Based on some of these results, we also investigate if we are able to obtain useful information about our model or training procedure such as detecting the existence of mislabeled data in the training dataset.

3. Methods

In this section, we define the different formulations of a cooperative game in the ML setting which we used in our experiments. Each of these game formulations $G = (\mathcal{D}, M_{\mathcal{D}}, v_{M_{\mathcal{D}}})$ comprises of a dataset $\mathcal{D}$, a model $M_{\mathcal{D}}$, and an evaluation metric $v_{M_{\mathcal{D}}}$. In addition, we also present the different methods we used to compute the Shapley values given a game formulation G.

3.1 Players: Dataset

To get an initial baseline understanding of our research questions, we focused on binary classification with 2-dimensional synthetic datasets. This not only allows us to visualize the dataset, but it also simplifies the analysis and guarantees a manageable runtime. Throughout our experiments, we used three types of datasets derived from Gaussian distributions, three derived from uniform distributions, and another two derived from uniform distributions more specifically to mimic the behavior of an XOR function.

Out of the three Gaussian datasets, the first two consist of two aligned Gaussians, where the optimal decision boundary is linear. The only difference between these two is that the Gaussians in the second Gaussian dataset are more separated, allowing the models to distinguish the two gaussians more easily. The third dataset consists of two Gaussians aligned "perpendicularly", such that the optimal decision boundary is non-linear.

Definition 8 (Gaussian Datasets) *We define three binary-class datasets $\mathcal{D}_{G1,n}$, $\mathcal{D}_{G2,n}$, $\mathcal{D}_{G3,n}$ each of size $2n$ such that*

1. *$\mathcal{D}_{G1,n}$ - n points $(\mathbf{x}_i, +1)$ such that $\mathbf{x}_i \sim \mathcal{N}\left(\begin{pmatrix}1\\-1\end{pmatrix}, \begin{pmatrix}4 & 2\\2 & 4\end{pmatrix}\right)$ and n points $(\mathbf{x}_i, -1)$ such that $\mathbf{x}_i \sim \mathcal{N}\left(\begin{pmatrix}-1\\1\end{pmatrix}, \begin{pmatrix}4 & 2\\2 & 4\end{pmatrix}\right)$*
2. *$\mathcal{D}_{G2,n}$ - n points $(\mathbf{x}_i, +1)$ such that $\mathbf{x}_i \sim \mathcal{N}\left(\begin{pmatrix}2\\-2\end{pmatrix}, \begin{pmatrix}4 & 2\\2 & 4\end{pmatrix}\right)$ and n points $(\mathbf{x}_i, -1)$ such that $\mathbf{x}_i \sim \mathcal{N}\left(\begin{pmatrix}-2\\2\end{pmatrix}, \begin{pmatrix}4 & 2\\2 & 4\end{pmatrix}\right)$*
3. *$\mathcal{D}_{G3,n}$ - n points $(\mathbf{x}_i, +1)$ such that $\mathbf{x}_i \sim \mathcal{N}\left(\begin{pmatrix}1.25\\-1.25\end{pmatrix}, \begin{pmatrix}4 & 2\\2 & 4\end{pmatrix}\right)$ and n points $(\mathbf{x}_i, -1)$ such that $\mathbf{x}_i \sim \mathcal{N}\left(\begin{pmatrix}-1.25\\1.25\end{pmatrix}, \begin{pmatrix}4 & -2\\-2 & 4\end{pmatrix}\right)$*

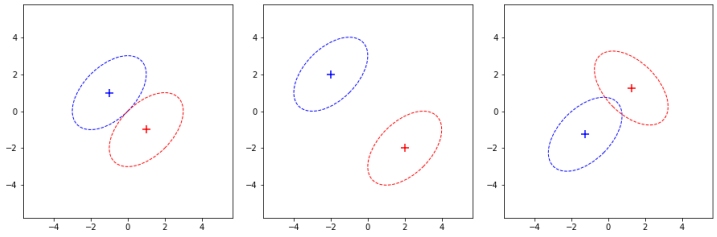

Figure 1: Gaussian datasets $\mathcal{D}_{G1,n}$, $\mathcal{D}_{G2,n}$, $\mathcal{D}_{G3,n}$

We work with three datasets derived from uniform distributions. They differ in the relative size of the two regions that the two classes are sampled from, and in the relative size of the overlapping region.

Definition 9 (Uniform Datasets) *We define three binary-class datasets $\mathcal{D}_{U1,n}$, $\mathcal{D}_{U2,n}$, $\mathcal{D}_{U3,n}$ each of size $2n$ such that*

1. $\mathcal{D}_{U1,n}$ - *n points* $(\mathbf{x}_i, +1)$ *such that* $\mathbf{x}_i \sim \mathcal{U}([-1,1] \times [-1,1])$ *and n points* $(\mathbf{x}_i, -1)$ *such that* $\mathbf{x}_i \sim \mathcal{U}([0,2] \times [0,2])$

2. $\mathcal{D}_{U2,n}$ - *n points* $(\mathbf{x}_i, +1)$ *such that* $\mathbf{x}_i \sim \mathcal{U}([-2,1] \times [-2,1])$ *and n points* $(\mathbf{x}_i, -1)$ *such that* $\mathbf{x}_i \sim \mathcal{U}([0,3] \times [0,3])$

3. $\mathcal{D}_{U3,n}$ - *n points* $(\mathbf{x}_i, +1)$ *such that* $\mathbf{x}_i \sim \mathcal{U}([-3,1] \times [-3,1])$ *and n points* $(\mathbf{x}_i, -1)$ *such that* $\mathbf{x}_i \sim \mathcal{U}([-1,3] \times [0,2])$

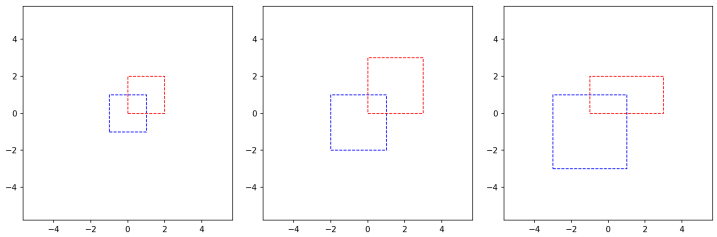

Figure 2: Uniform datasets $\mathcal{D}_{U1,n}$, $\mathcal{D}_{U2,n}$, $\mathcal{D}_{U3,n}$

We further work with two datasets that mimic the behavior of the XOR function, which is not linearly separable. The first XOR dataset, where the data points of different classes are sampled from disjoint regions, is perfectly separable, while the second XOR dataset introduces some overlapping of the classes.

Definition 10 (XOR Datasets) *We define two binary-class datasets* $\mathcal{D}_{X1,n}$, $\mathcal{D}_{X2,n}$ *each of size* $2n$ *such that*

1. $\mathcal{D}_{X1,n}$ - *n points* $(\mathbf{x}_i, +1)$ *such that* $\mathbf{x}_i \sim \mathcal{U}([-5,0] \times [-5,0] \cup [0,5] \times [0,5])$ *and n points* $(\mathbf{x}_i, -1)$ *such that* $\mathbf{x}_i \sim \mathcal{U}([-5,0] \times [0,5] \cup [0,5] \times [-5,0])$

2. $\mathcal{D}_{X2,n}$ - *n points* $(\mathbf{x}_i, +1)$ *such that* $\mathbf{x}_i \sim \mathcal{U}([-5,0.5] \times [-5,0.5] \cup [-0.5,5] \times [-0.5,5])$ *and n points* $(\mathbf{x}_i, -1)$ *such that* $\mathbf{x}_i \sim \mathcal{U}([-5,0.5] \times [-0.5,5] \cup [-0.5,5] \times [-5,0.5])$

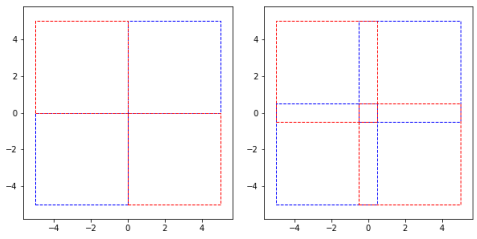

Figure 3: XOR datasets $\mathcal{D}_{X1,n}$, $\mathcal{D}_{X2,n}$

3.2 Characteristic Function: Model and Evaluation Metric

Throughout our experiments, we used four ML models from the scikit-learn library: logistic regression[1] (LR), SVM with linear kernel[2] (SVM-lin), SVM with RBF kernel[3] (SVM-rbf), and multi-layer perceptron[4] (MLP). We mainly focused on rather simple models, to reduce the runtime and to match the simplicity of the datasets we used. We report the choice of parameters for each model in Table 1.

Model	input normalized	parameters
LR	yes	penalty='l2'
SVM-lin	yes	penalty='l2', loss='squared_hinge'
SVM-rbf	no	kernel='rbf', gamma=0.7
MLP	yes	hidden_layer_sizes=(10, 10), activation='relu', solver='lbfgs'

Table 1: Choice of parameters for each of the four models

For the evaluation metric of each of the models, we follow Example 6 and compute the mean accuracy of the trained model on a test dataset sampled from the same distribution as the train dataset. Since in our case we are not limited by the amount of available labeled data, we chose a significantly larger test dataset than train dataset, such that the mean accuracy is closer to the real model performance.

3.3 Shapley Value Computation

3.3.1 Computing Valuation

The core part of computing one Shapley value is computing the valuation $v(S)$ of an exponential amount of subsets S of players. In the ML setting, computing one valuation $v(S)$ of a subset $S \subseteq \mathcal{D}$ corresponds to training the model M on S and evaluating the model M_S. Since this is the main bottleneck for the runtime, we want to avoid computing the same valuation multiple times. Hence we apply memoization by storing the valuations of subsets and reusing their values.

Algorithm 1 Computing Valuation

```
function VALUATE(S, withMemo)
    if withMemo then                                  ▷ apply memoization
        if S not in memo then
            memo[S] ← v(S)
        return memo[S]
    else                                              ▷ skip memoization
        return v(S)
```

1. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
3. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
4. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

3.3.2 Computing Exact Shapley Values

Here we propose two algorithms to compute the Shapley value s_i of a player exactly, based on the two equivalent definitions of a Shapley value. Following Definition 3, we can iterate over all possible permutations $\pi \in \Pi(\mathcal{D})$, or following Definition 4, we can iterate over all possible subsets $S \subseteq \mathcal{D} \setminus \{i\}$.

Algorithm 2 Computing Shapley Values Exactly (based on permutations)

function ComputeShapleyValues1($\mathcal{D}$, v, $withMemo$)
  $s[i] \leftarrow 0$ for all $i \in \mathcal{D}$ ▷ initialize Shapley values
  for $\pi \in \Pi(\mathcal{D})$ **do** ▷ iterate over all permutations
    for i in $\pi(\mathcal{D})$ **do**
      $s[i] \leftarrow s[i] + (\text{Valuate}(\pi_{\leq i}, withMemo) - \text{Valuate}(\pi_{<i}, withMemo))$
  $s \leftarrow \frac{1}{n!}s$
  return s

Algorithm 3 Computing Shapley Values exactly (based on subsets)

function ComputeShapleyValues2($\mathcal{D}$, v, $withMemo$)
  $s[i] \leftarrow 0$ for all $i \in \mathcal{D}$ ▷ initialize Shapley values
  for $S \in 2^{\mathcal{D}} \setminus \{\mathcal{D}\}$ **do** ▷ iterate over all subsets of size 0 to $|\mathcal{D}| - 1$
    for i in $\mathcal{D} \setminus S$ **do**
      $s[i] \leftarrow s[i] + \frac{1}{n\binom{n-1}{|S|}}(\text{Valuate}(S \cup \{i\}, withMemo) - \text{Valuate}(S, withMemo))$
  return s

3.3.3 Approximating Shapley Values

Computing the exact Shapley values can be computationally very demanding. In particular, training and evaluating an ML model an exponential number of times is not feasible for a large dataset. In view of this, we used a Monte-Carlo simulation-based approximation scheme for the Shapley values from Castro et al. (2009). The key idea is to sample m permutations from $\Pi(\mathcal{D})$ instead of calculating for all possible π.

Algorithm 4 Approximating Shapley Values (using m permutations)

function ApproximateShapleyValues($\mathcal{D}$, v, m)
  $s[i] \leftarrow 0$ for all $i \in \mathcal{D}$ ▷ initialize Shapley values
  repeat m times ▷ iterate over m randomly sampled permutations
    $\pi \leftarrow$ random sample from $\Pi(\mathcal{D})$
    for i in $\pi(\mathcal{D})$ **do**
      $s[i] \leftarrow s[i] + (\text{Valuate}(\pi_{\leq i}, False) - \text{Valuate}(\pi_{<i}), False)$
  $s \leftarrow \frac{1}{m}s$
  return s

3.3.4 Runtime Analysis

In Table 2, we summarize the runtime of each of the three algorithms presented in the previous part.

Model	Memo	Runtime
ComputeShapleyValues1	no	$\Theta(n \cdot n!)$
	yes	$\Theta(2^n)$
ComputeShapleyValues2	no	$\Theta(n \cdot 2^n)$
	yes	$\Theta(2^n)$
ApproximateShapleyValues	no	$\Theta(n \cdot m)$

Table 2: Runtime analysis of three different methods to compute Shapley values

4. Results

4.1 Robustness of Shapley Values

In this section, we present empirical results about the dependence of Shapley values on different components of the cooperative game formulation. For this analysis, we mainly focus on four characteristics of the Shapley values: the sign and magnitude of each Shapley value, the ranking of the data points according to their Shapley values, and the distribution of Shapley values. Using these characteristics, we attempt to gain insights about what kind of information Shapley values are able to capture from data points.

4.1.1 Dependence on Model

We first investigate the changes in the Shapley values when using different models. In particular, we want to analyze, whether the payoff assigned to data points are specific to a model or whether it reflects some notion of universal importance. In our experiments we compute the Shapley values for the five datasets $\mathcal{D}_{G1,200}, \mathcal{D}_{G2,200}, \mathcal{D}_{G3,200}, \mathcal{D}_{X1,200}, \mathcal{D}_{X2,200}$ using the four models LR, SVM-lin, SVM-rbf and MLP and a test dataset of size 2000.

Based on our results, we make the following conclusions: Assuming that the used models are expressive enough to learn patterns from the dataset, which is not the case for applying linear classifiers to an XOR dataset (see Figure 7 and Figure 8), Shapley values are able to capture some weak notion of importance of a data point for the overall model performance. To be precise, changing the model influences the four mentioned characteristics of Shapley values in the following way (see Section A.1):

- The sign of a Shapley value is mostly preserved. This indicates that Shapley values are able to capture the tendency of data points to improve or degrade the general model performance. Intuitively, this captures the notion that the optimal decision boundary is independent of the used model and data points on the wrong side of the optimal decision boundary (e.g. noise, mislabeled data, etc.) should not improve the performance of any model in general.

- The magnitude of a Shapley value is mostly *not* preserved. This indicates that Shapley values are not able to capture the absolute (i.e. model-independent) importance of data points, which would be the strongest possible notion of importance.

- The ranking of the data points according to their Shapley values is *partially* preserved. This indicates that Shapley values are able to capture some weak notion of relative importance. This means that it is more likely that data points with large Shapley values for some model A also have larger Shapley values for some other model B.

- The distribution of the Shapley values is similar for similarly expressive models (e.g. LR and SVM-lin, SVM-rbf and MLP). This indicates that Shapley values must be able to capture some hidden structure between the used model and dataset, which can be used later for retrieving useful information about the model or training (see Section 4.2).

To summarize, the Shapley values can capture some importance of data points, but the sign, distribution, or relative ranking matters more than the exact numerical values.

4.1.2 Dependence on Evaluation Metric

Here, we compare the Shapley values when the model is evaluated with the mean test accuracy versus the mean train accuracy. Similar to the previous part, we analyze the characteristics of the Shapley values (see Section A.2):

- The magnitude of a Shapley value is generally preserved. Intuitively, this means that the importance of a data point perceived by the model (i.e. Shapley value based on train accuracy) is similar to the actual importance of this data point (i.e. Shapley value based on test accuracy). However, larger deviations are observed when overfitting occurs (see Section 4.2).

- Given that the magnitudes of Shapley values are mostly preserved, this already implies that the signs, rankings and distributions are also mostly preserved.

4.1.3 Dependence on Other Data Points

In this part, we investigate the dependence of Shapley values of a fixed data point on the existence of other data points in the same dataset. In particular, we tracked the change of the Shapley values of a fixed dataset after removing 1%, 5% and 10% of the data points with the highest or lowest Shapley values. Using the same analysis as before, we make the following observations (see Section A.3):

- The sign of a Shapley value is mostly preserved.

- The magnitude of a Shapley value is *not* preserved.

- The ranking of the Shapley values is *not* preserved. This allows us to conclude that the Shapley values are dependent on the other data points in the dataset. When a data point with a high Shapley value is removed, that Shapley value is "redistributed" among the other points in a non-uniform manner. This means that a previously less

important data point can rise to become a more important data point. For example, points that are very close to the most important data point might be overlooked when the more significant point exists in the dataset, but one of these points might become more significant when this most important point is removed.

- The distribution of Shapley values is mostly preserved.

From these results, we conclude that Shapley values are "relative" to the Shapley values of other data points in the given dataset in the sense that they depend on each other. Hence, the computed Shapley values are mostly with respect to a specific dataset and can vary when changing parts of the dataset.

4.2 Information Retrieval from Shapley Values

In this section, we investigate if we can utilize the computed Shapley values to obtain useful information about the model and the training process. We establish a few hypotheses associating Shapley values with certain properties of our model and dataset and verify them using the results of our baseline experiments.

4.2.1 Detecting Suitability of Model for Dataset

We claim that using Shapley values, one can measure the suitability of a model for a given dataset to a certain extent, including tendency of over- and underfitting. We determined that the main characteristic of an overfitting model is a more heavy-tailed distribution of Shapley values (see Shapley values of MLP in Section A.1). This means that the model performance is mostly determined by a small number of "very helpful" data points and "very bad" data points, while most of the other data points are relatively less important in training. For non-overfitting models, the range of Shapley values is significantly smaller.

Another useful feature to extract is the deviation between the Shapley values computed based on the test accuracy and based on the train accuracy. The advantage of using Shapley values as an indicator for overfitting is that it also provides information about which data point causes the model to overfit, which might help in understanding the performance of the model and its learning procedure (see Shapley values of MLP in Figure 9).

4.2.2 Detecting Mislabeled Data

Real-life data is often noisy (Frenay and Verleysen, 2014) or corrupted (Steinhardt et al., 2017), and it is possible that there is mislabeled data in the training dataset. Previous works on data valuation with Shapley values (Ghorbani and Zou, 2019; Tang et al., 2021) showed that Shapley values are able to discern between mislabeled data and correctly labeled data better than other metrics. That is, the data points with lowest Shapley values are more likely to have been mislabeled than the points that are least "significant" in other metrics. This suggests that we can attempt to eliminate mislabeled data by removing data points with low Shapley values.

However, this approach requires the prior knowledge of the proportion of mislabeled data. Otherwise, there is no way of knowing how many data points to remove; we may even end up removing data points that were correctly labeled, thereby potentially losing information about the dataset. Therefore, we investigate how the distribution of the Shapley values changes with different fractions of mislabeled data. We run the analysis on twelve datasets ($\mathcal{D}_{G1,100}$, $\mathcal{D}_{G1,200}$, $\mathcal{D}_{G2,100}$, $\mathcal{D}_{G2,200}$, $\mathcal{D}_{G3,100}$, $\mathcal{D}_{G3,200}$, $\mathcal{D}_{U1,100}$, $\mathcal{D}_{U1,200}$, $\mathcal{D}_{U2,100}$, $\mathcal{D}_{U2,200}$, $\mathcal{D}_{U3,100}$, $\mathcal{D}_{U3,200}$) with three models (LR, SVM-lin, SVM-rbf) where each data point is independently mislabeled with a certain probability (0%, 5%, 10%, 15%, 20%).

We report the following trends in the probability density function with increasing fraction of mislabeled data (see Section B for details):

- The peak of the probability density function is lower
- The variance is higher
- The fraction of data points with negative Shapley values increases

Any of these features in the probability density function can potentially be used as an indicator of presence of mislabeled data in the dataset. For example, the following plot in Figure 4 shows a box plot of the fraction of data points with negative Shapley values when each of the data points were mislabeled with probability 0%, 5%, 10%, 15%, 20%. Given an arbitrary dataset, we can examine the fraction of data points with negative Shapley values, and use this value to estimate the amount of mislabeled data. If we use multiple features of the dataset, the accuracy of this estimation is expected to increase.

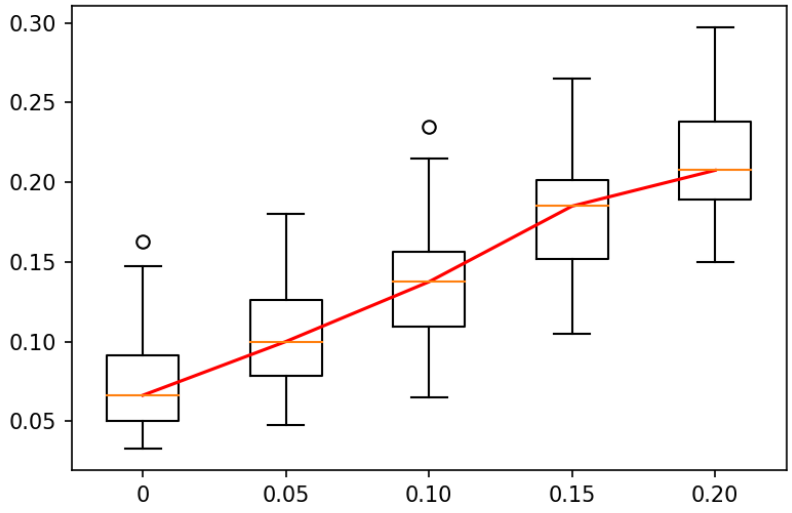

Figure 4: Fraction of data points with negative Shapley values when each point was independently mislabeled with probability 0%, 5%, 10%, 15%, 20% (left to right)

5. Conclusion and Future Work

In this paper, we empirically test whether the sign, magnitude, the relative ordering, and the distribution of the Shapley values are invariant to changes in the formulation of the cooperative game in the ML setting. We notice that some of these characteristics are dependent on the specific dataset, model, or the evaluation metric. We then explore ways to use the Shapley values to extract information about the dataset and the model. We suggest that they can be used to detect underfitting/overfitting or the presence of mislabeled data.

In future works, we wish to extend our analaysis to real-word benchmark datasets, more complicated models based on neural networks, and different evaluation metrics like the F1-score. We are also interested in whether the performance or the runtime of an ML model can be improved if the dataset is presented in decreasing order of the estimated Shapley values.

References

Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009. ISSN 0305-0548. URL https://www.sciencedirect.com/science/article/pii/S0305054808000804.

Benoit Frenay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 2014. doi: 10.1109/TNNLS.2013.2292894.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2242–2251. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/ghorbani19c.pdf.

Benedek Rozemberczki, Lauren Watson, Péter Bayer, Hao-Tsung Yang, Olivér Kiss, Sebastian Nilsson, and Rik Sarkar. The shapley value in machine learning, 2022. URL https://arxiv.org/abs/2202.05594.

L. S. Shapley. *17. A Value for n-Person Games*, pages 307–318. Princeton University Press, 1953. URL https://doi.org/10.1515/9781400881970-018.

Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3520–3532, 2017.

S. Tang, A. Ghorbani, R. Yamashita, et al. Data valuation for medical imaging using shapley value and application to a large-scale chest x-ray dataset. *Scientific Reports*, 11 (8366), 2021. URL https://www.nature.com/articles/s41598-021-87762-2.

Appendix A. Plots from Section 4.1

In this section, we present plots omitted from Section 4.1. For each cooperative game formulation, we present three plots:

1. The first plot shows the decision boundary of the model and the full dataset, where the size of each data point corresponds to its Shapley value. The size of each data point is computed with

$$p_i = \sqrt{\text{clip}\left(10\frac{s_i}{s_{avg}}, 1, 100\right)} \text{ px} \quad \text{with } s_{avg} = \frac{1}{n}\sum_{i=1}^{n} s_i$$

2. The second plot presents the Shapley values in a bar chart, where the data points are sorted according to the same ordering across the row (typically in decreasing order of the Shapley values of the first plot in the row) for better comparability between individual data points. The dashed horizontal line shows the average Shapley value.

3. The third plot presents the Shapley values in a bar chart, where the data points of each plot are sorted in decreasing order of the Shapley values of that plot for better comparability of the distribution of Shapley values. The dashed horizontal line shows the average Shapley value.

A.1 Dependence on Model

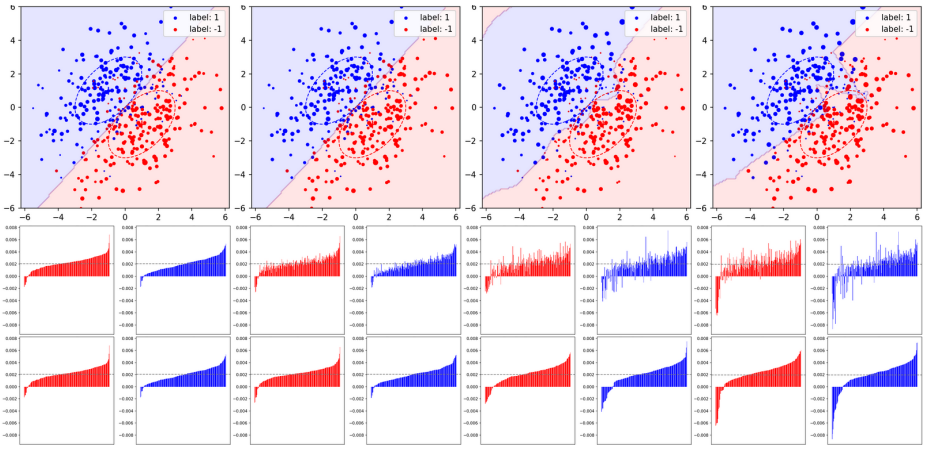

Figure 5: Shapley values of $\mathcal{D}_{G1,200}$ for LR, SVM-lin, SVM-rbf, MLP (left to right)

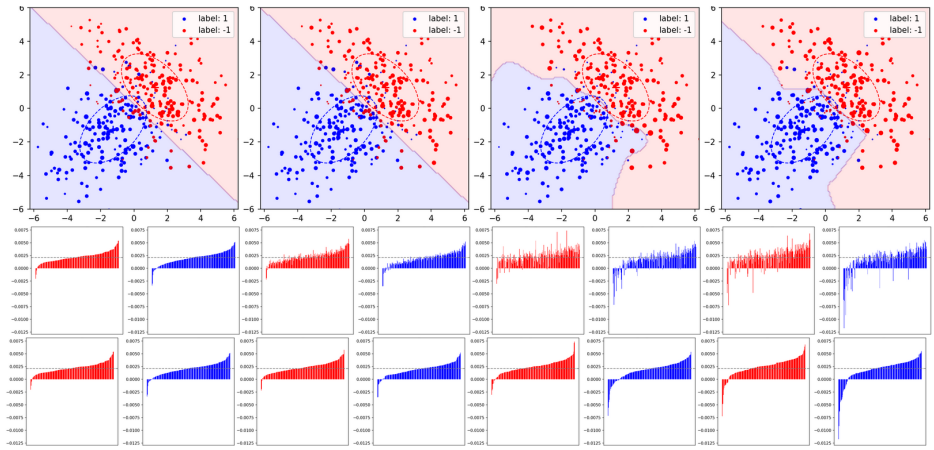

Figure 6: Shapley values of $\mathcal{D}_{G3,200}$ for LR, SVM-lin, SVM-rbf, MLP (left to right)

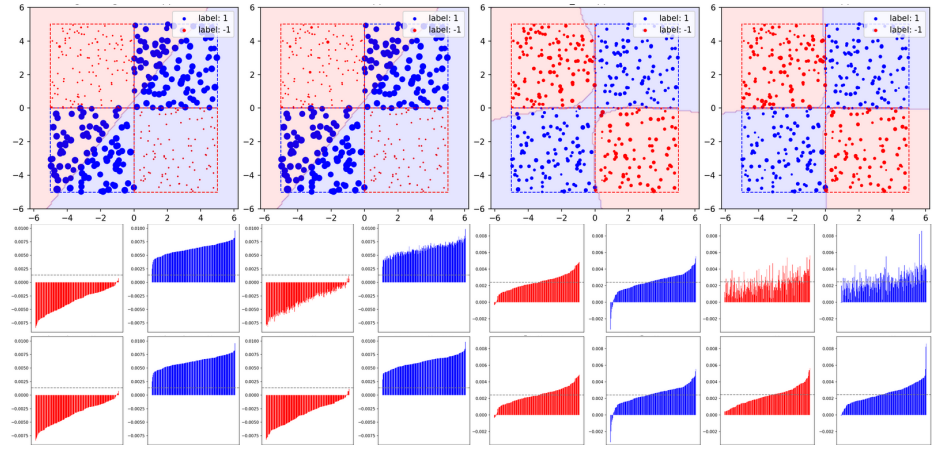

Figure 7: Shapley values of $\mathcal{D}_{X1,200}$ for LR, SVM-lin, SVM-rbf, MLP (left to right)

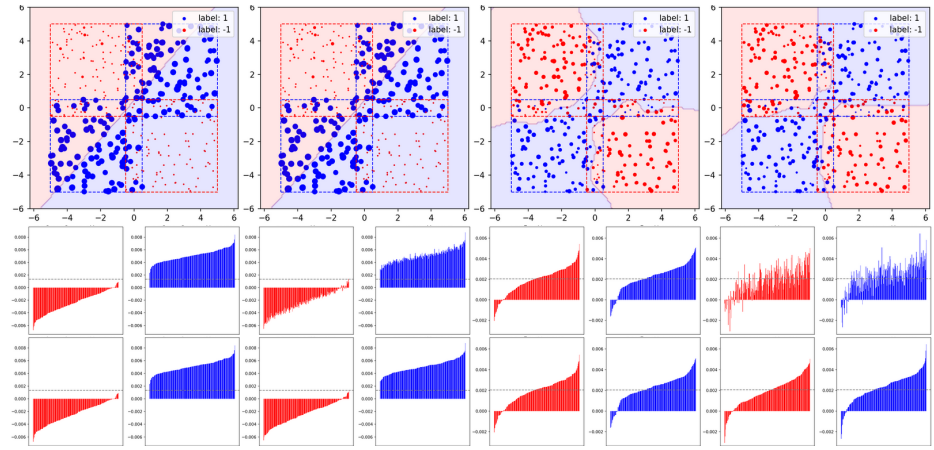

Figure 8: Shapley values of $\mathcal{D}_{X2,200}$ for LR, SVM-lin, SVM-rbf, MLP (left to right)

A.2 Dependence on Evaluation Metric

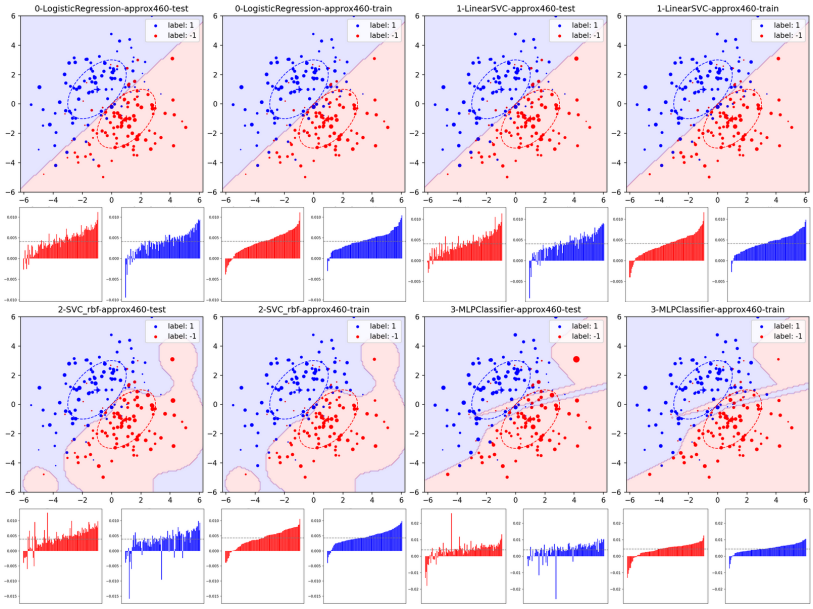

Figure 9: Shapley values of $\mathcal{D}_{G1,100}$ based on test accuracy (left) and train accuracy (right)

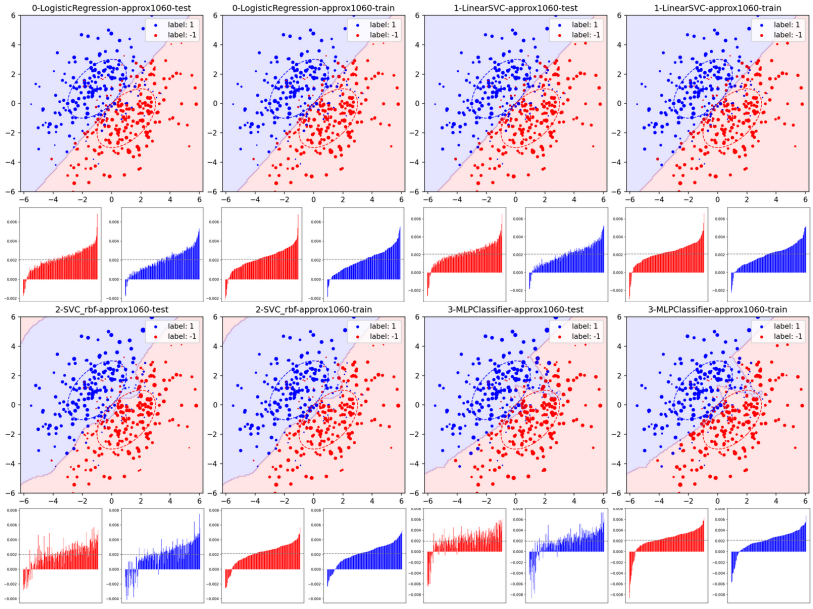

Figure 10: Shapley values of $\mathcal{D}_{G1,200}$ based on test accuracy (left) and train accuracy (right)

A.3 Dependence on other Data Points

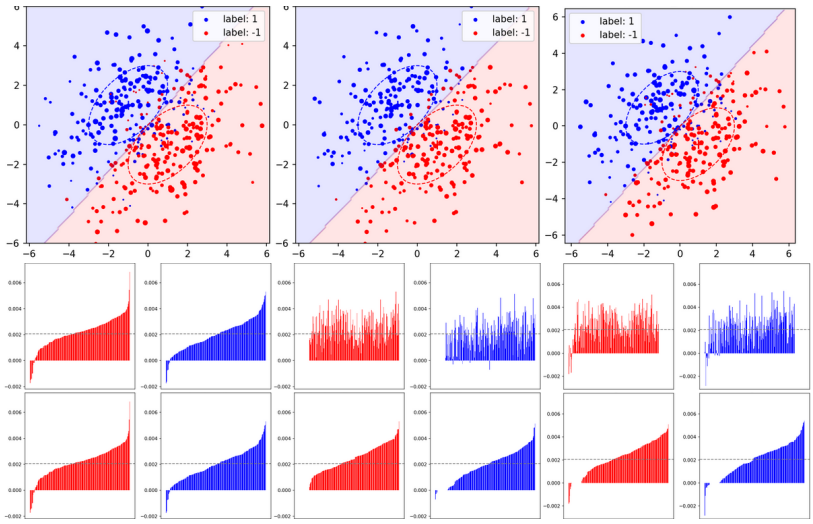

Figure 11: Shapley values of $\mathcal{D}_{G1,200}$ for LR on full dataset (left), without bottom 10% (middle), without top 10% (right)

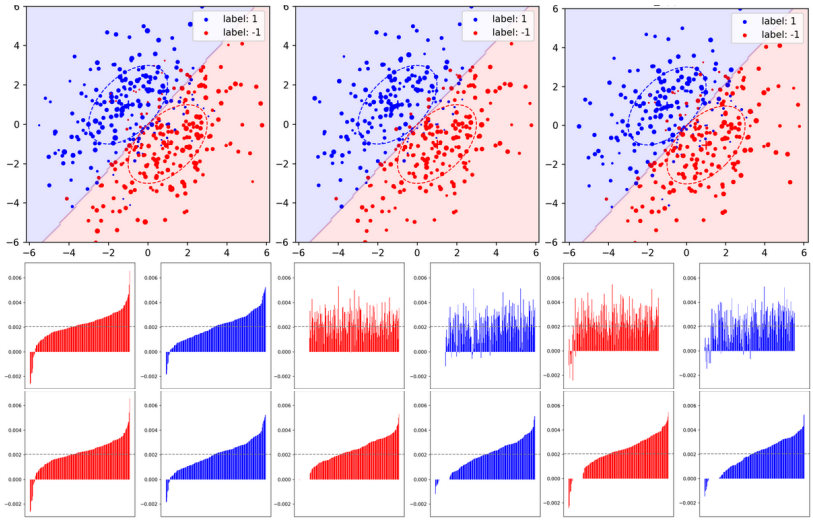

Figure 12: Shapley values of $\mathcal{D}_{G1,200}$ for SVM-lin on full dataset (left), without bottom 10% (middle), without top 10% (right)

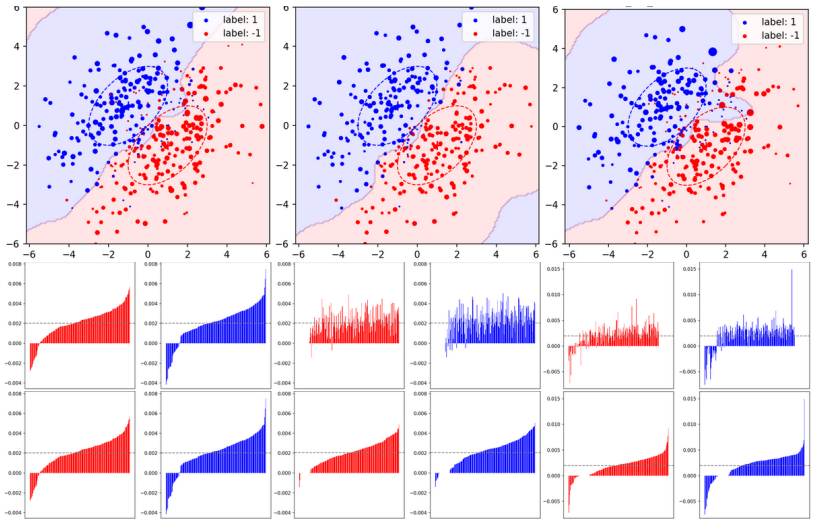

Figure 13: Shapley values of $\mathcal{D}_{G1,200}$ for SVM-rbf on full dataset (left), without bottom 10% (middle), without top 10% (right)

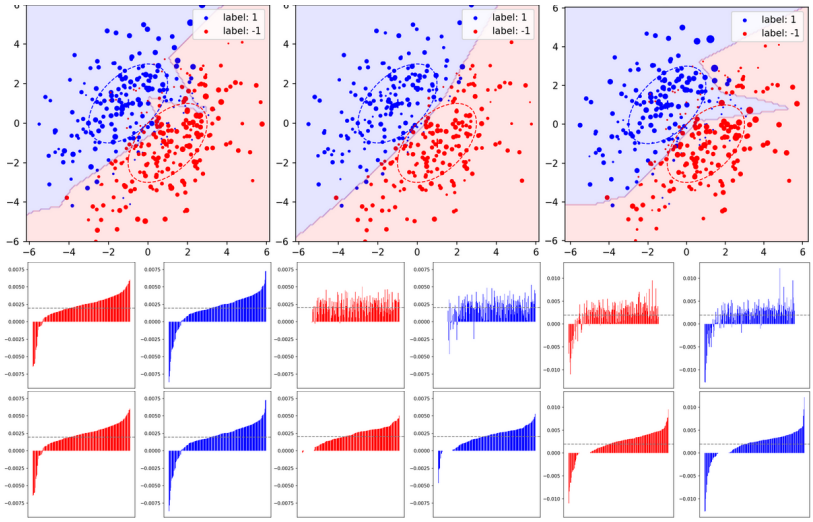

Figure 14: Shapley values of $\mathcal{D}_{G1,200}$ for MLP on full dataset (left), without bottom 10% (middle), without top 10% (right)

Appendix B. Plots from Section 4.2

Here we include the plots that were omitted from Section 4.2. The plots show the distribution of Shapley values of a dataset when each data point is mislabeled with probability 0% (black), 5% (red), 10% (green), 15% (blue), and 20% (purple).

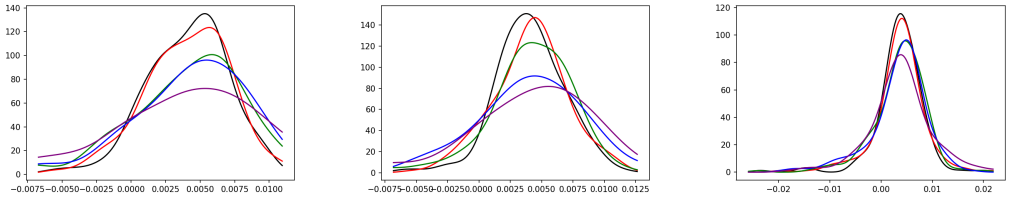

Figure 15: Distribution of Shapley values of $\mathcal{D}_{G1,100}$ with data points mislabeled with probability 0%, 5%, 10%, 15%, 20% for LR, SVM-lin, SVM-rbf (left to right)

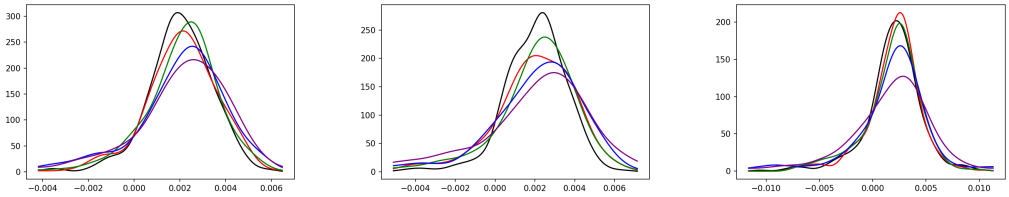

Figure 16: Distribution of Shapley values of $\mathcal{D}_{G1,200}$ with data points mislabeled with probability 0%, 5%, 10%, 15%, 20% for LR, SVM-lin, SVM-rbf (left to right)

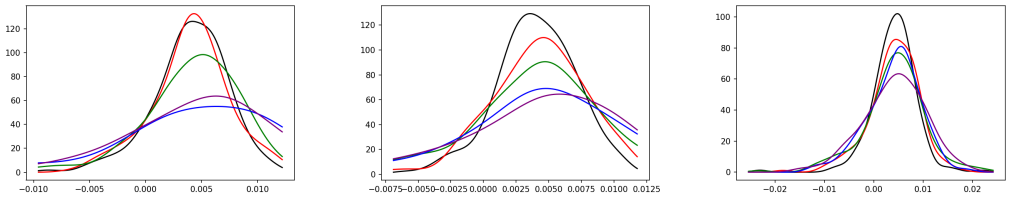

Figure 17: Distribution of Shapley values of $\mathcal{D}_{G2,100}$ with data points mislabeled with probability 0%, 5%, 10%, 15%, 20% for LR, SVM-lin, SVM-rbf (left to right)

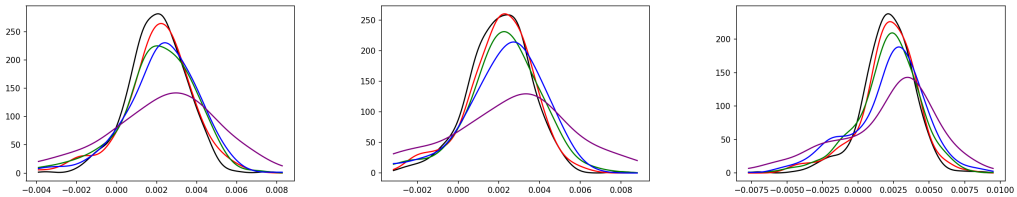

Figure 18: Distribution of Shapley values of $\mathcal{D}_{G2,200}$ with data points mislabeled with probability 0%, 5%, 10%, 15%, 20% for LR, SVM-lin, SVM-rbf (left to right)

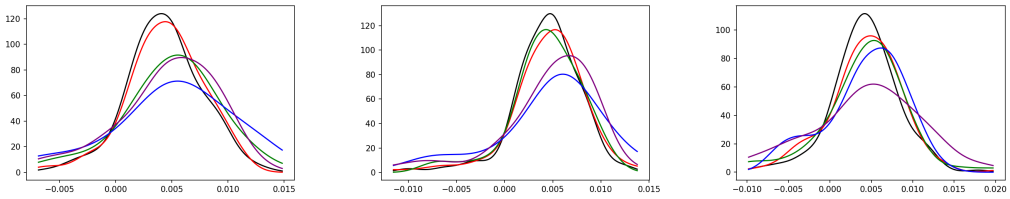

Figure 19: Distribution of Shapley values of $\mathcal{D}_{G3,100}$ with data points mislabeled with probability 0%, 5%, 10%, 15%, 20% for LR, SVM-lin, SVM-rbf (left to right)

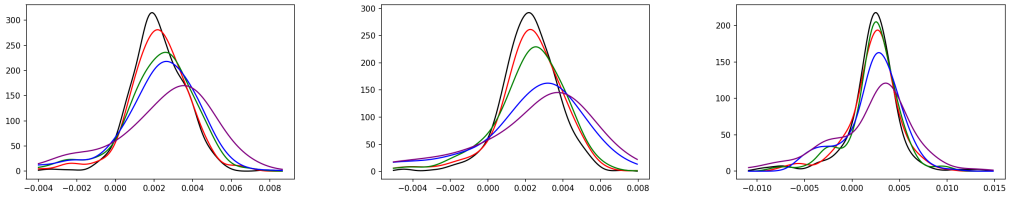

Figure 20: Distribution of Shapley values of $\mathcal{D}_{G3,200}$ with data points mislabeled with probability 0%, 5%, 10%, 15%, 20% for LR, SVM-lin, SVM-rbf (left to right)

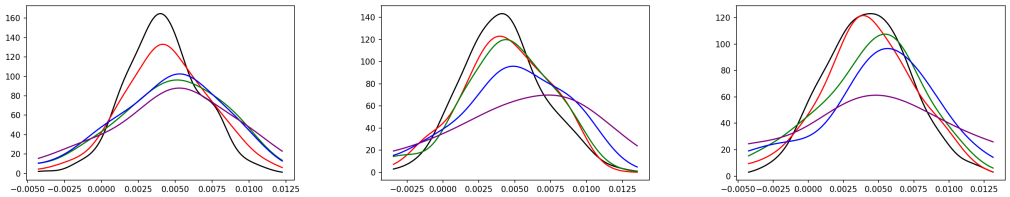

Figure 21: Distribution of Shapley values of $\mathcal{D}_{U1,100}$ with data points mislabeled with probability 0%, 5%, 10%, 15%, 20% for LR, SVM-lin, SVM-rbf (left to right)

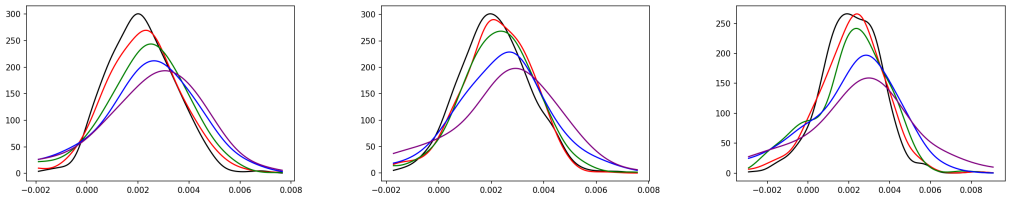

Figure 22: Distribution of Shapley values of $\mathcal{D}_{U1,200}$ with data points mislabeled with probability 0%, 5%, 10%, 15%, 20% for LR, SVM-lin, SVM-rbf (left to right)

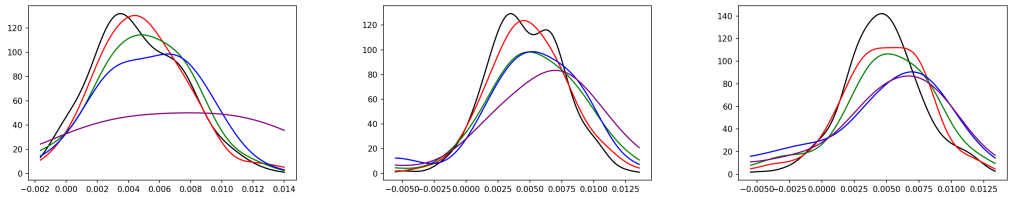

Figure 23: Distribution of Shapley values of $\mathcal{D}_{U2,100}$ with data points mislabeled with probability 0%, 5%, 10%, 15%, 20% for LR, SVM-lin, SVM-rbf (left to right)

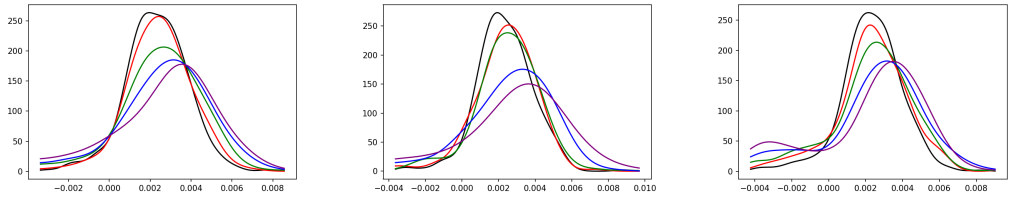

Figure 24: Distribution of Shapley values of $\mathcal{D}_{U2,200}$ with data points mislabeled with probability 0%, 5%, 10%, 15%, 20% for LR, SVM-lin, SVM-rbf (left to right)

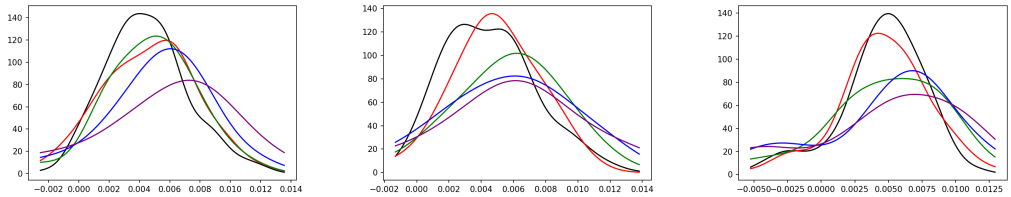

Figure 25: Distribution of Shapley values of $\mathcal{D}_{U3,100}$ with data points mislabeled with probability 0%, 5%, 10%, 15%, 20% for LR, SVM-lin, SVM-rbf (left to right)

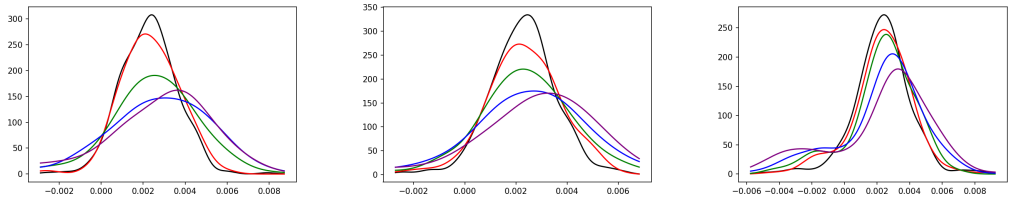

Figure 26: Distribution of Shapley values of $\mathcal{D}_{U3,200}$ with data points mislabeled with probability 0%, 5%, 10%, 15%, 20% for LR, SVM-lin, SVM-rbf (left to right)