Efficient Data Valuation for Weighted Nearest Neighbor Algorithms

Anonymous Author(s)
Affiliation
Address
email

Abstract

Data Shapley is a principled way to assess the importance of individual training data sources for machine learning (ML) applications. However, it often comes with computational challenges in calculating exact Data Shapley scores. KNN-Shapley [7], which assigns data value leveraging the efficiently computable Data Shapley score of K nearest neighbors (KNN), has gained popularity as a viable alternative due to its computationally efficient nature. However, [7] only gives a practical algorithm for computing Data Shapley for unweighted KNN, but weighted KNN is more prevalently used in practice.

This work addresses the computational challenges of calculating the exact Data Shapley for weighted KNN classifiers (WKNN-Shapley). By making small adjustments to KNN configurations, we recast the computation of WKNN-Shapley into a counting problem and introduce an $O(K^2N^2)$ algorithm, presenting a notable improvement from the naive, impractical $O(N^K)$ algorithm. We also develop a deterministic approximation algorithm that further improves computational efficiency while maintaining the key fairness properties of the Shapley value. These advancements position WKNN-Shapley as a compelling alternative to KNN-Shapley. In particular, WKNN-Shapley can select high-quality data points and improve the performance of retrieval-augmented language models.

1 Introduction

Data is the backbone of machine learning (ML) models, but not all data is created equally. In real-world scenarios, data often carries noise and bias, sourced from diverse origins and data collection and labeling processes [19]. Against this backdrop, data valuation emerges as a growing research field, aiming to quantify the impact of individual data sources on ML training. Data valuation techniques are critical in explainable ML to diagnose influential training instances and in data marketplaces for fair compensation. The importance of data valuation is highlighted by legislative efforts such as the DASHBOARD Act of 2019 [30], which mandates companies to provide users with an estimate of their data's economic value. Moreover, the vision statements from leading companies like OpenAI underscore the importance of distributing AI benefits equitably [20].

The Shapley value for Data Valuation. Drawing on cooperative game theory, the technique of using the Shapley value for data valuation was pioneered by [5, 8]. The Shapley value is a renowned solution concept in game theory for fair profit attribution [22]. In the context of data valuation, individual data points or sources are regarded as "players" in a cooperative game, and *Data Shapley* refers to the suite of data valuation techniques that use the Shapley value as the contribution measure for each data owner. Numerous follow-up works of Data Shapley have been conducted [7, 4, 29, 1, 13, 17, 31, 9, 25, 28], underscoring its effectiveness in quantifying the impact of individual data sources on model performance.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

Data Shapley for Unweighted KNN. Despite offering a rigorous approach to data valuation with a solid theoretical foundation, the exact calculation of the Shapley value has the time complexity of $O(2^N)$ where N refers to the number of players (i.e., the number of data points/sources in the context of ML). While various Monte Carlo-based approximation algorithms for Data Shapley have been proposed (e.g., [8, 18, 17]), these approaches still require substantial computational resources due to model retraining. Fortunately, a breakthrough by [7] showed that computing the *exact* Data Shapley for unweighted K-Nearest Neighbors (KNN), one of the oldest yet still popular ML algorithms, is surprisingly easy and efficient. *KNN-Shapley* refers to the technique of quantifying data value based on KNN's Data Shapley score. Here, KNN can be regarded as a proxy model for the original complicated learning algorithm. KNN-Shapley can be applied to large, high-dimensional datasets by calculating the value scores on the features extracted from neural network embeddings. Due to its superior computational efficiency and adeptness at discerning data quality, KNN-Shapley is currently recognized as one of the most practical data valuation techniques [21], and it has found applications across various ML domains [6, 23, 15, 14, 2].

Question left from [7]: efficient computation of weighted KNN-Shapely. The insightful work of [7] introduced a highly efficient $O(N \log N)$ algorithm to compute the exact Data Shapley for unweighted KNN classifiers. However, while they also demonstrated that the exact Data Shapley for *weighted* KNN classifiers can be computed in polynomial time, the associated algorithm proposed in their work has the time complexity surges to $O(N^K)$—considerably larger that of its unweighted counterpart, and is impractical for actual implementation. Closing this efficiency gap is important, especially given the inherent advantages and wider application of weighted KNN. Compared with the unweighted counterpart, weighted KNN takes into account the distances between data points, attributing different importance levels to neighbors based on proximity. This makes weighted KNN provide significantly better performance while maintaining model interpretability. For instance, weighted KNN is being used in critical domains like healthcare [32] and anomaly detection [16], where both the performance and interpretability of the adopted ML model are important. Recent research has also highlighted weighted KNN's capability to improve language model's performance [11]. Given the broader use cases and advantages of weighted KNN in real-world applications, it is important to develop more efficient algorithms for the computation of WKNN-Shapley.

Settings of Weighted KNN Considered in this Work. Our preliminary investigations indicate that improving the computational efficiency of WKNN-Shapley for soft-label KNN classifiers with continuous weight values (the setting considered in [7]), poses considerable challenges. Consequently, we make necessary modifications to the specific KNN classifiers' configuration and shift our focus to hard-label KNN classifiers with discrete weight values. The justification for these changes and their practical relevance is elaborated in Section 3.2. Furthermore, we emphasize that small adjustments to the underlying KNN's configuration are crucial for the development of new data valuation techniques with desired properties. For instance, [28] considers a simple variant termed *Threshold KNN* and develops an alternative of KNN-Shapley that is privacy-friendly and more computationally efficient.

Technical Overview. (1) Binary Classification Setting. Given the configurations of the weighted KNN we described, we develop an algorithm with a quadratic runtime for the exact computation of WKNN-Shapley using dynamic programming. To further improve the computational efficiency, we propose a deterministic approximation algorithm (not based on Monte Carlo), which retains the crucial fairness properties of the original Shapley value (i.e., Symmetry and Null player axiom). **(2) Multi-class Classification Setting.** Directly adapting our WKNN-Shapley computation technique from binary to multi-class classifiers can significantly increase the overall time complexity. Instead, we present an alternative utility function for measuring the performance of WKNN classifiers. The Data Shapley calculation for the proposed utility function can be conveniently reduced to the WKNN-Shapley computation for binary classifiers, thanks to the linearity axiom of the Shapley value. Noteworthily, this approach outperforms its binary classification counterpart in efficiency for balanced datasets.

We showcase the application of WKNN-Shapley in selecting high-quality data points, and in particular it can be used for improving the performance of retrieval-augmented language models. In summary, our findings indicate that with minor adjustments to the KNN configurations, WKNN-Shapley can achieve significant computational efficiency. This makes WKNN-Shapley a viable and effective alternative to the original KNN-Shapley, marking a pivotal advancement in the realm of data valuation.

2 Preliminaries

We review the problem of data valuation for ML, and revisit the techniques of Data Shapley and KNN-Shapley.

Setup & Goal. Given a labeled dataset $D := \{z_i\}_{i=1}^N$ where each data point $z_i := (x_i, y_i)$, data valuation aims to assign a score to each training data point z_i, reflecting its importance for the trained ML model's performance. Formally, we seek a score vector $(\phi_{z_i})_{i=1}^N$ where each ϕ_{z_i} denotes the value of the data point z_i.

2.1 Data Shapley

The Shapley value (SV) [22] originates from game theory and is used to fairly attribute the total profit among all participated players. We first introduce the concept of *utility function*, and then state the definition of the Shapley value.

Utility Function. The Shapley value is defined based on the concept of *utility function*, which maps an input dataset to a score indicating the utility of the dataset for model training. Often, this function is chosen as the validation accuracy of a model trained on the given dataset. That is, given a training set S, the utility function $v(S) := \texttt{ValAcc}(\mathcal{A}(S))$, where $\mathcal{A}$ represents a learning algorithm that trains a model on dataset S, and $\texttt{ValAcc}(\cdot)$ is a function assessing the model's performance, such as its accuracy on a validation set.

Definition 1 (Shapley value [22])**.** *Given a utility function $v(\cdot)$ and a training set D of size N, the Shapley value of a data point $z \in D$ is defined as*

$$\phi_z(v) := \frac{1}{N} \sum_{k=1}^{N} \binom{N-1}{k-1}^{-1} \sum_{S \subseteq D_{-z}, |S|=k-1} [v(S \cup \{z\}) - v(S)] \tag{1}$$

In simple terms, the Shapley value is a weighted average of the utility changes when the point is added to different subsets of the training set. For notation simplicity, when the context is clear, we omit the utility function and simply write ϕ_z. The popularity of the Shapley value is attributable to the fact that it is the *unique* data value notion satisfying four axioms: Dummy player, Symmetry, Linearity, and Efficiency. We refer the readers to [5, 8] and the references therein for a detailed discussion about the interpretation and necessity of the four axioms in the ML context.

2.2 KNN-Shapley

A well-known disadvantage of the Shapley value is that its computation can be infeasible in general, as it requires evaluating $v(S)$ for all possible subsets $S \subseteq D$. A surprising result in [7, 26] showed that for unweighted KNN classifier, there exists a highly efficient algorithm for computing its exact Data Shapley score. Specifically, [7] considers the utility function for unweighted, soft-label KNN on a validation point $z^{(\mathrm{val})}$:

$$v(S; z^{(\mathrm{val})}) := \frac{\sum_{j=1}^{\min(K,|S|)} \mathbb{1}[y_{\alpha_{x^{(\mathrm{val})}}^{(S,j)}} = y^{(\mathrm{val})}]}{\min(|S|, K)} \tag{2}$$

where $\alpha_{x^{(\mathrm{val})}}^{(S,j)}$ denotes the index (among D) of jth closest data point in S to $x^{(\mathrm{val})}$. The main result in [7] shows that we can compute the *exact* Shapley value $\phi_{z_i}\left(v(\cdot; z^{(\mathrm{val})})\right)$ for *all* $z_i \in D$ by using a recursive formula within a total runtime of $O(N \log N)$. After computing the Shapley value $\phi_{z_i}\left(v(\cdot; z^{(\mathrm{val})})\right)$ for each $z^{(\mathrm{val})} \in D^{(\mathrm{val})}$, one can compute the Shapley value corresponding to the utility function on the full validation set $v(S; D^{(\mathrm{val})}) := \sum_{z^{(\mathrm{val})} \in D^{(\mathrm{val})}} v(S; z^{(\mathrm{val})})$ by simply taking the sum $\phi_{z_i}\left(v(\cdot; D^{(\mathrm{val})})\right) = \sum_{z^{(\mathrm{val})} \in D^{(\mathrm{val})}} \phi_{z_i}\left(v(\cdot; z^{(\mathrm{val})})\right)$ due to the linearity property of the Shapley value.

Remark 1. *Following the previous literature [7, 28], when we talk about runtime complexity of KNN-Shapley, we refer to the total runtime required to compute* all *data value scores $(\phi_{z_1}, \ldots, \phi_{z_N})$, as in practice a typical objective is to compute the data value scores for all data points within the training set. Moreover, we state the runtime with respect to a single validation point $z^{(\mathrm{val})}$, and the overall runtime can be obtained by multiplying by the size of $D^{(\mathrm{val})}$.*

Since its introduction, KNN-Shapley has quickly become a popular technique for data valuation due to its efficiency and effectiveness in assessing data quality. KNN-Shapley has been applied across various machine learning domains [6, 15, 14, 2]. Notably, recent studies have advocated it as *"the most practical data valuation technique capable of handling large-scale data effectively"* [21, 9].

3 Baseline Algorithms & Challenges

For unweighted KNN classifiers, [7] develops an efficient $O(N \log N)$ algorithm to calculate the exact Data Shapley. However, when it comes to weighted KNN classifiers, the proposed method has a time complexity of $O(N^K)$. While it is still in polynomial time when K is considered a constant, the runtime can be prohibitively large for practical use even when K is very small (e.g., 5). In this section, we provide a brief review of the high-level idea of the baseline algorithm from [7] and discuss the challenges in improving its computational efficiency.

3.1 Baseline Algorithm for Computing Data Shapley for Weighted KNN Classifiers

Given a validation data point $z^{(\mathrm{val})} = (x^{(\mathrm{val})}, y^{(\mathrm{val})})$ and a distance metric $d(\cdot,\cdot)$, we sort the training set $D = \{z_i = (x_i, y_i)\}_{i=1}^N$ according to their distance to the validation point $d(x_i, x^{(\mathrm{val})})$ in non-descending order. Throughout the entire paper, we assume that $d(x_i, x^{(\mathrm{val})}) \le d(x_j, x^{(\mathrm{val})})$ for any $i \le j$ unless otherwise specified. **Weight of each data point:** in weighted KNN, each data point z_i is associated with a weight $w_i := \omega_{x^{(\mathrm{val})}}(x_i)$. Such a weight is usually determined based on the distance between x_i and the queried example $x^{(\mathrm{val})}$. For example, a popular choice of the weight function is the RBF kernel $\omega_{x^{(\mathrm{val})}}(x_i) = \exp(-d(x_i, x^{(\mathrm{val})}))$. Without loss of generality, in this paper we assume $w_i \in [0,1]$.

Baseline $O(N^K)$ algorithm from [7]. [7] considers weighted, soft-label KNN with the following utility function:

$$v(S; z^{(\mathrm{val})}) := \frac{\sum_{j=1}^{\min(K,|S|)} w_{\alpha_{x^{(\mathrm{val})}}^{(S,j)}} \mathbb{1}\left[y_{\alpha_{x^{(\mathrm{val})}}^{(S,j)}} = y^{(\mathrm{val})} \right]}{\sum_{j=1}^{\min(K,|S|)} w_{\alpha_{x^{(\mathrm{val})}}^{(S,j)}}} \tag{3}$$

The intuition of the $O(N^K)$ algorithm for computing the exact Data Shapley for this utility function developed in [7] is as follows: from Definition 1, the Shapley value for z_i is a weighted average of the *marginal contribution (MC)* $v(S \cup \{z_i\}) - v(S)$; hence, we only need to study those S whose utility might change due to the inclusion of z_i. In the context of KNN, those are the subsets S where z_i is within the K nearest neighbors of $x^{(\mathrm{val})}$ after being added into S. It is critical to notice that the utility of any dataset only depends on the K nearest neighbors of $x^{(\mathrm{val})}$ in S. Given that there are only $\sum_{j=0}^K \binom{N}{j}$ unique subsets of size $\le K$, we can simply query the value of the MC $v(S \cup \{z_i\}) - v(S)$ for all S of size $\le K$. For any larger S, the value of MC must be the same as its subset of K nearest neighbors. We can then compute the Shapley value as a weighted average of these MC values by counting the number of subsets that share the same MC values through simple combinatorial analysis. Such an algorithm results in the runtime of $\sum_{j=0}^K \binom{N}{j} = O(N^K)$. See Section 4 in [7] for algorithm details.

3.2 Challenges & Solutions

We point out the major challenges associated with directly improving the computational efficiency for the problem setup considered in [7], and propose small but effective changes that enable more efficient algorithms for computing WKNN-Shapley.

Challenge #1: weights normalization term. The key behind the $O(N \log N)$ algorithm for unweighted KNN-Shapley from [7] is that, the values of MC are the same for many different Ss even when $|S| \le K$. That is, for unweighted, soft-label KNN with utility function in (2), if z_i is within the K nearest neighbors of $x^{(\mathrm{val})}$ among $S \cup \{z_i\}$, we have $v(S \cup \{z_i\}) - v(S) = \frac{1}{K}\left(\mathbb{1}[y_i = y^{(\mathrm{val})}] - \mathbb{1}[y_{\alpha_{x^{(\mathrm{val})}}(S,K)} = y^{(\mathrm{val})}] \right)$. Hence, we can just count the number of subsets $S \subseteq D \setminus \{z_i\}$ where z_i is within the K nearest neighbors of $x^{(\mathrm{val})}$ among $S \cup \{z_i\}$, and have the same Kth nearest neighbor to $z^{(\mathrm{val})}$. In this way, one can avoid the burden of evaluating $v(S)$ for all $S \subseteq D$. However, for the utility function in (3), for each $|S| \leq K$, there is little chance that $v(S \cup \{z_i\}) - v(S)$ can have the same value due to the weights normalization term. Therefore, in this work, we instead consider the utility function for weighted *hard-label* KNN classifier. "Hard-label" refers to the classifiers that output the predicted class instead of the confidence scores (see the details in Section A). Hard-label KNN is arguably used more frequently in practice. More importantly, its prediction only depends on the weight comparison between different classes, and hence its utility function does not have a normalization term. **Challenge #2: continuous weights.** If the weights are on the continuous space, there are infinitely many possibilities of voting results of the K nearest neighbors. Similar to the issue caused by the weights normalization term, this also makes it difficult for any S_1, S_2 of size $\leq K$ to share the same MC value. Therefore, we consider a more tractable setting where the weights lie in a discrete space. Such a change is reasonable since the weights are stored in terms of finite bits and hence it is also in the discrete space in practice. Moreover, rounding is a deterministic operation and does not change the ranking of the original weights. Hence, the Shapley value computed based on the discrete weights has the same ranking order compared with the Shapley value computed on the continuous weights (it might create ties but will not reverse the original order). While it is difficult to derive the exact error in the computed Shapley value due to discretization, we empirically verified in Appendix D.2 that the discretization does not cause a large error in the final Shapley value.

4 Data Shapley for Weighted KNN Classifiers (Overview)

In this section, we provide a high-level overview of the efficient algorithms for computing and approximating the Data Shapley for discrete weighted, hard-label KNN classifiers. The detailed but notation-heavy descriptions are deferred to Appendix A (for binary setting) and B (for multi-class setting).

Utility Function for Weighted Hard-Label KNN Classifiers. The utility function of weighted hard-label KNN can be written as

$$v(S; z^{(\mathrm{val})}) = \mathbb{1}\left[y^{(\mathrm{val})} \in \operatorname*{argmax}_{c \in \mathbf{C}} \sum_{j=1}^{\min(K, |S|)} w_{\alpha_{x^{(\mathrm{val})}}^{(S,j)}} \mathbb{1}[y_{\alpha_{x^{(\mathrm{val})}}^{(S,j)}} = c] \right]$$

where $\mathbf{C} = \{1, \ldots, C\}$ is the space of classes, and C is the number of classes[1]. We omit the input of $z^{(\mathrm{val})}$ and simply write $v(S)$ when the validation point is clear from the context.

High-level Idea for Exact Data Shapley Calculation. For simplicity, we focus on the techniques for binary classification setting here. For binary classification, by taking $\widetilde{w}_j := (2\mathbb{1}[y^{(\mathrm{val})} = y_j] - 1) w_j$, we can rewrite the utility function in a more compact way: $v(S) = \mathbb{1}\left[\sum_{j=1}^{\min(K,|S|)} \widetilde{w}_{\alpha_{x^{(\mathrm{val})}}^{(S,j)}} \geq 0\right]$. Given that the Shapley value is a weighted average of the marginal contribution $v(S \cup \{z_i\}) - v(S)$, we first study the expression of $v(S \cup \{z_i\}) - v(S)$ for a fixed subset $S \subseteq D \setminus \{z_i\}$ with such a utility function (see Theorem 2 in Appendix A). Since $v(S \cup \{z_i\}) - v(S) \in \{\pm 1, 0\}$, from the formula of the Shapley value (Definition 1), we can reframe the problem of computing the Shapley value for a weighted, hard-label KNN-Shapley as a counting problem for the number of S of certain sizes such that $v(S \cup \{z_i\}) - v(S) = 1$ (or -1), and then take the weighted average of the counts for different sizes (Theorem 4). We can then solve this counting problem through dynamic programming, and we discover mathematical short-cuts (Theorem 8) to further improve the computational efficiency of WKNN-Shapley to $O(K^2 N^2)$ (Theorem 9).

Deterministic approximation. If we only require an approximation of the Shapley value, we show that we can further speed up the Shapley value calculation. We derive a deterministic approximation algorithm by skipping the counting for those Ss with certain conditions where we believe $v(S \cup \{z_i\}) - v(S) = 0$. We derive the error bound for such an approximation (Theorem 11). We note that our approximation algorithm preserves the important Symmetry and Null player axioms for the Shapley value. On the contrary, the prevalent Monte Carlo-based approximation techniques give randomized solutions and arguably muddy the clarity of Shapley value's axioms.

[1]For the case of multiple classes having the same top counts, we assume the utility is 1 as long as $y^{(\mathrm{val})}$ is among the majority classes.

5 Applications of WKNN-Shapley

With our efficient algorithms to compute or approximate Data Shapley for weighted KNN classifiers, WKNN-Shapley now stands as another practical data valuation method. In this section, we showcase WKNN-Shapley's potential in identifying high-quality data points for weighted KNN. This selection method can be further used for improving the performance of K nearest neighbor language models (KNN-LMs) [11], a famous type of retrieval-augmented language model nowadays. Figure 1 (a) shows WKNN's performance on CIFAR10 [12] when trained on data points that receive the highest data value scores (computed based on the associated data valuation techniques). Evidently, both the exact and approximated WKNN-Shapley offer comparable results. Remarkably, the approximation algorithm achieves this while being 5 times faster than its exact counterpart. Additionally, both of them outperform the original KNN-Shapley considerably. Figure 1 (b) shows KNN-LM's performance on the WNLI dataset [24], where the data store incorporates only those data points that receive the highest value scores. Again, both the exact and approximated WKNN-Shapley stand out and outperforms the original unweighted KNN-Shapley by a large margin. We note that when leveraging $> 55\%$ of the entire data store, KNN-LM performs even worse than the original, unaugmented LM due to the relatively low quality of the benchmark dataset. This underscores the important role of selecting high-quality data points, where WKNN-Shapley proves to be an effective tool.

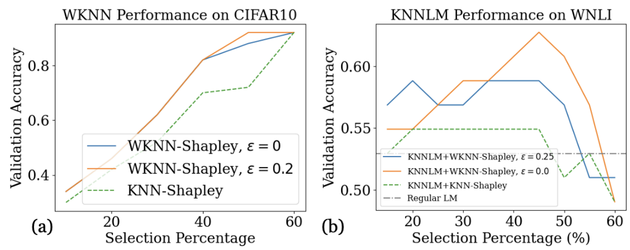

Figure 1: (a) The performance of WKNN on the CIFAR10 subset selected by different data valuation techniques. We set $K = 25$ for all methods here. (b) The performance of KNN-LM on the WNLI dataset [24]'s subset selected by different data valuation techniques. We set $K = 25$ for all methods here. KNN-LM is a popular retrieval-augmented language model where the output of the original LM is being interpolated with the output of the KNN classifiers, i.e., $p_{KNN-LM}(y) := \lambda p_{KNN}(y) + (1 - \lambda)p_{LM}(y)$. Here, we set $\lambda = 0.5$. We use BERT [10] as the language model here.

6 Conclusion

In this work, we tackle the problem of computing and approximating Data Shapley for weighted KNN classifiers. We first identify the challenges of directly improving the computational efficiency for the utility function of weighted soft-label KNN with continuous weights. Instead, we consider weighted hard-label KNN with discretized weights, where we derive an $O(K^2N^2)$ algorithm for computing the exact Data Shapley. We demonstrate the applications of WKNN-Shapley on data selection for retrieval-augmented language models.

Future works: Characterizing the class of learning algorithms whose Data Shapley can be computed in polynomial-time. The popularity of KNN-Shapley lies in its computational efficiency. The polynomial-time algorithm for computing the exact Data Shapley for KNN is a surprising result since the Shapley value requires exponential time to compute for general utility functions. KNN-Shapley outperforms Monte Carlo-based approximation for the original Data Shapley due to its deterministic nature [25]. It is interesting to consider whether there exist other learning algorithms whose Data Shapley can be computed in polynomial time, and whether we can characterize the properties of those "Shapley-friendly" learning algorithms.

References

[1] Yatao Bian, Yu Rong, Tingyang Xu, Jiaxiang Wu, Andreas Krause, and Junzhou Huang. Energy-based learning for cooperative games, with applications to valuation problems in machine learning. *arXiv preprint arXiv:2106.02938*, 2021.

[2] Christie Courtnage and Evgueni Smirnov. Shapley-value data valuation for semi-supervised learning. In *Discovery Science: 24th International Conference, DS 2021, Halifax, NS, Canada, October 11–13, 2021, Proceedings 24*, pages 94–108. Springer, 2021.

[3] Andrea Dal Pozzolo, Olivier Caelen, Reid A Johnson, and Gianluca Bontempi. Calibrating probability with undersampling for unbalanced classification. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 159–166. IEEE, 2015.

[4] Amirata Ghorbani, Michael Kim, and James Zou. A distributional framework for data valuation. In *International Conference on Machine Learning*, pages 3535–3544. PMLR, 2020.

[5] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pages 2242–2251. PMLR, 2019.

[6] Amirata Ghorbani, James Zou, and Andre Esteva. Data shapley valuation for efficient batch active learning. In *2022 56th Asilomar Conference on Signals, Systems, and Computers*, pages 1456–1462. IEEE, 2022.

[7] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas J Spanos, and Dawn Song. Efficient task-specific data valuation for nearest neighbor algorithms. *Proceedings of the VLDB Endowment*, 2019.

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

[9] Bojan Karlaš, David Dao, Matteo Interlandi, Bo Li, Sebastian Schelter, Wentao Wu, and Ce Zhang. Data debugging with shapley importance over end-to-end machine learning pipelines. *arXiv preprint arXiv:2204.11131*, 2022.

[10] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[11] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*, 2019.

[12] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[13] Yongchan Kwon and James Zou. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 8780–8802. PMLR, 2022.

[14] Weixin Liang, Kai-Hui Liang, and Zhou Yu. Herald: An annotation efficient method to detect user disengagement in social conversations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3652–3665, 2021.

[15] Weixin Liang, James Zou, and Zhou Yu. Beyond user self-reported likert scale ratings: A comparison model for automatic dialog evaluation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1363–1374, 2020.

[16] Yihua Liao and V Rao Vemuri. Use of k-nearest neighbor classifier for intrusion detection. *Computers & security*, 21(5):439–448, 2002.

[17] Jinkun Lin, Anqi Zhang, Mathias Lécuyer, Jinyang Li, Aurojit Panda, and Siddhartha Sen. Measuring the effect of training data on deep learning predictions via randomized experiments. In *International Conference on Machine Learning*, pages 13468–13504. PMLR, 2022.

[18] Rory Mitchell, Joshua Cooper, Eibe Frank, and Geoffrey Holmes. Sampling permutations for shapley value estimation. 2022.

[19] Curtis G Northcutt, Anish Athalye, and Jonas Mueller. Pervasive label errors in test sets destabilize machine learning benchmarks. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[20] OpenAI. Planning for agi and beyond. `https://openai.com/blog/planning-for-agi-and-beyond`, 2023.

[21] Konstantin D Pandl, Fabian Feiland, Scott Thiebes, and Ali Sunyaev. Trustworthy machine learning for health care: scalable data valuation with the shapley value. In *Proceedings of the Conference on Health, Inference, and Learning*, pages 47–57, 2021.

[22] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[23] Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9630–9638, 2021.

[24] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2018.

[25] Jiachen T Wang and Ruoxi Jia. Data banzhaf: A robust data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 6388–6421. PMLR, 2023.

[26] Jiachen T Wang and Ruoxi Jia. A note on" efficient task-specific data valuation for nearest neighbor algorithms". *arXiv preprint arXiv:2304.04258*, 2023.

[27] Jiachen T Wang and Ruoxi Jia. A note on" towards efficient data valuation based on the shapley value”. *arXiv preprint arXiv:2302.11431*, 2023.

[28] Jiachen T Wang, Yuqing Zhu, Yu-Xiang Wang, Ruoxi Jia, and Prateek Mittal. Threshold knn-shapley: A linear-time and privacy-friendly approach to data valuation. *arXiv preprint arXiv:2308.15709*, 2023.

[29] Tianhao Wang, Johannes Rausch, Ce Zhang, Ruoxi Jia, and Dawn Song. A principled approach to data valuation for federated learning. In *Federated Learning*, pages 153–167. Springer, 2020.

[30] Mark Warner. Warner & hawley introduce bill to force social media companies to disclose how they are monetizing user data. Government Document, 2019.

[31] Zhaoxuan Wu, Yao Shu, and Bryan Kian Hsiang Low. Davinz: Data valuation using deep neural networks at initialization. In *International Conference on Machine Learning*, pages 24150–24176. PMLR, 2022.

[32] Wenchao Xing and Yilin Bei. Medical health big data classification based on knn classification algorithm. *IEEE Access*, 8:28808–28819, 2019.

[33] Tom Yan and Ariel D Procaccia. If you like shapley then you’ll love the core, 2020.

A Data Shapley for Weighted KNN Binary Classifiers

In this section, we use $\mathtt{V}$ to denote the discretized space of $[0,1]$, where we create 2^b equally spaced points within the interval when we use b bits for discretization. We denote $V := |\mathtt{V}| = 2^b$ the size of the weight space. Furthermore, we use $\mathtt{V}_{(K)}$ to denote the discretized space of $[0,K]$ (where we create $K2^b$ equally spaced points within the interval). We use $\mathtt{NB}_{x^{(\mathrm{val})},K}(S)$ to denote the set of data points that is within the K-nearest neighbors of $x^{(\mathrm{val})}$ among S.

Utility Function for Weighted Hard-Label KNN Classifiers. The utility function of weighted hard-label KNN can be written as

$$v(S; z^{(\mathrm{val})}) = \mathbb{1}\left[y^{(\mathrm{val})} \in \operatorname*{argmax}_{c \in \mathbf{C}} \sum_{j=1}^{\min(K,|S|)} w_{\alpha_{x^{(\mathrm{val})}}^{(S,j)}} \mathbb{1}[y_{\alpha_{x^{(\mathrm{val})}}^{(S,j)}} = c]\right] \tag{4}$$

where $\mathbf{C} = \{1, \ldots, C\}$ is the space of classes, and C is the number of classes[2]. We omit the input of $z^{(\mathrm{val})}$ and simply write $v(S)$ when the validation point is clear from the context. For binary classification, by taking $\widetilde{w}_j := (2\mathbb{1}[y^{(\mathrm{val})} = y_j] - 1)w_j$, we can rewrite the utility function in a more compact way:

$$v(S) = \mathbb{1}\left[\sum_{j=1}^{\min(K,|S|)} \widetilde{w}_{\alpha_{x^{(\mathrm{val})}}^{(S,j)}} \geq 0\right] \tag{5}$$

A.1 Exact Shapley value Calculation

A.1.1 Computing SV is a Counting Problem

Given that the Shapley value is a weighted average of the marginal contribution $v(S \cup \{z_i\}) - v(S)$, we first study the expression of $v(S \cup \{z_i\}) - v(S)$ for a fixed subset $S \subseteq D \setminus \{z_i\}$ with the utility function in (5).

Theorem 2. *For any data point $z_i \in D$ and any subset $S \subseteq D \setminus \{z_i\}$, the marginal contribution is*

$$v(S \cup \{z_i\}) - v(S)$$
$$= \begin{cases} 1 & z_i \in \mathtt{NB}_{x^{(\mathrm{val})},K}(S \cup \{z_i\}), y_i = y^{(\mathrm{val})}, \sum_{z_j \in S} \widetilde{w}_j \in [-\widetilde{w}_i, 0) \text{ if } |S| \leq K-1 \\ & \qquad \sum_{j=1}^{K-1} \widetilde{w}_{\alpha_{x^{(\mathrm{val})}}^{(S,j)}} \in \left[-w_i, -\widetilde{w}_{\alpha_{x^{(\mathrm{val})}}(S,K)}\right) \text{ if } |S| \geq K \\ -1 & z_i \in \mathtt{NB}_{x^{(\mathrm{val})},K}(S \cup \{z_i\}), y_i \neq y^{(\mathrm{val})}, \sum_{z_j \in S} \widetilde{w}_j \in [0, -\widetilde{w}_i) \text{ if } |S| \leq K-1 \\ & \qquad \sum_{j=1}^{K-1} \widetilde{w}_{\alpha_{x^{(\mathrm{val})}}^{(S,j)}} \in \left[-\widetilde{w}_{\alpha_{x^{(\mathrm{val})}}(S,K)}, \; -w_i\right) \text{ if } |S| \geq K \\ 0 & \text{Otherwise} \end{cases}$$

From Theorem 2 and the formula of the Shapley value (Definition 1), we can reframe the problem of computing the Shapley value for a weighted, hard-label KNN-Shapley. This involves counting the following quantity:

Definition 3. *Let $\mathtt{G}_{i,\ell}$ denote the count of subsets $S \subseteq D \setminus z_i$ of size ℓ that satisfy the conditions below:*

1. $x_i \in \mathtt{NB}_{x^{(\mathrm{val})},K}(S \cup \{z_i\})$.
2. *For $y_i = y^{(\mathrm{val})}$:*
   - *If $|S| = \ell \leq K-1$, then $\sum_{z_j \in S} \widetilde{w}_j \in [-\widetilde{w}_i, 0)$.*
   - *If $|S| = \ell \geq K$, then $\sum_{j=1}^{K-1} \widetilde{w}_{\alpha_{x^{(\mathrm{val})}}^{(S,j)}} \in \left[-w_i, -\widetilde{w}_{\alpha_{x^{(\mathrm{val})}}(S,K)}\right)$.*
3. *For $y_i \neq y^{(\mathrm{val})}$:*

[2] For the case of multiple classes having the same top counts, we assume the utility is 1 as long as $y^{(\mathrm{val})}$ is among the majority classes.

- *If* $|S| = \ell \le K-1$*, then* $\sum_{z_j \in S} \widetilde{w}_j \in [0, -\widetilde{w}_i)$.
- *If* $|S| = \ell \ge K$*, then* $\sum_{j=1}^{K-1} \widetilde{w}_{\alpha^{(S,j)}_{x^{(\text{val})}}} \in \left[-\widetilde{w}_{\alpha_{x^{(\text{val})}}(S,K)}, \ -w_i \right)$.

Theorem 4. *For a weighted, hard-label KNN binary classifier using the utility function given by (5), the Shapley value can be expressed as:*

$$\phi_{z_i} = \frac{2\mathbb{1}[y_i = y^{(\text{val})}] - 1}{N} \sum_{\ell=0}^{N-1} \binom{N-1}{\ell}^{-1} \mathtt{G}_{i,\ell} \tag{6}$$

Proof. This immediately follows from the definition of the Shapley value. □

A.1.2 Computing $\mathtt{G}_{i,\ell}$ via Dynamic Programming

In this section, we show how to compute $\mathtt{G}_{i,\ell}$ with dynamic programming techniques. Before diving into the algorithm, we first introduce an intermediary quantity that serves as the crux of our dynamic programming formulation.

Definition 5. *Let* $\mathtt{F}_i\,[m, \ell, s]$ *denote the count of subsets* $S \subseteq D \setminus \{z_i\}$ *of size* ℓ *that satisfy the conditions below:*

1. $x_i \in \mathtt{NB}_{x^{(\text{val})},K}(S \cup \{z_i\})$.
2. *Within* S*, the data point* z_m *is the* $\min(\ell, K)$*-th closest to the query example* $x^{(\text{val})}$.
3. $\sum_{j=1}^{\min(\ell,K-1)} \widetilde{w}_{\alpha^{(S,j)}_{x^{(\text{val})}}} = s$.

We can relate this auxiliary quantity to our desired $\mathtt{G}_{i,\ell}$ as follows:

Theorem 6 (Relation between $\mathtt{G}_{i,\ell}$ and $\mathtt{F}_i$)**.** *For* $y_i = y^{(\text{val})}$*, we can compute* $\mathtt{G}_{i,\ell}$ *from* $\mathtt{F}_i$ *as follows:*

$$\mathtt{G}_{i,\ell} = \begin{cases} \sum_{m \in [N] \setminus i} \sum_{s \in [-\widetilde{w}_i, 0)} \mathtt{F}_i\,[m, \ell, s] & \text{for } \ell \le K-1, \\ \sum_{m \in [N] \setminus i} \sum_{s \in [-\widetilde{w}_i, -\widetilde{w}_m)} \mathtt{F}_i\,[m, \ell, s] & \text{for } \ell \ge K. \end{cases}$$

For $y_i \neq y^{(\text{val})}$*, we have:*

$$\mathtt{G}_{i,\ell} = \begin{cases} \sum_{m \in [N] \setminus i} \sum_{s \in [0, -\widetilde{w}_i)} \mathtt{F}_i\,[m, \ell, s] & \text{for } \ell \le K-1, \\ \sum_{m \in [N] \setminus i} \sum_{s \in [-\widetilde{w}_m, -\widetilde{w}_i)} \mathtt{F}_i\,[m, \ell, s] & \text{for } \ell \ge K. \end{cases}$$

Proof. This follows immediately from the definition of $\mathtt{G}_{i,\ell}$. □

When $K > 1$,[3] it is easy to see that for $\ell = 1$ have

$$\mathtt{F}_i[m, 1, s] = \begin{cases} 1 & s = w_m \\ 0 & s \neq w_m \end{cases}$$

We can then compute $\mathtt{F}_i[m, \ell, s]$ for $\ell \ge 2$ with the following theorem:

Theorem 7. *We have the following recursive relation of* $\mathtt{F}_i[m, \ell, s]$.

1. ***Case of*** $\ell \le K-1$***:***

$$\mathtt{F}_i[m, \ell, s] = \sum_{t=1}^{m-1} \mathtt{F}_i[t, \ell-1, s-w_m] \tag{7}$$

2. ***Case of*** $\ell \ge K$***:***

[3] Since [7] has shown that weighted KNN-Shapley can be computed in $O(N^K)$ time complexity, we focus on the setting where $K > 1$.

(a) *When* $m < i$*:* $\mathtt{F}_i[m, \ell, s] = 0$.

(b) *When* $m > i$*:* $\mathtt{F}_i[m, \ell, s] = \sum_{t=1, t \neq i}^{m-1} \mathtt{F}_i[t, K-1, s] \binom{N-m}{\ell-K}$.

So far, it seems that a simple solution would be first use the recursive formula in (7) to compute $\mathtt{F}_i[\cdot, \ell, \cdot]$ for $\ell \leq K-1$, and then use the explicit formula in Theorem 7 to compute $\mathtt{F}_i[\cdot, \ell, \cdot]$ for $\ell \geq K$. This renders an $O(N^2 V)$ runtime to compute $\mathtt{F}_i[m, \ell, s]$ for all of $m = 1, \ldots, N, \ell = 1, \ldots, K, s \in \mathtt{V}$. However, it is possible to further improve the computational efficiency by circumventing explicit computations for $\mathtt{F}_i[\cdot, \ell, \cdot], \ell \geq K$. Specifically, after we compute $\mathtt{F}_i[m, K-1, s]$, there is in fact a short-cut formula to directly compute the summation of $\sum_{\ell=K}^{N-1} \frac{\mathtt{G}_{i,\ell}}{\binom{N-1}{\ell}}$.

Theorem 8. *For a weighted, hard-label KNN binary classifier using the utility function given by (5), the Shapley value can be expressed as:*

$$\phi_{z_i} = \frac{2\mathbb{1}[y_i = y^{(\mathrm{val})}] - 1}{N} \left(\sum_{\ell=0}^{K-1} \binom{N-1}{\ell}^{-1} \mathtt{G}_{i,\ell} + \sum_{m=\max(i+1, K+1)}^{N} \mathtt{R}_{i,m} \binom{m-1}{K}^{-1} \frac{N}{m} \right) \quad (8)$$

where $\mathtt{R}_{i,m} := \begin{cases} \sum_{t=1}^{m-1} \sum_{s \in [-\widetilde{w}_i, -\widetilde{w}_m)} \mathtt{F}_i[t, K-1, s] & \text{for } y_i = y^{(\mathrm{val})} \\ \sum_{t=1}^{m-1} \sum_{s \in [-\widetilde{w}_m, -\widetilde{w}_i)} \mathtt{F}_i[t, K-1, s] & \text{for } y_i \neq y^{(\mathrm{val})} \end{cases}.$

Based on the above findings, we develop Algorithm 1 for computing the exact Shapley value for weighted KNN binary classifier. While Algorithm 1 itself does not achieve the time complexity of $O(K^2 N^2)$, we note that the for-loops for computing $\mathtt{F}_i$ and $\mathtt{R}_{i,m}$ can be further optimized, and we show the version of pseudocode that optimize for the computational efficiency Algorithm 2 in Appendix C. Nevertheless, we put the more readable (but less efficient) version of the pseudocode here for the ease of reader's understanding.

Theorem 9. *Algorithm 2 (in Appendix C) computes the exact Shapley value and achieves* $O(K^2 N^2 V)$ *time complexity.*

Algorithm 1 Weighted KNN-Shapley for binary classification (reader-friendly version)

Input:
- K – hyperparameter of weighted KNN algorithm.
- $z^{(\text{val})} = (x^{(\text{val})}, y^{(\text{val})})$ – the validation point.
- $D = \{z_i = (x_i, y_i)\}_{i=1}^N$ – sorted training set where $d(x_i, x^{(\text{val})}) \leq d(x_j, x^{(\text{val})})$ for any $i \leq j$.
- $M^\star$ – hyperparameter for SV approximation (Section A.2). $M^\star = N$ for exact SV calculation.

Compute the weight $w_i = \omega_{x^{(\text{val})}}(x_i)$ for $i \in \{1, \ldots, N\}$.
$\widetilde{w}_j = (2\mathbb{1}[y^{(\text{val})} = y_j] - 1)w_j$ for $i \in \{1, \ldots, N\}$.

for $i \in \{1, \ldots, N\}$ **do**

  // Initialize $\mathtt{F}_i$
  Initialize $\mathtt{F}_i[m, \ell, s] = 0$ for $m \in \{1, \ldots, M^\star\}, \ell \in \{1, \ldots, K-1\}, s \in \mathtt{V}_{(K)}$.
  for $m \in \{1, \ldots, M^\star\} \setminus \{i\}$ **do**
    $\mathtt{F}_i[m, 1, \widetilde{w}_m] = 1$

  // Compute $\mathtt{F}_i$ (Runtime-optimized version in Appendix C)
  for $\ell \in \{2, \ldots, K-1\}$ **do**
    for $m \in \{\ell, \ldots, M^\star\} \setminus \{i\}$ **do**
      for $s \in \mathtt{V}_{(K)}$ **do**
        $\mathtt{F}_i[m, \ell, s] = \sum_{t=1}^{m-1} \mathtt{F}_i[t, \ell-1, s-\widetilde{w}_m]$

  // Compute $\mathtt{R}_{i,m}$ (Runtime-optimized version in Appendix C)
  for $m \in \{\max(i+1, K+1), \ldots, M^\star\}$ **do**
    $$\mathtt{R}_{i,m} = \begin{cases} \sum_{t=1}^{m-1} \sum_{s \in [-\widetilde{w}_i, -\widetilde{w}_m)} \mathtt{F}_i[t, K-1, s] & \text{for } y_i = y^{(\text{val})} \\ \sum_{t=1}^{m-1} \sum_{s \in [-\widetilde{w}_m, -\widetilde{w}_i)} \mathtt{F}_i[t, K-1, s] & \text{for } y_i \neq y^{(\text{val})} \end{cases}$$

  // Compute $\mathtt{G}_{i,\ell}$
  $\mathtt{G}_{i,0} = \mathbb{1}[w_i < 0]$.[a]
  for $\ell \in \{1, \ldots, K-1\}$ **do**
    $$\mathtt{G}_{i,\ell} = \begin{cases} \sum_{m \in [M^\star] \setminus i} \sum_{s \in [-\widetilde{w}_i, 0)} \mathtt{F}_i[m, \ell, s] & \text{for } y_i = y^{(\text{val})} \\ \sum_{m \in [M^\star] \setminus i} \sum_{s \in [0, -\widetilde{w}_i)} \mathtt{F}_i[m, \ell, s] & \text{for } y_i \neq y^{(\text{val})} \end{cases}$$

  // Compute the Shapley value for z_i
  $$\phi_{z_i} = \mathtt{sign}(w_i) \left[\frac{1}{N} \sum_{\ell=0}^{K-1} \frac{\mathtt{G}_{i,\ell}}{\binom{N-1}{\ell}} + \sum_{m=\max(i+1,K+1)}^{M^\star} \frac{\mathtt{R}_{i,m}}{m\binom{m-1}{K}} \right]$$ [b]

---

[a] Recall that we define $v(S) = \mathbb{1}\left[\sum_{j=1}^{\min(K,|S|)} \widetilde{w}_{(j)} \geq 0\right]$, hence $v(\{z_i\}) - v(\emptyset) \in \{-1, 0\}$ and is equal to -1 if and only if $w_i < 0$.

[b] $$\mathtt{sign}(w) = \begin{cases} 1 & w > 0 \\ 0 & w = 0 \\ -1 & w < 0 \end{cases}.$$

A.2 Deterministic Approximation for Weighted KNN-Shapley

The overall time complexity for computing exact WKNN-Shapley with Algorithm 2 is $O(K^2N^2)$. In this section, we show that if we only require an approximation of the Shapley value, we can significantly speed up the Shapley value calculation.

Intuition of approximation technique. From Theorem 7, we know that in order to compute $\mathtt{F}_i[m, \ell, s]$, we only need to know $\mathtt{F}_i[t, \ell-1, s]$ with $t \leq m-1$. Moreover, observe that the building blocks for $\mathtt{G}_{i,\ell}$ (or $\mathtt{R}_{i,m}$), $\sum_{s\in[-\widetilde{w}_i, 0)} \mathtt{F}_i[t, \ell, s]$ (or $\sum_{s\in[-\widetilde{w}_i, -\widetilde{w}_m)} \mathtt{F}_i[t, K-1, s]$), can be considerably smaller than their counterpart that takes the summation over the entire range of $\mathtt{V}$. Hence, we can use $\widehat{\mathtt{F}}_i[m, \ell, s] = 0$ as an approximation for $\mathtt{F}_i[m, \ell, s]$ for all $m \geq M^\star + 1$ with some prespecified threshold $M^\star$. Similarly, we can use $\widehat{\mathtt{R}}_{i,m} = 0$ as an approximation for $\mathtt{R}_{i,m}$ for all $m \geq M^\star + 1$. The resultant simple approximation for the Shapley value ϕ_{z_i} is stated as follows:

Definition 10. *We define the approximation $\widehat{\phi}_{z_i}^{(M^\star)}$ as*

$$\widehat{\phi}_{z_i}^{(M^\star)} := \mathtt{sign}(w_i) \left[\frac{1}{N} \sum_{\ell=0}^{K-1} \frac{\widetilde{\mathtt{G}}_{i,\ell}}{\binom{N-1}{\ell}} + \sum_{m=\max(i+1, K+1)}^{M^\star} \mathtt{R}_{i,m} \left(\frac{1}{m}\right) \binom{m-1}{K}^{-1} \right] \tag{9}$$

where $\widetilde{\mathtt{G}}_{i,\ell} := \begin{cases} \sum_{m=1}^{M^\star} \sum_{s\in[-\widetilde{w}_i, 0)} \mathtt{F}_i[m, \ell, s] & \text{for } y_i = y^{(\mathrm{val})} \\ \sum_{m=1}^{M^\star} \sum_{s\in[0, -\widetilde{w}_i)} \mathtt{F}_i[m, \ell, s] & \text{for } y_i \neq y^{(\mathrm{val})} \end{cases}.$

Following this approximation methodology, it is only necessary to compute $\mathtt{F}_i[m, \ell, s]$ and $\mathtt{R}_{i,m}$ for $1 \leq m \leq M^\star$, thereby reducing the runtime of Algorithm 1 to $O(K^2NM^\star V)$ with minimal modification to the original algorithm. In the following, we derive the error bound of this approximation.

Theorem 11. *For any $i = 1, \ldots, N$, the approximated Shapley value $\widehat{\phi}_{z_i}^{(M^\star)}$ has the property of $\left|\widehat{\phi}_{z_i}^{(M^\star)}\right| \leq |\phi_{z_i}|$ and the approximation error is bounded by $\left|\widehat{\phi}_{z_i}^{(M^\star)} - \phi_i\right| \leq \varepsilon(M^\star)$ where*

$$\varepsilon(M^\star) := \sum_{m=M^\star+1}^{N} \left(\frac{1}{m-K} - \frac{1}{m}\right) + \frac{1}{N} \sum_{\ell=1}^{K-1} \frac{\binom{N}{\ell} - \binom{M^\star}{\ell}}{\binom{N-1}{\ell}} = O\left(\frac{K}{M^\star}\right)$$

Determining the Interval for Exact Shapley Value. Given the nice property that $|\widehat{\phi}_{z_i}^{(M^\star)}| \leq |\phi_{z_i}|$ and taking into account that $\widehat{\phi}_{z_i}^{(M^\star)}$ and ϕ_{z_i} invariably share the same sign, we can pinpoint a deterministic interval within which ϕ_{z_i} always resides based on the error bound in Theorem 11. Specifically, when $y_i = y^{(\mathrm{val})}$, we have $\phi_{z_i} \in \left[\widehat{\phi}_{z_i}^{(M^\star)}, \widehat{\phi}_{z_i}^{(M^\star)} + \varepsilon(M^\star)\right]$, and when $y_i \neq y^{(\mathrm{val})}$, we have

$$\phi_{z_i} \in \left[\widehat{\phi}_{z_i}^{(M^\star)} - \varepsilon(M^\star), \widehat{\phi}_{z_i}^{(M^\star)}\right] \tag{10}$$

The quality of the approximation of $\widehat{\phi}_{z_i}^{(M^\star)}$ is empirically studied in Section 5 and Appendix D.3.

Preservation of Shapley Axioms for approximated WKNN-Shapley. The Shapley value's axiomatic properties, particularly the Symmetry and Null Player axioms, are of paramount importance for upholding fairness when attributing value to individual players. These fundamental axioms have fostered widespread adoption of the Shapley value in various domains including data valuation and feature attribution. A credible approximation of the Shapley value, therefore, must preserve at least the Symmetry and Null Player axioms to ensure that the principal motivations for employing the Shapley value—fairness and equity—are not diminished. The prevalent Monte Carlo-based approximation techniques give randomized solutions and arguably muddy the clarity of Shapley value's axioms [8]. On the contrary, our deterministic approximation presented in Definition 10 preserves both pivotal axioms, as we show in the following theorem:

Theorem 12. *The approximated Shapley value $\{\widehat{\phi}_{z_i}^{(M^\star)}\}_{z_i \in D}$ satisfies the Symmetry and Null Player axiom.*

Moreover, while acknowledging that our approximation may not explicitly align with or is ill-defined in the context of the Efficiency and Linearity axioms, we note both of the two axioms have been questioned about their indispensability in the realm of data valuation [33, 13].

B Extension to multi-class classification setting

B.1 Naive Extension from Binary Classification Setting

We first discuss a simple, direct extension of our exact WKNN-Shapley algorithm from binary to multi-class classification setting. In Algorithm 1, the main idea is to maintain a record of $\mathtt{F}_i[m, \ell, s]$ for a singular scalar value s which represents the summation of "signed weights" $\widetilde{w}_j$. In order to extend this approach to the multi-class setting, it is natural to enhance this scalar representation to a "histogram" depiction, $\mathtt{F}_i[m, \ell, \mathbf{s}]$, where $\mathbf{s}$ is the vector sum of weights for each data point, and the weights are in the form of one-hot encoding. That is, in the multi-class setting, $\mathtt{F}_i$ is augmented to record the number of subsets such that the sum of weights of the data points in the one-hot encoding is equal to the histogram $\mathbf{s}$ (subject to the conditions analog to those in Definition 5). While this direct extension can compute the exact Data Shapley for the utility function in (4), it has a time complexity of $O(K^{1+C}N^2V^C)$ as we need to record $\mathtt{F}_i[m, \ell, \mathbf{s}]$ for all possible histograms $\mathbf{s} \in \mathtt{V}_{(K)}^C$. This is manageable for datasets with a modest size of class space. However, for datasets with a large class space, this complexity can render the runtime prohibitively large.

B.2 Utility Function that Enables More Efficient Computation of WKNN-Shapley

Due to the above-mentioned computational bottleneck, we introduce an alternative utility function for weighted KNN classifiers, which not only reflects the KNN classifiers' performance but also paves the way for a more efficient Data Shapley computation analogous to that of the binary setting.

Alternative Utility Function for Weighted Hard-Label KNN Classifiers. For a class $c \neq y^{(\text{val})}$, we denote

$$v^{(c)}(S; z^{(\text{val})}) := \mathbb{1}\left[\sum_{j=1}^{\min(K, |S^{(c)}|)} w_{\alpha_{x^{(\text{val})}}^{(S^{(c)}, j)}} \mathbb{1}[y_{\alpha_{x^{(\text{val})}}^{(S^{(c)}, j)}} = y^{(\text{val})}] \geq \sum_{j=1}^{\min(K, |S^{(c)}|)} w_{\alpha_{x^{(\text{val})}}^{(S^{(c)}, j)}} \mathbb{1}[y_{\alpha_{x^{(\text{val})}}^{(S^{(c)}, j)}} = c]\right] \tag{11}$$

where $S^{(c)} := \{(x, y) \in S : y \in \{y^{(\text{val})}, c\}\}$ is the subset of S whose labels are either $y^{(\text{val})}$ and c, and we propose an alternative utility function as follows:

$$\widetilde{v}(S; z^{(\text{val})}) := \frac{1}{C-1} \sum_{c \in [C] \setminus y^{(\text{val})}} v^{(c)}(S^{(c)}; z^{(\text{val})}) \tag{12}$$

Note that for binary classifiers, the new utility function $\widetilde{v}$ reduces to the original v. **Interpretation of the Alternative Utility Function:** The alternative utility function, $\widetilde{v}$, captures a fine-grained view of the classifier's performance. Instead of just deciding based on whether a prediction is correct as the original utility function in (4), it assesses the *rank* of the prediction confidence for the correct class, $y^{(\text{val})}$, among all potential class predictions in the weighted KNN classifier. Hence, $\widetilde{v}$ provides insight into not just the correctness, but also the relative confidence of a prediction with respect to other classes.

Data Shapley for $\widetilde{v}$. The linearity axiom of the Shapley value provides that

$$\phi_{z_i}(\widetilde{v}) = \frac{1}{C-1} \sum_{c \in [C] \setminus y^{(\text{val})}} \phi_{z_i}(v^{(c)})$$

Furthermore, observe that $v^{(c)}$ can be equivalently rewritten in a more compact way:

$$v^{(c)}(S) = \mathbb{1}\left[\sum_{j=1}^{\min(K, |S|)} \widetilde{w}_{\alpha_{x^{(\text{val})}}^{(S, j)}} \geq 0\right] \tag{13}$$

where $\widetilde{w}_i = \begin{cases} w_i & y_i = y^{(\text{val})} \\ -w_i & y_i = c \\ 0 & \text{otherwise} \end{cases}$. This formulation (13) mirrors the structure of (5), differing only in the weight definition $\widetilde{w}_i$. This similarity means that Algorithm 1 can be easily adapted to compute the

Shapley value $\phi_{z_i}(v^{(c)})$. Hence, we can first compute $\phi_{z_i}(v^{(c)})$ for each $c \in [C] \setminus y^{(\mathrm{val})}$ individually, and then aggregate these values. While this might imply an inevitable factor of C in the computational complexity, efficiency gains can be made. Specifically, every data point z_i with $y_i \notin \{y^{(\mathrm{val})}, c\}$ has a weight $w_i = 0$. Hence it is a null player that yields a Shapley value of $\phi_{z_i}(v^{(c)}) = 0$. Moreover, a simple result from the literature is that excluding null players does not affect the Shapley values of other players (see Theorem 5 in [27]). Hence, we can instead compute the Shapley value for a more simplified utility function that is the same as (13) but narrow to the subset $D_{y^{(\mathrm{val})},c} \subseteq D$ that comprises only data points labeled $y^{(\mathrm{val})}$ or c. As a result, the computational time to compute the Shapley value for $v^{(c)}$ reduces to $O(K|D_{y^{(\mathrm{val})},c}|^2 V)$. This provides a huge runtime saving when the dataset is balanced.

Theorem 13. *For a class-balanced training dataset D with C classes, the time complexity is $\{\phi_{z_i}(\widetilde{v})\}_{z_i \in D}$ is $O(\frac{K^2 N^2 V}{C})$.*

Remarkably, this methodology is even more efficient than its binary classification counterpart.

C Detailed Pseudo-code used in Implmentation

Algorithm 2 Weighted KNN-Shapley for binary classification (reader-friendly version)

Input:

- K – hyperparameter of weighted KNN algorithm.
- $z^{(\text{val})} = (x^{(\text{val})}, y^{(\text{val})})$ – the validation point.
- $D = \{z_i = (x_i, y_i)\}_{i=1}^N$ – sorted training set where $d(x_i, x^{(\text{val})}) \le d(x_j, x^{(\text{val})})$ for any $i \le j$.
- $M^\star$ – hyperparameter for SV approximation (Section A.2). $M^\star = N$ for exact SV calculation.

Compute the weight $w_i = \omega_{x^{(\text{val})}}(x_i)$ for $i \in \{1, \ldots, N\}$.
$\widetilde{w}_j = (2\mathbb{1}[y^{(\text{val})} = y_j] - 1)w_j$ for $i \in \{1, \ldots, N\}$.

for $i \in \{1, \ldots, N\}$ **do**

    `// Initialize` $\mathtt{F}_i$
    Initialize $\mathtt{F}_i[m, \ell, s] = 0$ for $m \in \{1, \ldots, M^\star\}, \ell \in \{1, \ldots, K-1\}, s \in \mathtt{V}_{(K)}$.
    for $m \in \{1, \ldots, M^\star\} \setminus \{i\}$ **do**
        $\mathtt{F}_i[m, 1, \widetilde{w}_m] = 1$

    `// Compute` $\mathtt{F}_i$ `(Runtime-optimized version)`
    for $\ell \in \{2, \ldots, K-1\}$ **do**
        $F_0[:] = \sum_{t=1}^{\ell-1} \mathtt{F}_i[t, \ell-1, :]$
        for $m \in \{\ell, \ldots, M^\star\} \setminus \{i\}$ **do**
            for $s \in \mathtt{V}_{(K)}$ **do**
                $\mathtt{F}_i[m, \ell, s] = F_0[s - w_m]$

    `// Compute` $\mathtt{R}_{i,m}$ `(Runtime-optimized version)`
    for $s \in \mathtt{V}_{(K)}$ **do**
        $R_0[s] = \sum_{t=1, t\neq i}^{\max(i+1, K+1)-1} \mathtt{F}_i[t, K-1, s]$.
    for $m \in \{\max(i+1, K+1), \ldots, M^\star\}$ **do**
        $\mathtt{R}_{i,m} = \begin{cases} \sum_{s \in [-\widetilde{w}_i, -\widetilde{w}_m)} R_0[s] & \text{for } y_i = y^{(\text{val})} \\ \sum_{s \in [-\widetilde{w}_m, -\widetilde{w}_i)} R_0[s] & \text{for } y_i \neq y^{(\text{val})} \end{cases}$
        $R_0 = R_0 + \mathtt{F}_i[m, K-1, :]$

    `// Compute` $\mathtt{G}_{i,\ell}$
    $\mathtt{G}_{i,0} = \mathbb{1}[w_i < 0]$.[a]
    for $\ell \in \{1, \ldots, K-1\}$ **do**
        $\mathtt{G}_{i,\ell} = \begin{cases} \sum_{m \in [M^\star] \setminus i} \sum_{s \in [-\widetilde{w}_i, 0)} \mathtt{F}_i[m, \ell, s] & \text{for } y_i = y^{(\text{val})} \\ \sum_{m \in [M^\star] \setminus i} \sum_{s \in [0, -\widetilde{w}_i)} \mathtt{F}_i[m, \ell, s] & \text{for } y_i \neq y^{(\text{val})} \end{cases}$

    `// Compute the Shapley value for` z_i
    $\phi_{z_i} = \mathtt{sign}(w_i) \left[\frac{1}{N} \sum_{\ell=0}^{K-1} \frac{\mathtt{G}_{i,\ell}}{\binom{N-1}{\ell}} + \sum_{m=\max(i+1, K+1)}^{M^\star} \frac{\mathtt{R}_{i,m}}{m\binom{m-1}{K}} \right]$.[b]

---

[a] Recall that we define $v(S) = \mathbb{1}\left[\sum_{j=1}^{\min(K, |S|)} \widetilde{w}_{(j)} \ge 0\right]$, hence $v(\{z_i\}) - v(\emptyset) \in \{-1, 0\}$ and is equal to -1 if and only if $w_i < 0$.

[b] $\mathtt{sign}(w) = \begin{cases} 1 & w > 0 \\ 0 & w = 0 \\ -1 & w < 0 \end{cases}$.

D Evaluation Settings & Additional Experiments

D.1 Experiment Settings

In Section 5 in the main text, the weights used in KNN are based on ℓ_2 distance between the training point and queried example, and then normalize all weights to $[0, 1]$. That is, the weight function

$$\omega_{x^{(\mathrm{val})}}(x_i) := \frac{\left\|x_N - x^{(\mathrm{val})}\right\| - \left\|x_i - x^{(\mathrm{val})}\right\|}{\left\|x_N - x^{(\mathrm{val})}\right\| - \left\|x_1 - x^{(\mathrm{val})}\right\|}$$

The weights are then discretized by rounding to the nearest values that can be represented with b bits. We set the number of bits $b = 3$ in all experiments unless explicitly specified.

D.2 Error From Discretization

We empirically study the difference between WKNN-Shapley computed based on the original continuous weights and the discretized weights. However, for continuous weights, it is computationally infeasible to compute the exact Data Shapley. Therefore, we instead look at the computed Shapley values' difference when using b bits and $b+1$ bits for $b = 1, 2, \ldots$. Figure 2 shows the results for ℓ_2 and ℓ_∞ error. We have two observations here: **(1)** The error converges quickly as b increases and is near zero after $b \geq 5$. **(2)** The larger the dataset size N is, the smaller the error is. This interesting phenomenon is because the errors are dominated by the differences in the Shapley value computed for influential data points. When the dataset size is small, there are more influential data points since the performance of models trained on different data subsets can be significantly different from each other. On the other hand, when the dataset size is larger, there will be fewer influential points since most of the data subsets have a high utility (see Figure 3 for the visualization of the comparison between the distribution of data value scores).

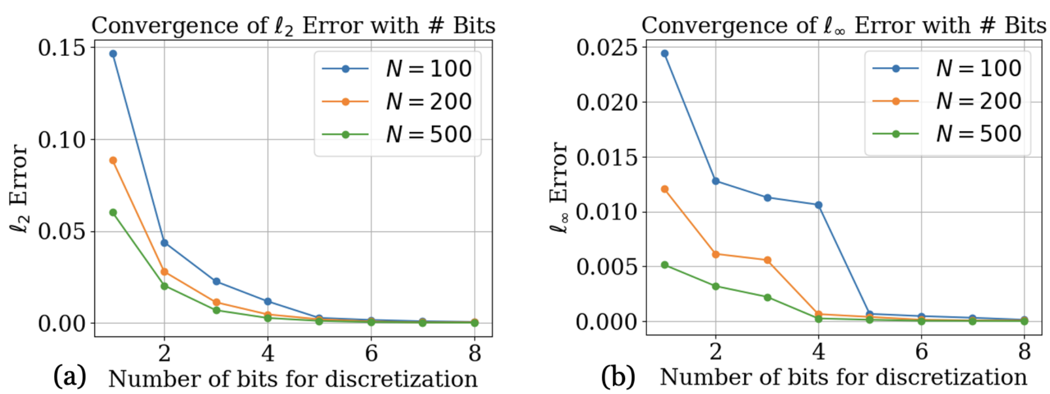

Figure 2: Convergence of the discretization error with the number of bits growth. The y-axis shows the ℓ_2 or ℓ_∞ norm of the difference between the Shapley values computed based on b bits and $b+1$ bits. The lower, the better. We use Fraud dataset from OpenML [3], and we use $K = 5$ here.

D.3 Error from Approximation

To visualize the quality of our approximation $\widehat{\phi}_{z_i}^{(M^\star)}$, Figure 4 provides a comparison between the exact Shapley value ϕ_{z_i}, and approximation $\widehat{\phi}_{z_i}^{(M^\star)}$, as well as the range introduced by Theorem 11. The figure shows that the true value always lies within the predicted range, which validates the correctness of our result. Moreover, we can see that even though the approximation $\widehat{\phi}_{z_i}^{(M^\star)}$ represents one end of the predicted range, the true value often comes with remarkable proximity to $\widehat{\phi}_{z_i}^{(M^\star)}$. It empirically reinforces our initial intuition: the building blocks for $\mathtt{G}_{i,\ell}$ (or $\mathtt{R}_{i,m}$), $\sum_{s\in[-\widetilde{w}_i,0)} \mathtt{F}_i[t,\ell,s]$ (or $\sum_{s\in[-\widetilde{w}_i,-\widetilde{w}_m)} \mathtt{F}_i[t,K-1,s]$), are often substantially more restrained in magnitude compared to their counterparts that encompass the entirety of $\mathtt{V}$.

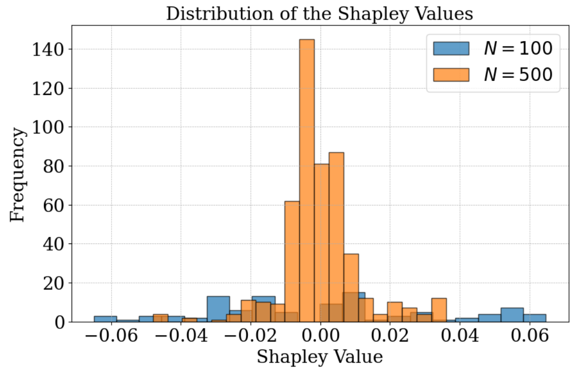

Figure 3: Distributions of WKNN-Shapley on different sizes of the subset of Fraud dataset from OpenML [3] (the number of bits for discretization $b = 5$ and $K = 5$).

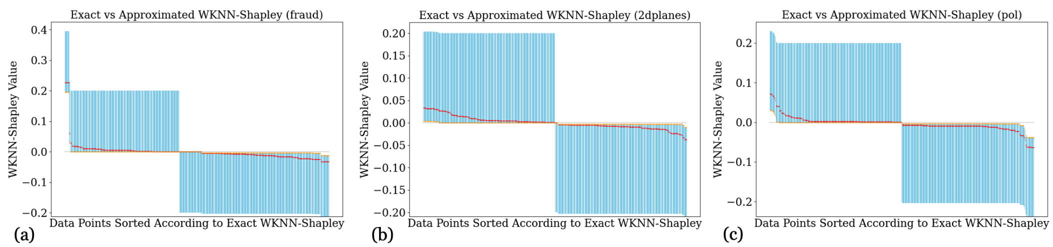

Figure 4: Visualization of the comparison between the exact and approximated WKNN-Shapley value on three OpenML datasets (Fraud, 2DPlanes, and Pol), as well as the interval devised by the approximation algorithm in (10). The red line corresponds to the exact WKNN-Shapley, and the orange line corresponds to the approximated WKNN-Shapley in (9), which is also . We adjust the value of $M^\star$ so that the error range $\varepsilon = 0.2$ for all three datasets.

E Missing Proofs

Theorem 14 (Restate of Theorem 2)**.** *For any data point $z_i \in D$ and any subset $S \subseteq D \setminus \{z_i\}$, the marginal contribution is*

$$
\begin{aligned}
&v(S \cup \{z_i\}) - v(S) \\
&= \begin{cases}
1 & z_i \in \texttt{NB}_{x^{(\mathrm{val})},K}(S \cup \{z_i\}), y_i = y^{(\mathrm{val})}, \sum_{z_j \in S} \widetilde{w}_j \in [-\widetilde{w}_i, 0) \text{ if } |S| \le K-1 \\
& \qquad \sum_{j=1}^{K-1} \widetilde{w}_{\alpha_{x^{(\mathrm{val})}}(S,j)} \in \left[-w_i, -\widetilde{w}_{\alpha_{x^{(\mathrm{val})}}(S,K)}\right) \text{ if } |S| \ge K \\
-1 & z_i \in \texttt{NB}_{x^{(\mathrm{val})},K}(S \cup \{z_i\}), y_i \neq y^{(\mathrm{val})}, \sum_{z_j \in S} \widetilde{w}_j \in [0, -\widetilde{w}_i) \text{ if } |S| \le K-1 \\
& \qquad \sum_{j=1}^{K-1} \widetilde{w}_{\alpha_{x^{(\mathrm{val})}}(S,j)} \in \left[-\widetilde{w}_{\alpha_{x^{(\mathrm{val})}}(S,K)}, \ -w_i\right) \text{ if } |S| \ge K \\
0 & \textit{Otherwise}
\end{cases}
\end{aligned}
$$

Proof. First of all, we observe that if $z_i \notin \texttt{NB}_{x^{(\mathrm{val})},K}(S \cup \{z_i\})$, i.e., if z_i is not within the K nearest neighbors of the queried example $x^{(\mathrm{val})}$ among the subset $S \cup \{z_i\}$, then the prediction of KNN classifier does not change, and hence we know that $v(S \cup \{z_i\}) = v(S)$.

If $z_i \in \texttt{NB}_{x^{(\mathrm{val})},K}(S \cup \{z_i\})$, we divide into two cases: ① If $|S| \le K-1$ we know that adding z_i will not exclude any other data point from the K nearest neighbors of $x^{(\mathrm{val})}$. Hence $v(S \cup \{z_i\}) - v(S) = 1$ if $y_i = y^{(\mathrm{val})}$ and $\sum_{z_j \in S} \widetilde{w}_j \in [-\widetilde{w}_i, 0)$, and $v(S \cup \{z_i\}) - v(S) = -1$ if $y_i \neq y^{(\mathrm{val})}$ and $\sum_{z_j \in S} \widetilde{w}_j \in [0, -\widetilde{w}_i)$. ② If $|S| \ge K$ we know that adding z_i will exclude the original Kth nearest neighbors of $x^{(\mathrm{val})}$ among dataset S. Hence, $v(S \cup \{z_i\}) - v(S) = 1$ if $y_i = y^{(\mathrm{val})}$ and

$\sum_{j=1}^{K-1} \widetilde{w}_{\alpha_{x^{(\text{val})}}(S,j)} \in \Big[-w_i, -\widetilde{w}_{\alpha_{x^{(\text{val})}}(S,K)}\Big)$, and $v(S \cup \{z_i\}) - v(S) = -1$ if $y_i \neq y^{(\text{val})}$ and $\sum_{j=1}^{K-1} \widetilde{w}_{\alpha_{x^{(\text{val})}}(S,j)} \in \Big[-\widetilde{w}_{\alpha_{x^{(\text{val})}}(S,K)}, \ -w_i\Big)$. □

Theorem 15 (Restate of Theorem 7)**.** *We have the following recursive relation of* $\texttt{F}_i[m, \ell, s]$.

1. ***Case of*** $\ell \leq K-1$***:***

$$\texttt{F}_i[m, \ell, s] = \sum_{t=1}^{m-1} \texttt{F}_i[t, \ell-1, s-w_m]$$

2. ***Case of*** $\ell \geq K$***:***

   (a) *When* $m < i$*:* $\texttt{F}_i[m, \ell, s] = 0$.

   (b) *When* $m > i$*:* $\texttt{F}_i[m, \ell, s] = \sum_{t=1, t\neq i}^{m-1} \texttt{F}_i[t, K-1, s]\binom{N-m}{\ell-K}$.

Proof. **Case of** $\ell \leq K-1$**:** This is because the inclusion of x_i in $\texttt{NB}_{x^{(\text{val})},K}(S \cup \{z_i\})$ is guaranteed for this range of ℓ. **Case of** $\ell \geq K$**:** Taking into account that x_m is the K-th nearest data point to $x^{(\text{val})}$ within S and that z_i invariably belongs to $\texttt{NB}_{x^{(\text{val})},K}(S \cup \{z_i\})$ because $i < m$. □

Theorem 16 (Restate of Theorem 8)**.** *For a weighted, hard-label KNN binary classifier using the utility function given by (5), the Shapley value can be expressed as:*

$$\phi_{z_i} = \frac{2\mathbb{1}[y_i = y^{(\text{val})}] - 1}{N} \left(\sum_{\ell=0}^{K-1} \binom{N-1}{\ell}^{-1} \texttt{G}_{i,\ell} + \sum_{m=\max(i+1,K+1)}^{N} \texttt{R}_{i,m} \binom{m-1}{K}^{-1} \frac{N}{m} \right)$$

where $\texttt{R}_{i,m} := \begin{cases} \sum_{t=1}^{m-1} \sum_{s \in [-\widetilde{w}_i, -\widetilde{w}_m)} \texttt{F}_i[t, K-1, s] & \textit{for } y_i = y^{(\text{val})} \\ \sum_{t=1}^{m-1} \sum_{s \in [-\widetilde{w}_m, -\widetilde{w}_i)} \texttt{F}_i[t, K-1, s] & \textit{for } y_i \neq y^{(\text{val})} \end{cases}$.

Proof. We state the proof for the case where $y_i = y^{(\text{val})}$, and the proof for the case where $y_i \neq y^{(\text{val})}$ is nearly identical. Recall that

$$\texttt{G}_{i,\ell} = \begin{cases} \sum_{m \in [N] \setminus i} \sum_{s \in [-\widetilde{w}_i, 0)} \texttt{F}_i\left[m, \ell, s\right] & \ell \leq K-1 \\ \sum_{m \in [N] \setminus i} \sum_{s \in [-\widetilde{w}_i, -\widetilde{w}_m)} \texttt{F}_i\left[m, \ell, s\right] & \ell \geq K \end{cases}$$

if $y_i = y^{(\text{val})}$.

When $\ell \geq K$, we have

$$\begin{aligned} \texttt{G}_{i,\ell} &= \sum_{m \in [N] \setminus i} \sum_{s \in [-\widetilde{w}_i, -\widetilde{w}_m)} \texttt{F}_i\left[m, \ell, s\right] \\ &= \sum_{m=\max(i+1,K+1)}^{N} \sum_{s \in [-\widetilde{w}_i, -\widetilde{w}_m)} \texttt{F}_i\left[m, \ell, s\right] \\ &= \sum_{m=\max(i+1,K+1)}^{N} \sum_{s \in [-\widetilde{w}_i, -\widetilde{w}_m)} \binom{N-m}{\ell-K} \sum_{t=1, t\neq i}^{m-1} \texttt{F}_i[t, K-1, s] \\ &= \sum_{m=\max(i+1,K+1)}^{N} \binom{N-m}{\ell-K} \sum_{s \in [-\widetilde{w}_i, -\widetilde{w}_m)} \sum_{t=1, t\neq i}^{m-1} \texttt{F}_i[t, K-1, s] \\ &= \sum_{m=\max(i+1,K+1)}^{N} \binom{N-m}{\ell-K} \texttt{R}_{i,m} \end{aligned}$$

where $\texttt{R}_{i,m} = \sum_{s \in [-\widetilde{w}_i, -\widetilde{w}_m)} \sum_{t=1, t\neq i}^{m-1} \texttt{F}_i[t, K-1, s]$.

$$
\begin{aligned}
\sum_{\ell=K}^{N-1} \frac{\mathtt{G}_{i,\ell}}{\binom{N-1}{\ell}} &= \sum_{\ell=K}^{N-1} \sum_{m=\max(i+1,K+1)}^{N} \frac{\binom{N-m}{\ell-K}\mathtt{R}_{i,m}}{\binom{N-1}{\ell}} \\
&= \sum_{m=\max(i+1,K+1)}^{N} \mathtt{R}_{i,m} \sum_{\ell=K}^{N-1} \frac{\binom{N-m}{\ell-K}}{\binom{N-1}{\ell}} \\
&= \sum_{m=\max(i+1,K+1)}^{N} \mathtt{R}_{i,m} \left(\sum_{\ell=K}^{N-1} \frac{\binom{m-1}{K}\binom{N-m}{\ell-K}}{\binom{N-1}{\ell}} \right) \binom{m-1}{K}^{-1} \\
&= \sum_{m=\max(i+1,K+1)}^{N} \mathtt{R}_{i,m} \left(\frac{N}{m} \right) \binom{m-1}{K}^{-1}
\end{aligned}
$$

□

Theorem 17 (Restate of Theorem 9)**.** *Algorithm 2 (in Appendix C) computes the exact Shapley value and achieves $O(K^2N^2V)$ time complexity.*

Proof. It is easy to see that the for-loop for computing $\mathtt{F}_i$ for $\ell \leq K$ requires a runtime of $O(KN|\mathtt{V}_{(K)}|)$. The for-loop for computing $\mathtt{R}_{i,m}$ requires a runtime of $O(N|\mathtt{V}_{(K)}|)$. The for-loop for computing $\mathtt{G}_{i,\ell}$ for $\ell \leq K$ requires a runtime of $O(KN|\mathtt{V}_{(K)}|)$. All of these subroutines are included in the outside for-loop for computing ϕ_{z_i} for all $z_i \in D$. Hence, the overall runtime is $O(KN^2|\mathtt{V}_{(K)}|) = O(K^2N^2V)$. □

Theorem 18 (Restate of Theorem 11)**.** *For any $i = 1, \ldots, N$, the approximated Shapley value $\widehat{\phi}_{z_i}^{(M^\star)}$ has the property of $\left|\widehat{\phi}_{z_i}^{(M^\star)}\right| \leq |\phi_{z_i}|$ and the approximation error is bounded by $\left|\widehat{\phi}_{z_i}^{(M^\star)} - \phi_i\right| \leq \varepsilon(M^\star)$ where*

$$
\varepsilon(M^\star) := \sum_{m=M^\star+1}^{N} \left(\frac{1}{m-K} - \frac{1}{m} \right) + \frac{1}{N} \sum_{\ell=1}^{K-1} \frac{\binom{N}{\ell} - \binom{M^\star}{\ell}}{\binom{N-1}{\ell}} = O\left(\frac{K}{M^\star} \right)
$$

Proof. In the exact algorithm 1, we have

$$
\phi_i = \underbrace{\frac{1}{N} \sum_{\ell=0}^{K-1} \frac{\mathtt{G}_{i,\ell}}{\binom{N-1}{\ell}}}_{(A)} + \underbrace{\sum_{m=\max(i+1,K+1)}^{N} \mathtt{R}_{i,m} \left(\frac{1}{m} \right) \binom{m-1}{K}^{-1}}_{(B)}
$$

First of all, note that

$$
\sum_{s \in \mathtt{V}} \mathtt{F}_i\left[m, \ell, s\right] = \binom{m-1-\mathbb{1}[i<m]}{\ell-1} \leq \binom{m-1}{\ell-1}
$$

for any $\ell \leq K$ since $\sum_{s\in\mathtt{V}} \mathtt{F}_i\left[m, \ell, s\right]$ is essentially the total number of subsets $S \subseteq D \setminus z_i$ of size ℓ where z_m is the farthest data point to the query example $x^{(\mathrm{val})}$.

Now, denote

$$
\widetilde{\mathtt{G}}_{i,\ell} := \sum_{m=1}^{M^\star} \sum_{s \in [-\widetilde{w}_i, 0)} \mathtt{F}_i\left[m, \ell, s\right]
$$

for $1 \leq \ell \leq K-1$. The gap between $\mathtt{G}_{i,\ell}$ and $\widetilde{\mathtt{G}}_{i,\ell}$ can be bounded as follows:

$$
\begin{aligned}
\left|\widetilde{\mathtt{G}}_{i,\ell} - \mathtt{G}_{i,\ell}\right| &= \sum_{m=M^\star+1}^{N} \sum_{s \in [-\widetilde{w}_i, 0)} \mathtt{F}_i\left[m, \ell, s\right] \\
&\leq \sum_{m=M^\star+1}^{N} \sum_{s \in \mathtt{V}} \mathtt{F}_i\left[m, \ell, s\right] \\
&\leq \sum_{m=M^\star+1}^{N} \binom{m-1}{\ell-1} \\
&= \sum_{m=\ell}^{N} \binom{m-1}{\ell-1} - \sum_{m=\ell}^{M^\star} \binom{m-1}{\ell-1} \\
&= \binom{N}{\ell} - \binom{M^\star}{\ell}
\end{aligned}
$$

Now we bound the error from taking the approximation $\widehat{\mathtt{R}}_{i,m} = 0$ for $m \geq M^\star + 1$. Since we have

$$
\begin{aligned}
\mathtt{R}_{i,m} &= \sum_{t=1}^{m-1} \sum_{s \in [-\widetilde{w}_i, -\widetilde{w}_m)} \mathtt{F}_i[t, K-1, s] \\
&\leq \sum_{t=1}^{m-1} \binom{t-1}{K-2} \\
&= \binom{m-1}{K-1}
\end{aligned}
$$

Hence

$$
\begin{aligned}
\sum_{m=\max(i+1, K+1, M^\star+1)}^{N} \mathtt{R}_{i,m} \left(\frac{1}{m}\right) \binom{m-1}{K}^{-1} &\leq \sum_{m=\max(i+1, K+1, M^\star+1)}^{N} \binom{m-1}{K-1} \left(\frac{1}{m}\right) \binom{m-1}{K}^{-1} \\
&\leq \sum_{m=M^\star+1}^{N} \binom{m-1}{K-1} \left(\frac{1}{m}\right) \binom{m-1}{K}^{-1} \\
&= \sum_{m=M^\star+1}^{N} \frac{K}{m(m-K)} \\
&= \sum_{m=M^\star+1}^{N} \left(\frac{1}{m-K} - \frac{1}{m}\right)
\end{aligned}
$$

Hence, for any data point z_i, we have

$$
\begin{aligned}
\left|\widehat{\phi}_{z_i}^{(M^\star)} - \phi_i\right| &= \frac{1}{N} \sum_{\ell=0}^{K-1} \frac{\left|\mathtt{G}_{i,\ell} - \widetilde{\mathtt{G}}_{i,\ell}\right|}{\binom{N-1}{\ell}} + \sum_{m=\max(i+1, K+1, M^\star+1)}^{N} \mathtt{R}_{i,m} \left(\frac{1}{m}\right) \binom{m-1}{K}^{-1} \\
&\leq \frac{1}{N} \sum_{\ell=1}^{K-1} \frac{\binom{N}{\ell} - \binom{M^\star}{\ell}}{\binom{N-1}{\ell}} + \sum_{m=M^\star+1}^{N} \left(\frac{1}{m-K} - \frac{1}{m}\right)
\end{aligned}
$$

□

The popularity of the Shapley value is attributable to the fact that it is the *unique* data value notion satisfying the following four axioms [22]:

- Null player: if $v\left(S \cup i\right) = v(S)$ for all $S \subseteq N \setminus i$, then $\phi\left(i; v\right) = 0$.

- Symmetry: if $v(S \cup i) = v(S \cup j)$ for all $S \subseteq N \setminus \{i, j\}$, then $\phi(i; v) = \phi(j; v)$.
- Linearity: For utility functions v_1, v_2 and any $\alpha_1, \alpha_2 \in \mathbb{R}$, $\phi\left(i; \alpha_1 v_1 + \alpha_2 v_2\right) = \alpha_1 \phi\left(i; v_1\right) + \alpha_2 \phi\left(i; v_2\right)$.
- Efficiency: for every v, $\sum_{i \in N} \phi(i; v) = v(N)$.

Among these axioms, linearity and efficiency are introduced for technical reasons and their necessity in machine learning has been questioned in the literature [33, 13]. On the other hand, Null player and Symmetry are generally interpreted as "fairness constraints", which are natural and important for data valuation. Here, we show that our approximation algorithm developed in Section A.2

Theorem 19 (Restate of Theorem 12)**.** *The approximated Shapley value* $\{\widehat{\phi}_{z_i}^{(M^\star)}\}_{z_i \in D}$ *satisfies the Symmetry and Null Player axiom.*

Proof. **Null Player.** If a data point z_i is a null player (i.e., $v(S \cup z_i) = v(S)$ for all $S \subseteq D \setminus \{z_i\}$), then it must have $\mathtt{R}_{i,m} = 0$ for all $0 \leq m \leq N$ and $\mathtt{G}_{i,\ell} = 0$ for all $0 \leq \ell \leq N-1$. Since $\widetilde{\mathtt{G}}_{i,\ell} \leq \mathtt{G}_{i,\ell}$, we know that $\widetilde{\mathtt{G}}_{i,\ell} = 0$ for all $0 \leq \ell \leq N-1$. Hence, we have $\widehat{\phi}_{z_i}^{(M^\star)} = 0$.

Symmetry. if two data points z_1, z_2 are symmetry (i.e., $v(S \cup z_1) = v(S \cup z_2)$ for all $S \subseteq D \setminus \{z_1, z_2\}$), then we must have $\widetilde{\mathtt{G}}_{1,\ell} = \widetilde{\mathtt{G}}_{2,\ell}$ and $\mathtt{R}_{1,m} = \mathtt{R}_{2,m}$. Therefore, we have $\widehat{\phi}_{z_1}^{(M^\star)} = \widehat{\phi}_{z_2}^{(M^\star)}$. □