Efficient Banzhaf-Based Data Valuation for k-Nearest Neighbors Classification

Guangyi Zhang*
Shenzhen Technology University
Shenzhen, China
zhangguangyi@sztu.edu.cn

Lutz Oettershagen
University of Liverpool
Liverpool, United Kingdom
lutz.oettershagen@liverpool.ac.uk

Lixu Wang
Nanyang Technological University
Singapore
wanglixu4334@gmail.com

Aristides Gionis
KTH Royal Institute of Technology
Digital Futures
Stockholm, Sweden
argioni@kth.se

ABSTRACT

Data valuation, the task of quantifying the contribution of individual data points to model performance, has emerged as a fundamental challenge in machine learning. Game-theoretic approaches, such as the Banzhaf value, offer principled frameworks for fair data valuation; however, they suffer from exponential computational complexity. We address this challenge by developing efficient algorithms specifically tailored for computing Banzhaf values in k-nearest neighbor (kNN) classifiers. We first establish the theoretical hardness of the problem by proving that it is #P-hard. Despite this intractability, we exploit the locality properties of kNN classifiers to develop practical exact algorithms. Our main contribution is a dynamic programming framework that achieves significant computational improvements: we present a pseudo-polynomial algorithm with $O(Wkn^2)$ time complexity for weighted kNN classifiers, where W is the maximum sum of top-k weights, and a specialized algorithm for unweighted kNN that achieves $O(nk^2)$ time complexity, that is, linear in the number of data points. We also offer efficient Monte Carlo estimation methods. Extensive experiments on real-world datasets demonstrate the practical efficiency of our approach and its effectiveness in data valuation applications.

PVLDB Reference Format:
Guangyi Zhang, Lutz Oettershagen, Lixu Wang, and Aristides Gionis. Efficient Banzhaf-Based Data Valuation for k-Nearest Neighbors Classification. PVLDB, 19(9): 1880 - 1892, 2026.
doi:10.14778/3819518.3819521

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Guangyi-Zhang/knn-banzhaf-code.

*Corresponding author.

Proceedings of the VLDB Endowment, Vol. 19, No. 9 ISSN 2150-8097.
doi:10.14778/3819518.3819521

1 INTRODUCTION

Data valuation has emerged as a fundamental problem in machine learning, addressing the critical question of how to quantify the contribution of individual data points to model performance [10, 17, 34]. As machine learning systems increasingly rely on large datasets assembled from diverse sources, understanding the relative importance of different data points becomes essential for various applications, such as enabling practitioners to make informed decisions regarding data acquisition, identifying low-quality or redundant data, and ensuring fair compensation for data providers in collaborative-learning scenarios.

Cooperative game theory [7] provides a principled framework for addressing data-valuation tasks. By modeling the data valuation problem as a cooperative game where data points are players and the model performance represents the value function, we can leverage established solution concepts from cooperative game theory to assign fair and meaningful values to individual data points. Among the various solution concepts, the Shapley value [33] and the Banzhaf value [2] have gained particular attention due to their desirable axiomatic properties. In particular, the Banzhaf value has been adopted as a measure of voting power in the analysis of voting in the Council of the European Union [38].

Both values are based on the idea of quantifying the *marginal contribution* of a particular data point to the performance of the model. Arguably, the simplest way that leverages the same idea to quantify the value of a data point z in a dataset D is the *leave-one-out* (LOO) value, which measures the difference in the model performance before and after the data point is removed from the dataset, that is, $\mathrm{LOO}(z) = v(D) - v(D \setminus \{z\})$, where $v : 2^N \to \mathbb{R}$ is the value function that represents the model performance, for example, the accuracy of a model. However, the LOO value is not informative in the presence of data redundancy [23]. Besides, the impact of a single data point given the remaining large amount of data is often negligible [3, 13].

The Shapley and Banzhaf values further develop the idea by aggregating the marginal contributions of a data point to all possible subsets of the dataset. For example, the Banzhaf value of a data point z is defined precisely as the average marginal contribution of

z to all possible subsets $S \subseteq D \setminus \{z\}$ of the dataset, that is,

$$\beta(z) = \frac{1}{2^{n-1}} \sum_{S \subseteq D \setminus \{z\}} [v(S \cup \{z\}) - v(S)].$$

The Shapley value is also defined similarly, but with a different weighting scheme over those subsets. Therefore, these two values give a more comprehensive view of the contribution of a data point to the model performance.

Critically, the Shapley value has been mathematically proved to be the *unique solution* that satisfies the axioms of *efficiency*, *symmetry*, *null player*, and *additivity* [33], while the Banzhaf value satisfies the same axioms except efficiency. These axioms are valuable properties that make both Shapley and Banzhaf values natural choices for fair and reliable data valuation. Among the axioms, efficiency requires that the value of the entire dataset is equal to the sum of the values of all data points, i.e.,

$$v(D) - v(\emptyset) = \sum_{z \in D} \phi(z),$$

where $\phi(z)$ is the Shapley value of data point z. The raw Banzhaf values do not satisfy the efficiency axiom, but this can be remedied by normalizing the raw Banzhaf values, a common post-processing step used in practice [36]. It is worth mentioning that though intuitively appealing, efficiency is not essential in many ranking-based applications, including dataset curation and data cleaning, where the focus is on the relative order of data-point values rather than their exact magnitudes. For these applications, Banzhaf can be a practical choice: it often yields sparser values and can be less affected by duplicate or noisy points. More broadly, the Banzhaf values can be more robust in certain scenarios [2, 19, 24, 39].

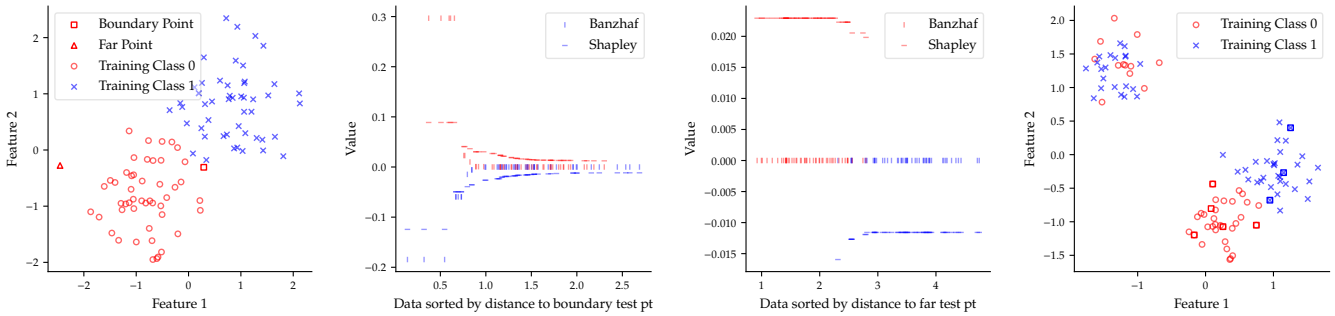
We illustrate the major differences between different values in Fig. 1. The values of all data points in a 2D dataset (Fig. 1a) are computed with respect to a boundary test point (square) and a test point that is far from the boundary (triangle). Only three training data points have non-zero LOO values. After duplicating the dataset by creating a copy of each original data instance, all LOO values become zero. In contrast, most non-zero game-theoretic values remain non-zero and informative after duplication. This simple observation suggests that LOO values are less informative, especially in the presence of data redundancy [3, 13, 23].

Next, we compare the Shapley and Banzhaf values to illustrate why they can behave differently and be preferable in different applications. There are three key observations in Fig. 1. First, the Banzhaf values are *sparser* than the Shapley values, while the Shapley values remain far from zero even for data points that are very far from the test point (Fig. 1b). This sparsity property can be very helpful in some applications, such as exemplar-based interpretability [31]. Second, for a test point (triangle) that is surrounded by a majority of data points of its own class, the Banzhaf values of these neighboring points are close to zero (Fig. 1c). Intuitively, this means that removing any of these neighboring points has little impact on the prediction of the test point. Thus, the Banzhaf values can not only highlight more concentrated relative importance of the data points to a boundary point, but also indicate whether the test point has been well-supported, and there is no need for further data enhancement. Third, the Banzhaf values are more robust to the noise in the dataset (Fig. 1d), where additional random-label training data points are injected (upper-left corner). The Banzhaf values of the noise are all zero, and the values of the original data points remain unchanged. In contrast, the total Shapley value of the original data points drops by over 25% due to the non-zero Shapley values of the noise data. The fragility of the Shapley values originates from its overallocation of values to extreme subsets; see definition in Eq. (3). Robustness is crucial in adversarial settings where malicious parties may inject useless or mislabeled data to dilute the value of legitimate contributors and siphon off monetary rewards in data marketplaces, or in environments with noisy data.

Despite the theoretical appeal of these game-theoretic approaches to data valuation, their practical application faces a significant obstacle: computational complexity. Computing exact Shapley or Banzhaf values requires evaluating the value function over all possible subsets of the dataset, leading to exponential complexity. In general, it has been proved that both Shapley and Banzhaf values are #P-hard to compute [6]. This computational burden becomes prohibitive for real-world datasets, which often contain thousands or millions of samples. Consequently, most existing work has focused on approximation methods, such as sampling-based approaches [1, 4, 8, 26, 28, 45], which trade accuracy for efficiency.

In this paper, we address the challenge by developing fast dynamic programming (DP) algorithms specifically tailored for computing Banzhaf values in k-nearest neighbor (kNN) *classifiers*. The classic kNN classifiers remain one of the most central and widely-used models in machine learning, especially when combined with recent pre-trained embedding techniques. Our approach exploits the locality property of kNN and computes exact Banzhaf values efficiently without resorting to approximation. Our algorithmic contributions target kNN classification with *hard-label* valuation (see definition in Section 2.2). We include soft-label and Shapley variants in experiments for comparison. We offer the fastest algorithms for hard-label Banzhaf valuation, which are even more efficient than their hard-label Shapley counterparts. Our contribution bridges the gap between theoretical foundations of cooperative game theory and practical demands of real-world machine-learning applications. In detail, this paper makes the following contributions:

- We prove that it is #P-hard to compute the Banzhaf values for the kNN classification model.
- We present the first dynamic programming algorithm for computing exact Banzhaf values in weighted kNN classifiers in pseudo-polynomial $O(Wn^4)$ time, where W is the maximum sum of top-k weights. Our algorithm enables exact computation of Banzhaf values without approximation.
- We then significantly improve the time complexity of the DP algorithm for computing exact Banzhaf values in weighted kNN classifiers to $O(Wkn^2)$, through novel algorithmic techniques including efficient partial sum computation and recurrence relations for binomial coefficients.
- We present a specialized efficient algorithm for unweighted kNN achieving $O(nk^2)$ time complexity. Thus, the running time of our algorithm is essentially *linear* in the number of training points, since k is typically a small constant in practice. To the best of our knowledge, this is the first near-linear time algorithm for computing the (hard-label) kNN-based game-theoretic values.



(a) Two selected test points (triangle and square) in a 2D dataset.

(b) The values of the data points with respect to a boundary test point.

(c) The values of the data points with respect to a test point that is far from the boundary.

(d) Injected random noise (upper-left corner) to the 2D dataset, with test points in square.

Figure 1: The difference between the Shapley and Banzhaf values. The model performance is measured by the accuracy of a kNN classifier with $k = 5$.

- We also provide efficient coalition-based and permutation-based sampling algorithms for computing approximate Banzhaf values, by exploiting the locality property of kNN classifiers.
- Our experimental evaluation demonstrates the practical efficiency of our algorithms on real-world datasets, and their effectiveness in various applications such as data acquisition and cleaning. In particular, the unweighted exact DP algorithm scales to much larger training sets, while the weighted exact DP algorithm is more efficient than Shapley-value baselines under the same hard-label setting.

We note that our framework does not directly extend to kNN regression. The standard regression utility introduces a normalization term that is not decomposable under our DP recurrence, and the continuous output eliminates the pivotal-subset counting structure used in hard-label classification. Therefore, regression is beyond the scope of this paper (see Appendix C.1.4 [44]for details).

The rest of the paper is organized as follows. We begin by introducing the necessary preliminaries in Section 2. Then, we present our dynamic programming and sampling algorithms in Section 3 and Section 4, respectively. The related work is discussed in Section 5 and our experimental evaluation is presented in Section 6. We conclude in Section 7.

2 PRELIMINARIES

We first define the Banzhaf and Shapley values in the general framework of cooperative game theory. Then, we introduce the Banzhaf values for the kNN classification model and we prove its hardness.

2.1 Banzhaf and Shapley values

The Banzhaf values are a fundamental concept in cooperative game theory. They provide a principled approach to distribute in a fair manner the total value generated by a coalition over its members. Originally introduced by Banzhaf III [2] in the context of voting power analysis, the idea has found widespread applications in various domains, including economics, political science, and more recently, machine learning for data-valuation problems.

To formally define the Banzhaf values, we begin with the standard framework of cooperative games. A cooperative game is characterized by a pair (N, v), where $N = \{1, 2, \ldots, n\}$ represents the set of players and $v : 2^N \to \mathbb{R}$ is a function that assigns a real value to each subset (coalition) $S \subseteq N$ of players. The value $v(S)$ represents the total value of the coalition S. By convention, we assume $v(\emptyset) = 0$, meaning that the empty coalition generates no value.

For any player $i \in N$ and coalition $S \subseteq N \setminus \{i\}$, the *marginal contribution* of player i to coalition S is defined as $v(S \cup \{i\}) - v(S)$. This quantity captures the additional value generated when player i joins an existing coalition S. The Banzhaf value of player i, denoted $\beta_i(v)$, is then defined as the expected marginal contribution of player i over all possible coalitions, that is,

$$\beta_i(v) = \frac{1}{2^{n-1}} \sum_{S \subseteq N \setminus \{i\}} [v(S \cup \{i\}) - v(S)]. \tag{1}$$

For comparison, we also define the Shapley value $\phi_i(v)$ for player i, which is defined as

$$\phi_i(v) = \frac{1}{n!} \sum_{\pi \in \Pi(N)} [v(\pi_i \cup \{i\}) - v(\pi_i)], \tag{2}$$

where $\Pi(N)$ denotes the set of all permutations of N, and π_i denotes the set of players that precede player i in π [33]. Notably, the Shapley value can be rewritten as the weighted average over all possible coalitions, as follows:

$$\phi_i(v) = \frac{1}{n} \sum_{S \subseteq N \setminus \{i\}} \binom{n-1}{|S|}^{-1} [v(S \cup \{i\}) - v(S)]. \tag{3}$$

Unlike the Banzhaf value that treats all coalitions equally, the Shapley value puts more emphasis on the marginal contribution of a player to extremely small and large coalitions, while underemphasizing those of medium sizes.

2.2 Banzhaf value for kNN classifier

In the context of machine learning, we adapt the Banzhaf value framework to quantify the contribution of individual training data points to model performance. Consider a training dataset D =

$\{z_1, \ldots, z_n\}$ where each data point $z_i = (x_i, y_i)$ represents a feature-label pair with $y_i \in \mathcal{Y}$, that is, $\mathcal{Y}$ is the set of labels. For a given test data point $z_{\text{test}} = (x_{\text{test}}, y_{\text{test}})$, we define the players in our cooperative game as the training data points. We specifically focus on the kNN classifier setting.

The valuation function v for the kNN classifier is defined as follows. For any subset $S \subseteq D$ of training data points and a fixed test point z_{test}, the valuation function $v(S \mid z_{\text{test}})$ represents the accuracy of the kNN classifier trained on the subset S when predicting the label of the test point, which is

$$v(S \mid z_{\text{test}}) = \begin{cases} 1 & \text{if } y_{\text{test}} = \arg\max_{y \in \mathcal{Y}} \sum_{i=1}^{\min\{|S|,k\}} w_i \, \mathbb{1}\{y_i(S) = y\} \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where $z_i(S)$ is the i-th closest point of S to x_{test}, and $w_i = w(z_i(S) \mid z_{\text{test}})$ is the weight of $z_i(S)$. Note that the valuation function is zero when $|S| = 0$. We also define it to be zero when the model predicts more than one label, which is possible because of ties. For the kNN classifier, the weight w_i is determined by a non-increasing function of the distance between $z_i(S)$ and x_{test} [11], or simply set to $w_i = 1$ for the unweighted kNN. Without loss of generality, we assume that the weights are integers.

In the case of binary classification, i.e., $|\mathcal{Y}| = 2$, we can further simplify the valuation function as follows. We focus on this case in the paper for simplicity, and discuss extension to the general case in Appendix C.1.3. Then, the valuation function is given by

$$v(S \mid z_{\text{test}}) = \begin{cases} 1 & \text{if } \sum_{i=1}^{\min\{|S|,k\}} w_i \, (-1)^{\mathbb{1}\{y_i(S) \neq y_{\text{test}}\}} > 0 \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $w_i = w(z_i \mid z_{\text{test}})$ and where the factor $(-1)^{\mathbb{1}\{y_i(S) \neq y_{\text{test}}\}}$ becomes -1 if $y_i \neq y_{\text{test}}$, or $+1$ otherwise. To keep the notation simple, we re-define $w(z_i \mid z_{\text{test}})$ as $w(z_i(S) \mid z_{\text{test}})(-1)^{\mathbb{1}\{y_i(S) \neq y_{\text{test}}\}}$ for binary classification in subsequent sections.

With the valuation function defined, the Banzhaf value of a training data point z_i is given by

$$\beta_i(v \mid z_{\text{test}}) = \frac{1}{2^{n-1}} \sum_{S \subseteq N \setminus \{i\}} [v(S \cup \{i\} \mid z_{\text{test}}) - v(S \mid z_{\text{test}})]. \tag{6}$$

We often omit the test point z_{test} in the notation for simplicity when the context is clear.

When provided with a test dataset D_{test} comprising n_{test} test data points, the final Banzhaf value of a training data point $z_i \in D$ is the average of the Banzhaf values over all test points:

$$\beta_i(v \mid D_{\text{test}}) = \frac{1}{n_{\text{test}}} \sum_{z_{\text{test}} \in D_{\text{test}}} \beta_i(v \mid z_{\text{test}}). \tag{7}$$

Remark 1. *The valuation function $v(S \mid z_{test})$ defined in Eq.* (4) *is also known as the* hard-label *valuation function. In contrast, there exists a* soft-label *counterpart [40] that is defined as follows.*

$$v(S \mid z_{test}) = \begin{cases} \frac{\sum_{i=1}^{\min\{|S|,k\}} w(z_i(S)|z_{test}) \, \mathbb{1}(y_i(S)=y_{test})}{\sum_{i=1}^{\min\{|S|,k\}} w(z_i(S)|z_{test})} & \text{if } |S| > 0 \\ 1/|\mathcal{Y}| & \text{if } |S| = 0. \end{cases} \tag{8}$$

The soft-label valuation function captures a more fine-grained notion of contribution, while the use of the hard-label valuation function identifies the most decisive players. We will demonstrate in the experiments that both versions are useful in different scenarios.

2.3 Hardness of computing Banzhaf values for kNN classifier

We now prove that computing the Banzhaf values for the kNN classifier, as defined in the previous section, is a #P-hard problem. We prove this hardness result via a reduction from the Weighted-Majority-Game (WMG) problem [29], a classical #P-hard problem in cooperative game theory. Our hardness result suggests that, under standard complexity assumptions, there is unlikely to exist a (strongly) polynomial-time algorithm for computing Banzhaf values for kNN classifier.

Theorem 1. *Given a training dataset $D = \{z_1, \ldots, z_n\}$ and a test data point z_{test}, computing the Banzhaf values $\beta_i(v \mid z_{test})$ for kNN classifier, as in Eq.* (6)*, for an arbitrary $z_i \in D$ is* #P*-hard.*

The proof is provided in Appendix A [44].

3 EFFICIENT COMPUTATION OF BANZHAF VALUES FOR kNN CLASSIFIER

We first describe how to compute the Banzhaf values of all training points in $O(n^4 W)$ time using a dynamic programming (DP) approach, where W is the maximum sum of the top-k weights. As mentioned before, we assume a single test data point z_{test}. Then, we improve the time complexity to $O(Wkn^2)$ by introducing many novel ideas. Finally, we develop a more efficient algorithm for unweighted kNN with $O(nk^2)$ time complexity. Note that in practice k is typically small, such as a value smaller than 10, so the time complexity is essentially linear in n. We defer the discussion about practical implementation, and extension to multi-class and regression settings to Appendix C [44].

3.1 A standard DP algorithm

Following the definition of Banzhaf values in Equation (1), to compute $\beta_i(v)$ for any $z_i \in D$ we need to count the number of subsets $S \subseteq D \setminus \{z_i\}$ for which z_i is *pivotal*, that is, the addition of z_i changes the outcome of the game. Interestingly, we can avoid enumerating all (exponentially many) subsets S by performing the counting via *dynamic programming*.

Before presenting the counting algorithm, it is useful to determine in advance what the possible outcomes of adding a data point z_i to a pivotal subset are.

Lemma 2. *If $y_i = y_{test}$, it is impossible for z_i to turn $v(S) = 1$ into $v(S \cup \{z_i\}) = 0$. Similarly, if $y_i \neq y_{test}$, it is impossible for z_i to turn $v(S) = 0$ into $v(S \cup \{z_i\}) = 1$.*

The proof is provided in Appendix A [44].

We now describe the dynamic-programming approach for the counting task. Given a test point z_{test}, we rename the data points in the training dataset D so that $z_1, \ldots, z_n$ are sorted by non-decreasing distance to z_{test}. Let $D_{\bar{i}} = D \setminus \{z_i\}$.

We define a function $f_i(w, s, m)$ to represent the number of subsets $S \subseteq D_{\bar{i}}$ such that (1) $|S| = s$; (2) the sum of top-$\min\{s, k\}$ weights equals w; and (3) the $\min\{s, k\}$-th closest point of S to z_{test}

is z_m. Mathematically,

$$f_i(w,s,m) = \sum_{S\subseteq D_i} \mathbb{1}\left(|S| = s \text{ and } \sum_{i=1}^{\min\{s,k\}} w(z_i(S)) = w \text{ and } z_{\min\{s,k\}}(S) = z_m\right). \tag{9}$$

Recall that $z_i(S)$ is the i-th closest point to z_{test} among the data points in S. We will see shortly that the Banzhaf value $\beta_i(v)$ can be obtained by aggregating the values of f_i. To efficiently compute f_i, we derive the following recurrence.

Lemma 3. *If $s \le k$, then*

$$f_i(w,s,m) = \sum_{m'<m:m'\neq i} f_i(w - w(z_m), s-1, m'),$$

else, if $m > i$, then

$$f_i(w,s,m) = f_i(w,k,m)\cdot\binom{n-m}{s-k}.$$

The proof is provided in Appendix A [44]. Note that when $s > k$, there is no need to compute $f_i(w,s,m)$ for $m < i$, as inserting z_i into those subsets has no effect. We choose to construct f_i by gradually increasing the subset size s. We start with a special base case, $f_i(0,0,0) = 1$, which will be useful for $y_i = y_{\text{test}}$ as $v(\emptyset) = 0$. The base case for $s = 1$ is $f_i(w(z_m),1,m) = 1$ for all $m \neq i$, and the rest values are initialized to zero. Then, we fill the DP table of f_i by enumerating the weight values from $-W$ to W, the subset size s from 2 to $n-1$, and m from 1 to n with $m \neq i$.

It is not hard to see that the time complexity is $O(Wn^3)$ for each f_i, as each entry may take $O(n)$ time. Note that for $s > k$, computing the binomial coefficient also requires $O(n)$ time. Hence, computing all values of f_i results in $O(Wn^4)$ time in total.

Once the values of f_i have been computed, we can obtain the Banzhaf value of z_i with $y_i = y_{\text{test}}$ by

$$\beta_i(v) = \frac{1}{2^{n-1}}\left(C_{<k} + C_{\ge k}\right), \tag{10}$$

where

$$C_{<k} = \sum_{\substack{w\in(-w(z_i),0]\\ s\in[0,k-1]\\ m\in[0,n]:m\neq i}} f_i(w,s,m),$$

and

$$C_{\ge k} = \sum_{\substack{w\in(-w(z_i)+w(z_m),0]\\ s\in[k,n-1]\\ m>i}} f_i(w,s,m).$$

Note that we use the notation $[a,b]$ to denote a closed interval in the set of integers here. Both terms count the number of valid subsets S where z_i is pivotal. By Lemma 2, a positive z_i can only turn the outcome value from 0 to 1. Here $C_{<k}$ counts the number of subsets S whose size is less than k, and $C_{\ge k}$ counts the number of those whose size is at least k. The difference is that for $C_{\ge k}$, inserting z_i will push the current k-th closest point z_m out of the top-k neighbors. Therefore, the range of weight where z_i can be pivotal has to be adjusted according to $w(z_m)$.

Similarly, if $y_i \neq y_{\text{test}}$, then

$$\beta_i(v) = -\frac{1}{2^{n-1}}\left(C_{<k} + C_{\ge k}\right), \tag{11}$$

Algorithm 1: DP framework for kNN-Banzhaf values

Input: Dataset D, test point z_{test}, integer k

Let $z_1,\dots,z_n$ be the data points sorted by non-decreasing distance to z_{test}
Let W be the maximum sum of top-k weights
for $i = 1$ **to** n **do**
  $\beta_i(v) \leftarrow$ `BanzhafDP`$(i,\{z_1,\dots,z_n\},k,W)$
return $\beta_1(v),\dots,\beta_n(v)$

Algorithm 2: Standard dynamic programming algorithm

Function `BanzhafDP`$(i,\{z_1,\dots,z_n\},k,W)$**:**
  ▷ Initialize the DP table
  $f_i(w,s,m) \leftarrow 0$ for all $w \in [-W,W]$, $s \in [0,n-1]$, $m \in [1,n]$
  $f_i(0,0,0) \leftarrow 1$
  for $m = 1$ **to** n *where* $m \neq i$ **do**
    $f_i(w(z_m),1,m) \leftarrow 1$
  ▷ fill the DP table
  for $s = 2$ **to** $n-1$ **do**
    for $w = -W$ **to** W **do**
      for $m = 1$ **to** n *where* $m \neq i$ **do**
        if $s \le k$ **then**
          $f_i(w,s,m) \leftarrow \sum_{m'<m:m'\neq i} f_i(w - w(z_m), s-1, m')$
        else
          if $m > i$ **then**
            $f_i(w,s,m) \leftarrow f_i(w,k,m)\cdot\binom{n-m}{s-k}$
  if $y_i = y_{test}$ **then**
    return $\frac{1}{2^{n-1}}\left(C_{<k} + C_{\ge k}\right)$ *according to Eq.* (10)
  else
    return $-\frac{1}{2^{n-1}}\left(C_{<k} + C_{\ge k}\right)$ *according to Eq.* (11)

where

$$C_{<k} = \sum_{\substack{w\in(0,-w(z_i)]\\ s\in[0,k-1]\\ m\in[1,n]:m\neq i}} f_i(w,s,m),$$

and

$$C_{\ge k} = \sum_{\substack{w\in(0,-w(z_i)+w(z_m)]\\ s\in[k,n-1]\\ m>i}} f_i(w,s,m).$$

One difference from the positive case is that the empty set S represented by $f_i(0,0,0)$ is excluded in $C_{<k}$.

The algorithm is shown in Algorithm 2 with an example illustrated in Example A1. Algorithm 1 invokes Algorithm 2 for each data point $z_i \in D$. We summarize the time complexity of our first non-optimized algorithm as follows.

Theorem 4 (Standard DP)**.** *The dynamic programming algorithm in Algorithms 1 and 2 for computing the kNN-Banzhaf values of all*

data points runs in $O(Wn^4)$ time, where W is the maximum sum of the top-k weights and n is the number of data points in D.

3.2 A more efficient DP algorithm

While the dynamic-programming algorithm we presented in the previous section has pseudo-polynomial complexity, its running time is prohibitively expensive in practice for any other than small datasets. To improve the running time, we introduce several novel techniques, including pre-computing partial sums and exploiting the recurrence relation of the binomial coefficients. The result is an improved version of the DP algorithm with running time $O(Wkn^2)$.

In our analysis, we distinguish two cases, $s \leq k$ and $s > k$.

3.2.1 Run time improvement for $s \leq k$ In the standard DP, the time complexity of computing $C_{<k}$ is $O(Wkn^2)$ for each f_i. We show how to improve it to $O(Wkn)$ by pre-computing partial sums.

For each term $f_i(w, s, m)$, we need to compute the sum of $f_i(w - w(z_m), s-1, m')$, for $m' < m$. This summation can be sped up if we pre-compute the partial sums for each w and s. In particular, we define a partial sum

$$M_i(w, s, m) = \sum_{m' < m : m' \neq i} f_i(w, s, m').$$

Note that for fixed values of w and s, the partial sums $M_i(w, s, m)$ can be computed in $O(n)$ time for all values of m, by the recurrence

$$M_i(w, s, m+1) = M_i(w, s, m) + f_i(w, s, m).$$

Then, for $s \leq k$, we can rewrite $f_i(w, s, m)$ in Lemma 3 as

$$f_i(w, s, m) = M_i(w - w(z_m), s-1, m). \tag{12}$$

By the above modification, we can compute each $f_i(w, s, m)$ entry in $O(1)$ time, at the expense of pre-computing $M_i(w, s, m)$ in $O(Wkn)$ time for all w, m and $s \leq k$. Thus, the time complexity of $C_{<k}$ reduces to $O(Wkn)$ for each f_i, and $O(Wkn^2)$ for all f_i.

3.2.2 Run time improvement for $s > k$ To accelerate the computation of $C_{\geq k}$, our first key observation is that $C_{\geq k}$ does not require access to individual $f_i(w, s, m)$ terms but their sum. Therefore, it is sufficient to compute the sum of f_i's for $s \geq k$. Formally, we define

$$\begin{aligned} F_i(w, m) &= \sum_{s \in [k, n-1]} f_i(w, s, m) = \sum_{s \in [k, n-1]} f_i(w, k, m) \cdot \binom{n-m}{s-k} \\ &= f_i(w, k, m) \cdot g(m), \end{aligned}$$

where we set $g(m) = \sum_{j \in [0, n-k-1]} \binom{n-m}{j}$. We will see shortly that to compute $C_{\geq k}$, it is sufficient to compute $F_i(w, m)$ for all w and m. Thus, if we can devise a fast way to compute $g(m)$ for all m, we can also improve the time complexity of $C_{\geq k}$. Note that $g(m)$ is independent of i, so we only need to compute it once for all i. Once we have pre-computed $g(m)$, each $F_i(w, m)$ entry can be computed in $O(1)$ time.

We exploit the recurrence relation of the binomial coefficients with different j to compute $g(m)$ efficiently, that is,

$$\binom{n-m}{j+1} = \binom{n-m}{j} \frac{n-m-j}{j+1}.$$

Therefore, fixing m, $g(m)$ can be computed in $O(n)$ time, as each summand costs $O(1)$ time by using the above recurrence relation.

Furthermore, we use the following recurrence for $g(m)$ across different values of m that further reduces the time complexity.

Lemma 5. *For the function $g(m) = \sum_{j \in [0, n-k-1]} \binom{n-m}{j}$ it holds that*

$$g(m) = 2 \cdot g(m+1) - \binom{n-m-1}{n-k-1}.$$

The proof is provided in Appendix A [44]. Lemma 5 allows us to compute $g(m)$ for all m in $O(n)$ time in total, rather than $O(n^2)$ if computed independently. More specifically, we first compute $g(m)$ for the largest value of m in $O(n)$ time, and then use the recurrence to compute the next $g(m-1)$ values in $O(1)$ time. This is possible because as m increases, the coefficient in Lemma 5 can be updated in $O(1)$ time using the following identity:

$$\binom{n-(m+1)-1}{n-k-1} = \binom{n-m-1}{n-k-1} \cdot \frac{k-m}{n-m-1}.$$

Actually, there are at most k such coefficients, as $\binom{n-m-1}{n-k-1} = 0$ for $m > k$. With the above adaptations, we now rewrite $C_{\geq k}$ when $y_i = y_{\text{test}}$ as

$$C_{\geq k} = \sum_{\substack{w \in (-w(z_i) + w(z_m), 0] \\ m > i}} F_i(w, m). \tag{13}$$

The case for $y_i \neq y_{\text{test}}$ is similar. For each value of i, the time complexity of $C_{\geq k}$ is $O(Wn)$ given pre-computed values of $g(m)$, which yields $O(Wn^2)$ in total, for all i. Hence, $C_{\geq k}$ is no longer the bottleneck of the DP algorithm. The total time complexity for both $C_{<k}$ and $C_{\geq k}$ reduces to $O(Wkn^2)$, for all i.

The improved DP algorithm is displayed in Algorithm 3, and its time complexity is summarized in the following theorem.

Theorem 6 (Efficient DP)**.** *The improved dynamic programming algorithm in Algorithms 1 and 3 for computing the Banzhaf values for the kNN classifier of all data points runs in $O(Wkn^2)$ time, where W is the maximum sum of the top-k weights, k is the number of nearest neighbors, and n is the number of data points in D.*

3.3 A near-linear time DP algorithm for unweighted kNN classifier

The improved DP algorithm presented in Section 3.2, although it brings a large improvement compared to standard DP, it still requires quadratic time in n and is not scalable for very large datasets. Thus, as a further improvement, we propose a specialized DP algorithm for unweighted kNN with a time complexity of $O(nk^2)$. Recall that k is usually small in practice, so the running time is essentially linear.

3.3.1 Re-designing the state space The design of the specialized DP algorithm deviates significantly from the algorithm presented in Section 3.2. In a nutshell, it relies on two novel ideas. First, we show how to reduce the size of the state space of the DP table, which was previously parameterized by three variables in each function f_i. Second, we show how to avoid similar computations of f_i for different values of i.

We now introduce our solution to the above two challenges.

Let $z_1, \ldots, z_n$ be the training data points sorted by non-decreasing distance to z_{test}, and let $D_i = \{z_1, \ldots, z_i\}$ and $D^i = \{z_i, \ldots, z_n\}$.

Algorithm 3: Efficient dynamic programming algorithm

Function `BanzhafFastDP`$(i, \{z_1, \ldots, z_n\}, k, W)$**:**

▷ Pre-compute $g(m)$ using recurrence relation in Lemma 5

$g(n) \leftarrow \sum_{j=0}^{n-k-1} \binom{0}{j} = 1$

for $m = n-1$ **to** 1 **do**

  $g(m) \leftarrow 2 \cdot g(m+1) - \binom{n-m-1}{n-k-1}$

▷ Initialize the DP table

$f_i(0,0,0) \leftarrow 1$

for $m = 1$ **to** n *where* $m \neq i$ **do**

  $f_i(w(z_m), 1, m) \leftarrow 1$

▷ Compute for case $s \leq k$ with partial sums optimization

for $s = 2$ **to** k **do**

  for $w = -W$ **to** W **do**

    $M_i(w, s-1, 1) \leftarrow 0$

    for $m = 2$ **to** n *where* $m \neq i$ **do**

      $M_i(w, s-1, m) \leftarrow M_i(w, s-1, m-1) + f_i(w, s-1, m-1)$

  for $w = -W$ **to** W **do**

    for $m = 1$ **to** n *where* $m \neq i$ **do**

      $f_i(w, s, m) \leftarrow M_i(w - w(z_m), s-1, m)$

▷ Compute $F_i(w, m)$ for case $s > k$

for $w = -W$ **to** W **do**

  for $m = i+1$ **to** n **do**

    $F_i(w, m) \leftarrow f_i(w, k, m) \cdot g(m)$

if $y_i = y_{test}$ **then**

  return $\frac{1}{2^{n-1}} \left(C_{<k} + C_{\geq k}\right)$ *according to Eqs.* (10) *and* (13)

else

  return $-\frac{1}{2^{n-1}} \left(C_{<k} + C_{\geq k}\right)$ *according to Eqs.* (11) *and* (13)

We define two functions

$$f_i(w,s) = \sum_{S \subseteq D_i} \mathbb{1}\left(|S| = s \text{ and } \sum_{i=1}^{\min\{|S|,k\}} w(z_i(S)) = w\right),$$

and

$$b_i(w,s) = \sum_{S \subseteq D^i} \mathbb{1}\left(|S| = s \text{ and } \sum_{i=1}^{\min\{|S|,k\}} w(z_i(S)) = w\right).$$

We call f_i the *forward function* and b_i the *backward function*. Compared with the standard DP, we choose not to keep track of the $\min\{|S|, k\}$-th closest point (previously denoted as z_m) in S. To make up for the missing information, we maintain two additional *signed* backward functions, which are defined as follows:

$$b_i^-(w,t) = \sum_{S \subseteq D^i} \mathbb{1}\left(y_t(S) \neq y_{\text{test}} \text{ and } \sum_{i=1}^{t} w(z_i(S)) = w\right),$$

and

$$b_i^+(w,t) = \sum_{S \subseteq D^i} \mathbb{1}\left(y_t(S) = y_{\text{test}} \text{ and } \sum_{i=1}^{t} w(z_i(S)) = w\right).$$

The rationale is that in the unweighted kNN case it is not necessary to know exactly which point gets pushed out of the top-k neighbors, when inserting z_i into a subset S of size larger than or equal to k. Instead, since the data points are unweighted, it is only necessary to know the sign of the data point. In the signed backward functions, the label of the t-th closest point of subset S determines the sign of the subset. Note that we do not require the subset S to be of size exactly t in the signed backward functions, and we only require the sum of its top-t weights to be w.

We now describe the recurrence relations for efficient construction of the above functions. We start with the forward and backward functions.

Lemma 7. *If* $s \leq k$*, then*

$$f_i(w,s) = f_{i-1}(w - w(z_i), s-1) + f_{i-1}(w,s),$$

and

$$b_i(w,s) = b_{i+1}(w - w(z_i), s-1) + b_{i+1}(w,s).$$

We omit the proof as it is fairly straightforward. Practically, we start with $f_i(0,0) = 1$ for all $i \in [n]$, and $f_1(w(z_1), 1) = 1$; the rest of the entries are initialized to zero. Then we process $f_2, \ldots, f_n$ in a forward manner. The procedure for b_i is similar except that it is done in a backward manner with an additional base case $b_{n+1}(0,0) = 1$. By Lemma 7, the time to construct f_i and b_i, for all $i \in [n]$ and $s \leq k$, is $O(Wnk)$.

We continue with the signed backward functions.

Lemma 8. *For* $t = 1$*, we have*

$$b_i^-(w(z_i), 1) = \begin{cases} 2^{n-i} + b_{i+1}^-(w(z_i), 1) & \text{if } y_i \neq y_{test} \\ b_{i+1}^-(w(z_i), 1) & \text{otherwise.} \end{cases}$$

For $t \in [2, k]$*, we have*

$$b_i^-(w,t) = b_{i+1}^-(w - w(z_i), t-1) + b_{i+1}^-(w,t).$$

The update rules for b_i^+ *are identical except for using the condition* $y_i = y_{test}$ *when* $t = 1$*.*

The proof is provided in Appendix A [44]. The update rules work backward from $i = n$ to 1 for each t with an additional base case $b_{n+1}(0,0) = 1$. Fixing a size t, for each i, we sum up the cases where z_i is in the subset S or not. Since we work backward, the newly added z_i is always the closest point to z_{test} in the current subset S. This makes the case of $t = 1$ special because if the current point z_i is included, it automatically decides the sign of the subset. Therefore, we need to check if its sign matches that of the function. If the signs match and it is included, then further including any point behind it will not change the sign and top-t weights, which means that there are 2^{n-i} feasible subsets. By Lemma 8, the time to construct b_i^- and b_i^+, for all $i \in [n]$ up to $t \leq k$, is $O(Wnk)$.

3.3.2 Computing the kNN-Banzhaf values via aggregation After the above preparations, we are ready to compute the kNN-Banzhaf values. We point out that all points z_i can share the above common states, as one can compute the kNN-Banzhaf values for all z_i by aggregating the needed subsets $S \subseteq D_i$ from D_{i-1} and D^{i+1} for each z_i. That is, if $y_i = y_{\text{test}}$, the Banzhaf value of z_i can be computed as

$$\beta_i(v) = \frac{1}{2^{n-1}} \left(C_- + C_{<k}\right), \tag{14}$$

where

$$C_- = \sum_{\substack{w\in(-W,W)\\ s\in[0,k-1]}} \left(f_{i-1}(w,s) \cdot \sum_{\substack{w'+w\in[-1,0]\\ t=k-s}} b^-_{i+1}(w',t) \right)$$

and

$$C_{<k} = \sum_{\substack{w\in(-W,W)\\ s\in[0,k-1]}} \left(f_{i-1}(w,s) \cdot \sum_{\substack{w'+w=0\\ s+t\in[0,k-1]}} b_{i+1}(w',t) \right).$$

In each C term, we combine a subset $S_1 \subseteq D_{i-1}$ and a subset $S_2 \subseteq D^{i+1}$ to form a valid subset $S = S_1 \cup S_2$ where z_i is pivotal. More specifically, we require the weight of S to fall into a certain range so that z_i can change the outcome. Note that $|S_1|$ has to be smaller than k, or otherwise z_i cannot be pivotal. Besides, we need to distinguish two cases depending on the size of the subset S. The first C_- term handles subsets S whose size is at least k, *and* whose k-th closest point is negative. This is because if the k-th closest point is positive, inserting the positive point z_i will not change the value of the subset. The second $C_{<k}$ term counts the number of valid subsets whose size is less than k. Since the subset size is less than k, no point will be pushed out of the top-k neighbors, so no signed functions are needed.

For each β_i, it costs $O(Wk)$ time to compute C_-, but needs $O(Wk^2)$ time to compute $C_{<k}$. To speed up the computation, we pre-compute the partial sum of the backward function b_i by letting

$$B_i(w,t) = \sum_{t'\in[0,t]} b_i(w,t'),$$

which can be computed in $O(Wnk)$ time for all w, i and t. Then, $C_{<k}$ can be rewritten as

$$C_{<k} = \sum_{\substack{w\in(-W,W)\\ s\in[0,k-1]}} \left(f_{i-1}(w,s) \cdot B_i(-w, k-1-s) \right),$$

which can now be computed in $O(Wk)$ time.

The case for $y_i \neq y_{\text{test}}$ is similar by summing up C_+ and $C_{<k}$ as

$$\beta_i(v) = -\frac{1}{2^{n-1}} \left(C_+ + C_{<k} \right), \tag{15}$$

where

$$C_+ = \sum_{\substack{w\in(-W,W)\\ s\in[0,k-1]}} \left(f_{i-1}(w,s) \cdot \sum_{\substack{w'+w\in[1,2]\\ t=k-s}} b^-_{i+1}(w',t) \right)$$

and

$$C_{<k} = \sum_{\substack{w\in(-W,W)\\ s\in[0,k-1]}} \left(f_{i-1}(w,s) \cdot \sum_{\substack{w'+w=1\\ s+t\in[0,k-1]}} b_{i+1}(w',t) \right).$$

Hence, the total time complexity of β_i for all i is $O(Wnk)$. Note that $W = O(k)$ for unweighted kNN, so it is $O(nk^2)$ in total.

The specialized unweighted DP algorithm (brief version) is displayed in Algorithm 4. The full version is deferred to Appendix A [44]due to space constraints. The algorithm does not follow the framework in Algorithm 1 because all data points share the same DP tables. We summarize its time complexity as follows.

Algorithm 4: Specialized DP algorithm for unweighted kNN (brief version)

Input: Dataset D, test point z_{test}, integer k
Sort $z_1, \ldots, z_n$ by non-decreasing distance to z_{test}
$W \leftarrow k+1$
Compute forward functions $f_i(w,s)$ according to Lemma 7
Compute backward functions $b_i(w,s)$ according to Lemma 7
Compute signed backward functions $b^+_i(w,t)$ and $b^-_i(w,t)$ according to Lemma 8
for $i = 1$ **to** n **do**
  if $y_i = y_{test}$ **then**
    $\beta_i(v) \leftarrow \frac{1}{2^{n-1}}(C_- + C_{<k})$ according to Eq. (14)
  else
    $\beta_i(v) \leftarrow -\frac{1}{2^{n-1}}(C_+ + C_{<k})$ according to Eq. (15)
return $\{\beta_1(v), \ldots, \beta_n(v)\}$

Theorem 9 (Efficient DP for unweighted kNN)**.** *The dynamic programming algorithm in Algorithm 4 for computing the Banzhaf values for the unweighted kNN classifier of all data points runs in $O(nk^2)$ time, where n is the number of data points and k is the number of nearest neighbors.*

We demonstrate the algorithm with an example in Example A2.

4 EFFICIENT MONTE CARLO ESTIMATION OF BANZHAF VALUES

While the dynamic-programming approach provides exact computation of Banzhaf values, it may become computationally prohibitive for very large datasets due to its quadratic complexity. In practice, it is often sufficient to obtain an approximate Banzhaf value. This motivates us to develop efficient Monte Carlo methods that provide unbiased estimates of Banzhaf values with controllable approximation error. We introduce two Monte Carlo methods, one by sampling coalitions and the other by sampling permutations. Both of them are optimized for the kNN value function with significantly stronger statistical efficiency. The sample complexity can be analyzed using standard techniques, e.g., applying the Chernoff bound; we refer the reader to [15, 26] for more details.

4.1 Efficient coalition-based sampling

The most straightforward Monte Carlo approach directly samples from the exponential number of possible coalitions. Recall that in Eq. (1), the Banzhaf value of player $i \in N$ is defined as the average marginal contribution of player i to all possible coalitions $S \subseteq N \setminus \{i\}$. Thus, it can be estimated by uniformly sampling coalitions $S \subseteq N \setminus \{i\}$ and computing the average marginal contribution. Specifically, for each player $i \in N$, we sample m coalitions $S_1, S_2, \ldots, S_m$ where each coalition S_j is constructed by including each player in $N \setminus \{i\}$ independently with probability $\frac{1}{2}$. Then, the

estimate of the Banzhaf value is

$$\beta_i(v) = \frac{1}{m} \sum_{j=1}^{m} \left[v(S_j \cup \{i\}) - v(S_j) \right]. \tag{16}$$

This approach has the advantage of being conceptually straightforward and providing unbiased estimates. However, it requires $O(mn)$ coalition evaluations, where each evaluation has complexity $O(n)$ for the kNN value function, resulting in an overall complexity of $O(mn^2)$.

We propose to improve this approach by exploiting the locality property of the kNN value function. The key idea is to sample one common coalition S where the marginal contribution of all players (data points) in D can be derived simultaneously. Specifically, we include each data point of D into S independently with probability $\frac{1}{2}$. Then, it is sufficient to maintain only the top $k+1$ players in the sorted list of S. To derive the marginal contribution of each player i, we remove it from S if it already exists in S, or compare it with the $\min\{k, |S|\}$-th closest player in S, and decide whether to insert it into S. Equipped with the above optimized operations, the complexity of processing m sampled coalitions for all players can be reduced to $O(mn \log k)$.

4.2 Efficient permutation-based sampling

We also consider a Monte Carlo method that leverages the permutation-based formulation of Banzhaf values. Permutation-based sampling is mostly well-known for the Shapley values, but it is not directly applicable to the Banzhaf values. However, we manage to obtain such a formulation by a re-weighting scheme. Specifically, the Banzhaf values can be expressed as:

$$\beta_i(v) = \frac{n}{2^{n-1}} \sum_{\pi \in \Pi(N)} \binom{n-1}{|\pi_i|} \left[v(\pi_i \cup \{i\}) - v(\pi_i) \right], \tag{17}$$

where $\Pi(N)$ denotes the set of all permutations of N, and π_i represents the set of players that precede player i in π. This formulation suggests that we can sample random permutations π and compute weighted marginal contributions as the estimate of the Banzhaf value. For each sampled permutation π, we traverse the permutation and compute the marginal contribution of each player when they join the coalition formed by their predecessors. Note that each permutation yields marginal contribution estimates for all players simultaneously. The key insight is that we only need to maintain a sorted list of at most k players in a kNN classifier, and process the traversal along a permutation efficiently within $O(n \log k)$ time. Since the set of all permutations is of size $n!$, we resort to sampling a subset of m random permutations as an approximation, and the total complexity is $O(mn \log k)$.

5 RELATED WORK

Data Valuation. Data valuation aims to assign importance scores to individual training data points [12, 17, 34]. Simple approaches such as the leave-one-out (LOO) value measure the marginal contribution of a data point to the value function (e.g., model accuracy) when it is removed from the training procedure. DataShapley [10, 16, 42] and its variants such as BetaShapley [22], DataBanzhaf [39], and least core [43] are all based on the LOO principle, but differ in the way the marginal contributions are aggregated. We discuss several notable options beyond Shapley values below. Feldman and Zhang [9] simulate the data values by LOO retraining, albeit constrained on a small sample of the training data, while DataModels [14] predict the LOO values by machine-learning models at the expense of exactness. Another line of popular methods is based on gradients. TracIn [30] estimates the importance of a training data point by tracing the change in the test loss caused by the data point during the training process. Variations of influence functions [21, 32] have their roots in robust statistics [5], and offer a gradient-based approximation of the LOO values.

kNN-based Game-theoretic Values. Unweighted kNN-Shapley with a soft label was first treated by Jia et al. [15] and later refined by Wang and Jia [40]. The weighted case with a hard label has been addressed by Wang et al. [41] by dynamic programming with a time complexity of $O(Wk^2n^2)$, which is slower than our Algorithm 3 by a factor of k. Their key idea is to have the binomial coefficients in Lemma 3 and those in the definition of the Shapley value (see Eq. (3)) cancel out. However, their technique does not carry over to the kNN-Banzhaf value here, as no binomial coefficient is involved in the definition of the Banzhaf value (Eq. (1)). To the best of our knowledge, we are the first to work on computing Banzhaf values for the kNN problem, though Banzhaf values have seen applications in other settings [19, 39].

Cooperative Game Theory. Apart from the Shapley [33] and Banzhaf [2] values mentioned above, the hard-label kNN-based values are also closely related to majority voting games in cooperative game theory. It has been shown that weighted majority voting is weakly **NP**-hard, and there exist dynamic programming algorithms for computing these values (also known as power indices) [27]. However, these algorithms are not directly applicable to the kNN-based values, as they do not require a ranking of the players. Yet, one technique in our unweighted DP algorithm that aggregates the forward and backward functions is inspired by Uno [35]. While our algorithms are tailored to kNN, analogous polynomial-time data-point valuation schemes are uncommon; existing efficient Shapley-value techniques such as TreeSHAP [25] apply to feature rather than data importance.

6 EXPERIMENTS

We conduct a comprehensive experimental evaluation for kNN classification from the perspectives of computational efficiency and practical applications, including point removal, data selection, and mislabel detection. Our main focus is hard-label kNN-Banzhaf valuation in both weighted and unweighted settings, with exact DP and Monte Carlo variants. We also include hard-label and soft-label Shapley baselines to make the trade-offs explicit, including settings where those baselines are more favorable. Due to the space limit, we defer missing details of the experiments to Appendix D [44].

Datasets. We use 12 classification datasets that vary in size, dimensionality, and number of classes [20, 37]. The datasets are: *phoneme*, *wind*, *cpu*, *vehicle*, *pol*, *fraud*, *2dplanes*, *apsfail*, *cifar10*, *mnist*, *creditcard*, and *click*. Detailed dataset statistics can be found in Table 1. We also use a synthetic dataset to evaluate the scalability of the proposed algorithms.

Table 1: Dataset statistics.

Dataset	# Data Points	# Features	# Classes
phoneme	5404	5	2
wind	6574	14	2
cpu	8192	21	2
vehicle	10000	100	2
pol	15000	48	2
fraud	28000	30	2
2dplanes	40768	10	2
apsfail	60000	170	2
cifar10	60000	3072	10
mnist	70000	784	10
creditcard	284807	23	2
click	1000000	11	2

Baselines. We adopt the following representative baselines:

- `sv` and `sv-uw`: the DP algorithm for hard-label weighted and unweighted kNN-Shapley [41].
- `sv-soft`: the analytical solution for soft-label unweighted kNN-Shapley [15].
- `beta-mc`: the Beta Shapley values that replace the weights in the Shapley values with a Beta-distributed weights [22].
- `lava`: the Lava values are based on the sensitivity analysis of the class-wise Wasserstein distance between the training and test datasets [18].
- `loo`: the classic leave-one-out (LOO) values.
- `inf`: the influence function which is a gradient-based approximation of the LOO values [21].
- `rand`: the random values.
- `bf`: the brute-force algorithm for kNN-Banzhaf.

We adopt the implementation of `beta-mc` (with default parameters $\alpha = 4, \beta = 1$), `lava` and `inf` by Jiang et al. [17]. Algorithms with a suffix of `-mc` are based on Monte Carlo sampling.

Our algorithms are abbreviated as follows:

- `bv`: the standard DP algorithm for weighted kNN-Banzhaf, described in Section 3.1.
- `bv-fast`: the improved DP algorithm for weighted kNN-Banzhaf, described in Section 3.2.
- `bv-uw`: the near-linear time DP algorithm for unweighted kNN-Banzhaf, described in Section 3.3.

We set the number of nearest neighbors to $k = 5$, a uniform weight function, and 5% of the raw data as the test set for all methods unless otherwise specified. For weighted kNN, we discretize the weight space of `bv`, `bv-fast`, and `sv` into 7 bits. For all experimental results, we report the average over five independent runs.

Experimental setting. All experiments are conducted on a machine with 20 cores of Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz and 128 GB RAM, running Ubuntu 20.04.3 LTS with Python 3.8.12. All runtime measurements use no algorithm-specific parallelism. We report wall-clock time and use a time budget of 10 hours.

6.1 Computational efficiency

Scalability of the DP algorithms. We compare the computational efficiency of our proposed algorithms and other methods by measuring their running time on a synthetic 32-dimensional dataset of varying sizes with one random test point. We use a uniform weight function. The results for our DP algorithms and other methods are shown in Fig. 2a.

The experimental results confirm that the observed running times of the proposed DP algorithms align closely with our theoretical time complexity analysis presented in Section 3. The brute-force algorithm cannot finish within 10 hours for a size as small as 30. The unweighted DP `bv-uw` indeed has an almost linear time complexity. Although we fix the parameter k, its running time still slightly deviates from the linear one. This is because of the additional logarithmic overhead of processing huge integers such as those involved in Lemma 8 that oversize the machine word size. Therefore, the nearly linear-time soft-label method `sv-soft` is more scalable than our `bv-uw`. The improved DP `bv-fast` is much more scalable than the standard DP `bv`. Under the 10-hour budget, the exact hard-label Shapley methods `sv` and `sv-uw` are unable to complete runs with $n \geq 10^4$, unlike `bv-fast` (10^4) and `bv-uw` (10^6).

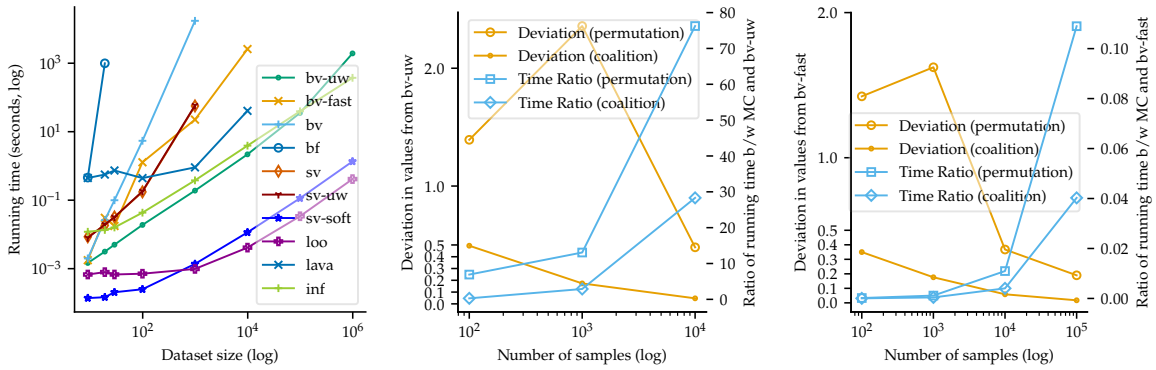
Performance of the Monte Carlo methods. We also compare the performance of the proposed Monte Carlo methods in Section 4 with the unweighted and weighted DP algorithms on a dataset with $n = 100\,\text{K}$ and $10\,\text{K}$, respectively. We define the *deviation* of the Monte Carlo estimates from the exact DP estimates as the maximum absolute difference between the two estimates among all data points, further normalized by the maximum value of the exact DP estimates. The results are shown in Fig. 2b and Fig. 2c against unweighted `bv-uw` and weighted `bv-fast`, respectively.

It is clear that the deviation of the permutation-based Monte Carlo estimate is less stable and converges more slowly than the coalition-based one, given the same number of samples. Besides, the former is also more time-consuming due to the maintenance of dynamic top-k neighbors and the additional overhead of computing re-weighting coefficients. The coalition-based method can provide a speed-up of 25× over the weighted DP (but not the unweighted DP) if one is willing to accept a deviation that is around 0.02. However, to obtain a further smaller deviation, the number of samples required can be large and may not be practical.

6.2 Point removal

To examine the ability of the proposed algorithms in discerning data quality, we consider an application that is commonly used in the literature, i.e., point removal. After obtaining a value for each data point, we gradually remove the data points with the largest values, one after another, and monitor the model performance on the remaining data. A sharper drop in performance indicates the effectiveness of the valuation. We use the accuracy of an unweighted kNN classifier over a test set as the performance metric.

The results are shown in Figs. 3a and 3d (more in Appendix D [44]). Among game-theoretic methods, the hard-label Banzhaf and Shapley values are often the best among many datasets, followed by `beta-mc` and the soft-label Shapley values `sv-soft`. It is worth noting that the weighted kNN methods `bv-fast` and `sv` often obtain a higher accuracy than the unweighted ones before removing



(a) Runtime performance of different algorithms on a synthetic dataset of varying sizes.

(b) The deviation of the Monte Carlo estimates from the unweighted DP estimates as a function of the number of samples.

(c) The deviation of the Monte Carlo estimates from the weighted DP estimates as a function of the number of samples.

Figure 2: Computational efficiency of the proposed algorithms.

any data points. Baseline `inf` underperforms, while `lava` exhibits inconsistent results; the model accuracy is expected to drop after removing the most valuable points. The accuracy of `loo` converges much earlier, as it is the simplest measure that only considers the impact of a point on the whole remaining data.

6.3 Data selection

Another popular application of data valuation is data selection, i.e., selecting a subset of data points that are most important to the model performance. To evaluate this approach, we start with a small number of random data points as the warm-up set, and gradually include more data points with the largest values. We monitor the model performance as the selected data points are incrementally included in the training set. The results are shown in Figs. 3b and 3e (more details in Appendix D [44]).

Similar to the point-removal experiment, `loo` and `rand` are less capable, while `lava` and `inf` exhibit inconsistent results. Among the rest, all game-theoretic methods are fairly comparable to each other, and our proposed algorithms `bv-fast` and `bv-uw` are either the best or very close to the best in many datasets. Our Banzhaf values thus offer a new efficient and robust way to select high-quality data points.

6.4 Mislabel detection

Another important application of data valuation is detecting mislabeled training examples, i.e., those data points of a low quality. We explicitly consider two objectives in this subsection: (i) detecting injected label noise, and (ii) identifying low-value points whose correction most improves model performance.

For objective (i), we randomly select 5% of the training data points, and flip their labels to simulate the ground truth of label noise (denoted as D_{noise}). After computing the values for the corrupted dataset, we identify the 5% of data points with the lowest values as our predicted mislabeled points, denoted as D_{pred}. We measure the detection performance using the $F1$ score between the predicted mislabeled set D_{pred} and the true mislabeled set D_{noise}, i.e., $F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}}$, where $\text{Precision} = \frac{|D_{\text{pred}} \cap D_{\text{noise}}|}{|D_{\text{pred}}|}$, and $\text{Recall} = \frac{|D_{\text{pred}} \cap D_{\text{noise}}|}{|D_{\text{noise}}|}$. A higher $F1$ score indicates more accurate detection of mislabeled training examples.

The results show that the soft-label Shapley values (`sv-soft`) and the influence function (`inf`) achieve substantially higher $F1$ (and AUC) than the other methods across datasets; detailed per-dataset numbers are provided in Tables A1 and A4. Thus, if the sole objective is to recover randomly flipped labels by $F1$ score, these methods are preferable.

For objective (ii), we conduct a further experiment to investigate the *influence* of the identified low-value points. While soft-label methods detect a larger fraction of the injected mislabeled points, many of the detected points have limited impact on overall model performance. The strongest signal for detecting a mislabeled point is its label inconsistency with its neighbors. Soft-label valuation can capture this signal as soon as some of its neighbors appear in the test set. In contrast, the hard-label valuation may not consider a noisy point harmful, because it further requires the noisy point to be a game changer for the prediction of its neighbors. However, when a negative noisy point is surrounded by a majority of positive points, it is unlikely for it to change the prediction. More importantly, suspected mislabeled points often require further verification by domain experts in practice, which is expensive and time-consuming. Thus, manually checking mislabeled but irrelevant points is clearly not desirable.

To verify the above analysis, our second experiment flips the labels of the identified mislabeled points (i.e., those with the smallest values), and monitors the model performance on the modified dataset. The results are shown in Figs. 3c and 3f (more in Appendix D [44]). It is evident that the model performance of `sv-soft` often stays unchanged after flipping the labels of the first few points. Many of those top points are indeed the mislabeled ones, but they contribute little to the model performance. In contrast, hard-label valuation like `bv-fast` can identify the critical low-quality points, which are not necessarily the random mislabeled ones, and increase the model accuracy significantly. Therefore, the decision whether to

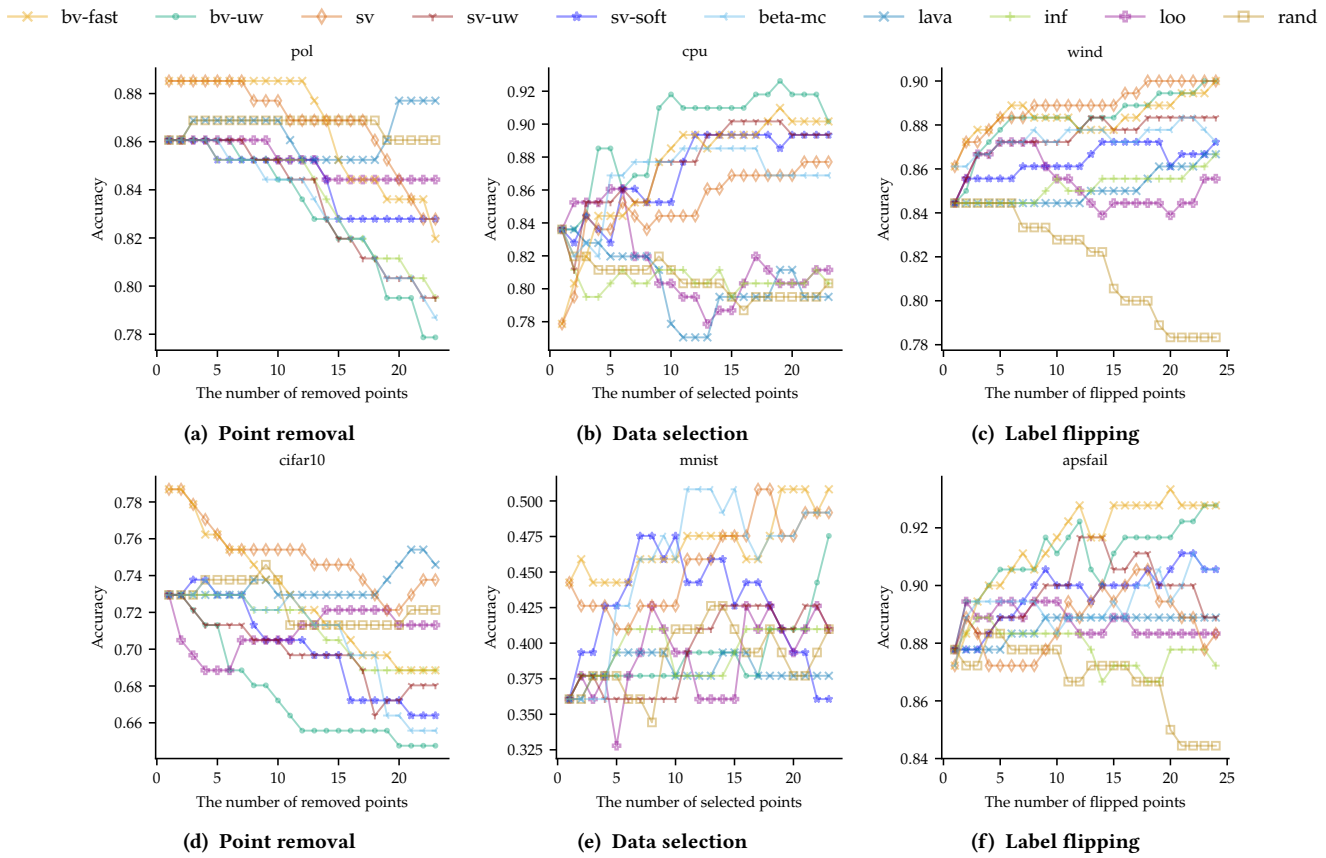


Figure 3: Point removal results on different datasets: model accuracy as the data points with the largest values are removed (3a, 3d). Data selection results on different datasets: model accuracy as the data points with the largest values are selected (3b, 3e). The effect of flipping the labels of the data points with the smallest values on the model performance (3c, 3f).

use soft-label or hard-label valuation depends on the application objective: pure noise detection versus identifying *decisive* low-quality points for performance-oriented data cleaning.

7 CONCLUSION

In this paper, we addressed the computational challenge of computing exact hard-label Banzhaf values for data valuation in k-nearest neighbor classifiers. Despite proving that the problem is #P-hard, we successfully exploited the locality properties inherent in kNN classifiers to develop practical and efficient algorithms, including dynamic programming algorithms with significantly improved time complexities: $O(Wkn^2)$ for weighted kNN and $O(nk^2)$ for unweighted kNN. We also provide efficient Monte Carlo estimation methods, and conduct comprehensive experimental evaluations that demonstrate both computational efficiency and practical effectiveness.

Several promising research directions emerge from this work, including extending our dynamic-programming framework to other game-theoretic values, developing hybrid approaches that combine the benefits of hard-label and soft-label valuation methods for more comprehensive data quality assessment, and exploring the integration of our algorithms with modern deep learning pipelines, in the contexts such as retrieval-augmented generation and embedding-based similarity search. It would also be interesting to extend our methods for other machine-learning tasks, such as regression and clustering.

ACKNOWLEDGMENTS

This work was supported by Guangdong Provincial College Youth Innovative Talent Project (Grant No. 2025KQNCX075), Natural Science Foundation of Top Talent of SZTU (Grant No. GDRC202520), the ERC Advanced Grant REBOUND (834862), the Swedish Research Council project ExCLUS (2024-05603), the Marie Curie Doctoral Training Network ARMADA (101168951), and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

REFERENCES

[1] Yoram Bachrach, Evangelos Markakis, Ezra Resnick, Ariel D Procaccia, Jeffrey S Rosenschein, and Amin Saberi. 2010. Approximating power indices: theoretical and empirical analysis. *Autonomous Agents and Multi-Agent Systems* 20, 2 (2010), 105–122.

[2] John F Banzhaf III. 1965. Weighted voting doesn't work: A mathematical analysis. *Rutgers Law Review* 19 (1965), 317–343.

[3] Olivier Bousquet and André Elisseeff. 2002. Stability and generalization. *Journal of machine learning research* 2, Mar (2002), 499–526.

[4] Javier Castro, Daniel Gómez, and Juan Tejada. 2009. Polynomial calculation of the Shapley value based on sampling. *Comput. Oper. Res.* 36, 5 (2009), 1726–1730.

[5] R. Dennis Cook and Sanford Weisberg. 1982. *Residuals and Influence in Regression*. Chapman and Hall, New York, NY, USA.

[6] Xiaotie Deng and Christos H. Papadimitriou. 1994. On the Complexity of Cooperative Solution Concepts. *Math. Oper. Res.* 19, 2 (1994), 257–266.

[7] Edith Elkind and Jörg Rothe. 2016. Cooperative game theory. *Economics and computation: an introduction to algorithmic game theory, computational social choice, and fair division* (2016), 135–193.

[8] S. Shaheen Fatima, Michael J. Wooldridge, and Nicholas R. Jennings. 2008. A linear approximation method for the Shapley value. *Artif. Intell.* 172, 14 (2008), 1673–1699.

[9] Vitaly Feldman and Chiyuan Zhang. 2020. What Neural Networks Memorize and Why: Discovering the Long Tail via Influence Estimation. *Advances in Neural Information Processing Systems* 33 (2020), 2881–2891.

[10] Amirata Ghorbani and James Y. Zou. 2019. Data Shapley: Equitable Valuation of Data for Machine Learning. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2242–2251.

[11] László Györfi, Michael Kohler, Adam Krzyżak, and Harro Walk. 2002. *A distribution-free theory of nonparametric regression*. Springer.

[12] Zayd Hammoudeh and Daniel Lowd. 2024. Training data influence analysis and estimation: a survey. *Mach. Learn.* 113, 5 (2024), 2351–2403.

[13] Moritz Hardt, Ben Recht, and Yoram Singer. 2016. Train faster, generalize better: Stability of stochastic gradient descent. In *International conference on machine learning*. PMLR, 1225–1234.

[14] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. 2022. Datamodels: Understanding Predictions with Data and Data with Predictions. In *Proceedings of the 39th International Conference on Machine Learning*. PMLR, 9525–9587.

[15] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gürel, Bo Li, Ce Zhang, Costas J. Spanos, and Dawn Song. 2019. Efficient Task-Specific Data Valuation for Nearest Neighbor Algorithms. *Proc. VLDB Endow.* 12, 11 (2019), 1610–1623.

[16] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. 2019. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 1167–1176.

[17] Kevin Fu Jiang, Weixin Liang, James Y. Zou, and Yongchan Kwon. 2023. OpenDataVal: a Unified Benchmark for Data Valuation. *Advances in Neural Information Processing Systems* 36 (2023), 28624–28647.

[18] Hoang Anh Just, Feiyang Kang, Tianhao Wang, Yi Zeng, Myeongseob Ko, Ming Jin, and Ruoxi Jia. 2023. LAVA: Data Valuation without Pre-Specified Learning Algorithms. In *The Eleventh International Conference on Learning Representations*. OpenReview.

[19] Adam Karczmarz, Tomasz Michalak, Anish Mukherjee, Piotr Sankowski, and Piotr Wygocki. 2022. Improved feature importance computation for tree models based on the Banzhaf value. In *Uncertainty in Artificial Intelligence*. PMLR, 969–979.

[20] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. 2023. The UCI Machine Learning Repository. https://archive.ics.uci.edu/

[21] Pang Wei Koh and Percy Liang. 2017. Understanding Black-box Predictions via Influence Functions. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 1885–1894.

[22] Yongchan Kwon and James Zou. 2022. Beta Shapley: a Unified and Noise-reduced Data Valuation Framework for Machine Learning. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*. PMLR, 8780–8802.

[23] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating Training Data Makes Language Models Better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 8424–8445.

[24] Weida Li and Yaoliang Yu. 2023. Robust data valuation with weighted banzhaf values. *Advances in Neural Information Processing Systems* 36 (2023), 60349–60383.

[25] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. 2020. From local explanations to global understanding with explainable AI for trees. *Nature machine intelligence* 2, 1 (2020), 56–67.

[26] Sasan Maleki, Long Tran-Thanh, Greg Hines, Talal Rahwan, and Alex Rogers. 2013. Bounding the Estimation Error of Sampling-based Shapley Value Approximation. arXiv:1306.4265 [cs.GT]

[27] Tomomi Matsui and Yasuko Matsui. 2000. A survey of algorithms for calculating power indices of weighted majority games. *Journal of the Operations Research Society of Japan* 43, 1 (2000), 71–86.

[28] Rory Mitchell, Joshua Cooper, Eibe Frank, and Geoffrey Holmes. 2022. Sampling permutations for shapley value estimation. *Journal of Machine Learning Research* 23, 43 (2022), 1–46.

[29] Kislaya Prasad and Jerry S Kelly. 1990. NP-completeness of some problems concerning voting games. *International Journal of Game Theory* 19 (1990), 1–9.

[30] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating Training Data Influence by Tracing Gradient Descent. *Advances in Neural Information Processing Systems* 33 (2020), 19920–19930.

[31] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. 2022. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistic Surveys* 16 (2022), 1–85.

[32] Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. 2022. Scaling Up Influence Functions. *Proceedings of the AAAI Conference on Artificial Intelligence* 36, 8 (2022), 8179–8186.

[33] Lloyd S. Shapley. 1953. A Value for n-Person Games. In *Contributions to the Theory of Games, Volume II*. Princeton University Press, Princeton, NJ, USA, 307–318.

[34] Rachael Hwee Ling Sim, Xinyi Xu, and Bryan Kian Hsiang Low. 2022. Data Valuation in Machine Learning: "Ingredients", Strategies, and Open Challenges. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*. International Joint Conferences on Artificial Intelligence Organization, 5607–5614.

[35] Takeaki Uno. 2012. Efficient computation of power indices for weighted majority games. In *Algorithms and Computation: 23rd International Symposium, ISAAC 2012, Taipei, Taiwan, December 19-21, 2012. Proceedings 23*. Springer, 679–689.

[36] Rene Van den Brink and Gerard Van der Laan. 1998. Axiomatizations of the normalized Banzhaf value and the Shapley value. *Social Choice and Welfare* 15, 4 (1998), 567–582.

[37] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. 2013. OpenML: networked science in machine learning. *SIGKDD Explorations* 15, 2 (2013), 49–60. https://doi.org/10.1145/2641190.2641198

[38] Diego Varela and Javie Prado-Dominguez. 2012. Negotiating the Lisbon Treaty: Redistribution, Efficiency and Power Indices. *AUCO Czech Economic Review* 6, 2 (2012).

[39] Jiachen T. Wang and Ruoxi Jia. 2023. Data Banzhaf: A Robust Data Valuation Framework for Machine Learning. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*. PMLR, 6388–6421.

[40] Jiachen T. Wang and Ruoxi Jia. 2023. A Note on "Efficient Task-Specific Data Valuation for Nearest Neighbor Algorithms". arXiv:2304.04258 [stat.ML]

[41] Jiachen T. Wang, Prateek Mittal, and Ruoxi Jia. 2024. Efficient Data Shapley for Weighted Nearest Neighbor Algorithms. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*. PMLR, 2557–2565.

[42] Jiachen T Wang, Prateek Mittal, Dawn Song, and Ruoxi Jia. 2025. Data Shapley in One Training Run. In *The Thirteenth International Conference on Learning Representations*. OpenReview.net.

[43] Tom Yan and Ariel D Procaccia. 2021. If You Like Shapley Then You'll Love the Core. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 6 (2021), 5751–5759.

[44] Guangyi Zhang, Lutz Oettershagen, Lixu Wang, and Aristides Gionis. 2026. Efficient Banzhaf-Based Data Valuation for k-Nearest Neighbors Classification. *arXiv preprint arXiv:2605.21033* (2026).

[45] Jiayao Zhang, Qiheng Sun, Jinfei Liu, Li Xiong, Jian Pei, and Kui Ren. 2023. Efficient sampling approaches to shapley value approximation. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–24.