Exact power indices for plurality-voting ensembles

Ilie Sarpe[*1†] **Theofanis Georgakopoulos**[*2] **Aristides Gionis**[1,3]
[1]*KTH Royal Institute of Technology*, Stockholm, Sweden
[2]*Ericsson BCSS*, Stockholm, Sweden
[3]*Digital Futures*, Stockholm, Sweden

Abstract

Plurality-voting ensembles aggregate the predictions of multiple base classifiers and output the most voted class, with ties typically broken at random. Despite the simple aggregation rule, attributing the impact of each base classifier to the ensemble's predictions is far from straightforward. That is, a single base classifier's impact depends on how it interacts with the votes of all other classifiers in the ensemble, for example, in resolving ties, and not only on its own predictions. We address the attribution problem using established tools from cooperative game theory, studying the Banzhaf and Shapley power indices for new games modeling plurality voting and random tie-breaking, for binary and multiclass classification settings. Computing power indices is, in general, intractable. Surprisingly, our methods, collectively named ENPOWER, enable the *exact* computation of Banzhaf and Shapley values in *linear time* for the binary classification setting and in *low-degree polynomial time* for the multiclass setting. The algorithms in ENPOWER rely on new combinatorial formulations of the Banzhaf and Shapley indices, yielding closed-form solutions for the binary case and ordinary generating function representations for the multiclass case. We validate empirically the methods in ENPOWER demonstrating their high applicability, providing attribution values not captured by existing methods. We apply the attribution of ENPOWER to: ensemble pruning and prompt-based ensembling of vision-language models.

1 Introduction

Ensemble methods, which combine the predictions of multiple *base classifiers* through a voting mechanism, are widely used in modern machine learning, as they achieve higher accuracy and robustness than individual models [37, 48]. Among the variety of aggregation rules [33], *plurality voting*—where the prediction corresponds to the most voted class, with ties broken uniformly at random—is a well-established scheme that applies directly to binary and multiclass classification and is used by many ensembles and popular software packages [24, 48].

Attribution is a central question for an ensemble, that is, quantifying how much each base classifier contributes to the ensemble's predictions. Simple heuristics, such as leave-one-out accuracy or per-model performance, ignore the interactions among base models: such heuristics measure the contribution of a base classifier to a single sub-ensemble, i.e., the full or empty one, respectively, among the $\Theta(2^n)$ sub-ensembles (where n corresponds to the number of base classifiers). For example, a single vote of a base classifier can modify the prediction of any sub-ensemble with a tie for the most voted class. Cooperative game theory addresses precisely this problem through *power indices*, in particular the *Banzhaf value* [4] and the *Shapley value* [38], uniquely characterized by natural axioms, faithfully capture the contribution of a player to the outcome of the game.

Computing power indices is intractable in general, involving a summation over an exponential number of coalitions, which we also refer to as sub-ensembles. Prior work has therefore resorted

[*]Equal contribution. [†]Contact author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

either to approximations [3, 36] or to restricted game classes, most prominently weighted majority games (WMGs) [42, 45]. However, these approaches do not capture plurality voting with random tie-breaking and cannot easily be applied to multiclass settings.

We first introduce a new game capturing plurality-voting ensembles with random tie-breaking. We then design ENPOWER, a collection of algorithms computing the *exact* Banzhaf and Shapley power indices of every base classifier in our game, for binary and multiclass classification. We consider base classifiers casting hard votes. Since our analysis considers only the votes of the base classifiers, our results are agnostic to the underlying models and apply to heterogeneous ensembles. The power indices we obtain can be used for many important applications such, ensemble pruning [19], data validation, and feature selection [16]. We first address the *binary* case, which differently from a WMG is not a simple game (i.e., a monotone game with values in $\{0, 1\}$), since random tie-breaking introduces a value of $\frac{1}{2}$ and the game is non-monotone due to the presence of incorrect classifiers. We obtain closed-forms for both power indices, computable in time $\mathcal{O}(|\mathcal{D}|n)$, where $|\mathcal{D}|$ is the number of test points. We then provide a new algorithmic contribution for the *multiclass* case with ℓ classes, where existing games do not capture faithfully ensemble aggregation. We show how to compute exactly the Banzhaf values in time $\mathcal{O}(|\mathcal{D}|n\ell^2)$ and the Shapley values in time $\mathcal{O}(|\mathcal{D}|n^3\ell)$, that is, polynomial in n and ℓ. We consider unweighted votes, which enable the computation of power indices in polynomial time; extending ENPOWER to weighted plurality voting is non-trivial.

Our technical results exploit *structural symmetries*: base classifiers with identical votes yield the same contributions to the ensemble. In the binary case, this yields closed-form solutions for both power indices. In the multiclass case, we encode coalition counts via *ordinary generating functions* [13] with low degree, computed efficiently with dynamic programming. Our techniques build on previous methods using ordinary generating functions to compute power indices in WMGs [1, 5]. In a WMG the degree of a generating function is bounded by the sum of the weights of the players, and the resulting algorithms are pseudo-polynomial, i.e., polynomial in n and in the sum of the weights. Our votes are unweighted; therefore, the degrees of our generating functions are polynomial in n and $\ell - 1$ and this is the source of our polynomial bounds.

In our experimental evaluation we use the Banzhaf and Shapley values computed with ENPOWER on two applications: *ensemble pruning* [18, 25], and *prompt-based ensembling of vision-language models* [35], on a zero-shot image classification task. Our empirical study shows that ENPOWER is highly efficient in practice computing *exact* power indices in a few seconds on most instances, and produces attribution values that are not captured by existing methods, directly characterizing plurality-voting ensembles with random tie-breaking.

2 Related work

Ensembles combine predictions of *base models* using a *voting mechanism* [37]. Prior work has considered learning algorithms to obtain homogeneous or heterogeneous base models, e.g., through boosting, bagging, or stacking [33], and different aggregation rules to combine predictions of base models. Widely-adopted rules include averaging, majority voting, and *plurality voting* [48], which is the focus of our work.

Attribution is a key challenge in explainable AI [30], and power indices have been used extensively to assign values to individual models or features [26, 40, 43]. To the best of our knowledge, our work is the first to consider the attribution problem in plurality-voting ensembles with random tie-breaking, for both binary and multiclass settings.

Earlier work has mostly focused on computing power indices for ensembles under *binary* classification, building on results from classical cooperative game theory [7, 10] and weighted majority games (WMGs) [12, 14, 15]. Under unitary weights, power indices for WMGs can be computed in polynomial time [42], while recent contributions use techniques similar to ours [17]. However, as we discuss in App. A, WMGs cannot be used effectively in our setting.

Close to our work, Rozemberczki and Sarkar [36] study a game for ensembles of soft binary classifiers aggregated through a *majority* rule. They design an approximation algorithm using multilinear extensions to estimate Shapley values. Ykhlef and Bouchaffra [46] use a classical result by Deng and Papadimitriou [10] to model ensemble diversity through Shapley values, capturing, however, only pairwise relationships among base models and not matching the ensemble's aggregation rule. Ykhlef and Bouchaffra [45] apply their diversity scoring to Banzhaf values computed over WMGs. In Sec. 5

we show that our exact methods are much faster than prior work [42, 46], and are comparable with the fastest approximate method by Rozemberczki and Sarkar [36]. Moreover, our algorithms yield new values not captured by existing methods. Our methods are specifically designed for multiclass settings, where prior work cannot be adapted; see App. A for details on additional related work.

3 Preliminaries

Let $\mathcal{C} = \{C_1, \ldots, C_n\}$ be an *ensemble* of n classifiers, where each classifier $C \in \mathcal{C}$ corresponds to a function $C : \mathcal{X} \mapsto \mathcal{Y}$ from a domain $\mathcal{X}$ to a target label set $\mathcal{Y} = \{y_1, \ldots, y_\ell\}$. We use $C(x) \in \mathcal{Y}$ to denote the class assigned by C to data point $x \in \mathcal{X}$. We do not assume that classifiers come from the same model family, neither we require access to training data.

Our goal is to characterize the contribution of each classifier $C \in \mathcal{C}$. Inspired by *attribution methods* in machine learning [16, 26, 41], we employ ideas from *cooperative game theory*.

Definition 3.1 ([7])**.** A *characteristic function game* (or simply *game*) G is given by a pair (N, v) where $N = [n]$ is a finite non-empty set of players, and $v : 2^N \to \mathbb{R}$ is a characteristic function mapping each *coalition* $S \subseteq N$ to a *value* $v(S)$.

Power indices are used to capture the contribution of players in games by considering changes in the value of coalitions that the players participate. Two of the most popular indices, widely adopted in the literature and considered in this paper, are the *Banzhaf* and *Shapley* values, defined below.

Definition 3.2. Given a game $G = (N, v)$ the *Banzhaf* value $\beta_i(G)$ (or simply β_i when the game G is clear) of player $i \in N$ and the *Shapley* value $\phi_i(G)$ (or simply ϕ_i) are defined respectively as,[1]

$$\beta_i \doteq \frac{1}{2^{n-1}} \sum_{i \in S \subseteq N} \left(v(S) - v(S - i)\right), \quad \phi_i \doteq \sum_{i \in S \subseteq N} \frac{(n - |S|)!(|S| - 1)!}{n!} \left(v(S) - v(S - i)\right). \quad (1)$$

The Shapley values are the *unique* values that satisfy the following desirable properties: *dummy player*, *symmetry*, *linearity*, and *efficiency* (see App. B.1). The Banzhaf values satisfy all the above properties *except* efficiency. While Shapley values are the natural choice when the values must sum to $v(N)$, the Banzhaf values are more robust to noise and easier to estimate, making them preferable in stochastic or large-scale regimes [43].

Next, we define the characteristic function v for our setting. Given a data point $x \in \mathcal{X}$ and a coalition $S \subseteq \mathcal{C}$, we let $\boldsymbol{p}(x, S) \in \mathbb{Z}^\ell$ be a *vote profile*, where the j-th component corresponds to the number of classifiers from S *voting* for y_j. We simplify $\boldsymbol{p}(x, S)$ to $\boldsymbol{p}(S)$ or $\boldsymbol{p}$ when x, S are clear from the context. We define the *majority* set of labels $\mathcal{M}(x, S) = \{y_j \in \mathcal{Y} : p_j \geq p_i, i \in [\ell]\}$, containing the labels receiving the most votes from the classifiers in the coalition $S \subseteq \mathcal{C}$.

We now introduce characteristic function games that model how plurality voting ensembles with random tie-breaking are used in practice for (1) *binary classification*; and (2) *multiclass classification*.

Binary classification. For a data point $z = (x, y) \in \mathcal{X} \times \mathcal{Y}$, with $\mathcal{Y} = \{0, 1\}$, and a coalition $S \subseteq \mathcal{C}, S \neq \emptyset$, we consider the *point-wise* characteristic function game $G_z = (\mathcal{C}, v_{\mathsf{B}})$ with

$$v_{\mathsf{B}}(z, S) \doteq \begin{cases} 1\,, & \text{if } \{y\} = \mathcal{M}(S)\,, \\ \frac{1}{2}\,, & \text{if } y \in \mathcal{M}(S) \text{ and } |\mathcal{M}(S)| = 2\,, \\ 0\,, & \text{otherwise}\,. \end{cases} \quad (2)$$

We let $v_{\mathsf{B}}(z, \emptyset) \doteq 0$ for technical purposes. In other words, for a data point $z = (x, y)$, the function v_{B} assigns a score to each coalition $S \subseteq \mathcal{C}$, which is: 1 if the majority of models in S correctly identify label y; 1/2 if there is a tie; and 0 if the majority of models in S make an incorrect prediction.

The game G_z is defined for a single data point z. For a test set $\mathcal{D}$, we define the game $G_{\mathcal{D}}$ by averaging the point-wise values $v_{\mathsf{B}}(z, S), z \in \mathcal{D}$, i.e., $v_{\mathsf{B}}(\mathcal{D}, S) = \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} v_{\mathsf{B}}(z, S)$. The Banzhaf and Shapley values satify linearity and an efficient computation of the values for $v_{\mathsf{B}}(z, S)$ implies efficient computation for the values for $v_{\mathsf{B}}(\mathcal{D}, S)$. Thus, we focus on computing the Banzhaf and Shapley values of G_z for a single point z. We drop the first argument of v_{B}, when clear from the context. It can be shown that v_{B} does not feature standard properties enabling efficient computation.

[1] For a set $S \subseteq N$ and a player $i \in N$, we define $S + i \doteq S \cup \{i\}$ and $S - i \doteq S \setminus \{i\}$.

Lemma 3.3. *The characteristic function $v_{\mathsf{B}}(z,\cdot) : 2^{\mathcal{C}} \to \{0, \frac{1}{2}, 1\}$ is neither supermodular, nor superadditive, nor monotone.*

All proofs can be found in App. C. Since $v_{\mathsf{B}}(z,\cdot)$ is not monotone and it can be $\frac{1}{2}$, our game is not a simple game (i.e., a monotone game with values in $\{0,1\}$), in particular not a WMG (see Def. B.6). Therefore, we cannot use existing standard algorithms to compute the power indices for our formulation. From the previous lemma, one may think that *exact* computation of power indices for our new game is intractable. Surprisingly, we can show that both the Banzhaf and Shapley values can be computed efficiently, see Sec. 4.1.

Theorem 3.4. *Let $\mathcal{C}$ be an ensemble of classifiers, $\mathcal{D}$ a test dataset, and $G_{\mathcal{D}} = (\mathcal{C}, v_{\mathsf{B}}(\mathcal{D},\cdot))$ the characteristic function game over $\mathcal{D}$. For all classifiers $C_i \in \mathcal{C}$ the Banzhaf values $\beta_i(G_{\mathcal{D}})$ and the Shapley values $\phi_i(G_{\mathcal{D}})$ can be computed in linear-time complexity $\mathcal{O}(|\mathcal{D}|n)$.*

The term $|\mathcal{D}|$ in the running time comes from averaging $\beta_i(G_z)$ and $\phi_i(G_z)$ over all $z \in \mathcal{D}$ to compute $\beta_i(G_{\mathcal{D}})$ and $\phi_i(G_{\mathcal{D}})$. The linear term $\mathcal{O}(n)$ is the time needed to compute $\beta_i(G_z)$ and $\phi_i(G_z)$.

Multiclass classification. We consider a set of labels $\mathcal{Y} = \{y_1, \ldots, y_\ell\}$ with $\ell > 2$. For a single data point $z = (x, y)$, we define the game $G_z = (\mathcal{C}, v_{\mathsf{MC}})$ where $v_{\mathsf{MC}}(z, \emptyset) \doteq 0$ and for $S \subseteq \mathcal{C}, S \neq \emptyset$ it is

$$v_{\mathsf{MC}}(z, S) \doteq \begin{cases} \frac{1}{|\mathcal{M}(S)|}\,, & \text{if } y \in \mathcal{M}(S)\,, \text{ and} \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{3}$$

We then define the characteristic function for dataset $\mathcal{D}$ by $v_{\mathsf{MC}}(\mathcal{D}, S) = \frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}} v_{\mathsf{MC}}(z, S)$. Clearly, the characteristic function v_{MC} generalizes v_{B} from Eq. (2). Finally, by linearity, the power indices of G_z can be expressed as the average over $\ell!$ binary games breaking ties according to a fixed total order over $\mathcal{Y}$; this average involves $\ell!$ games, none of which is a WMG (see App. B.2).

To state our result in the multiclass case we need one additional definition. First, without loss of generality (simply by permuting the class labels), we will assume that for any data point (x, y) it holds $y = y_1 \in \mathcal{Y}$. We let n_j be the number of classifiers predicting class $y_j, j \in [\ell]$ for data point (x, y). Finally, let $U = \{\boldsymbol{p}(x, \mathcal{C}) = (n_1, n_2, \ldots, n_\ell) : (x, y) \in \mathcal{D} \text{ and } n_2 \leq \cdots \leq n_\ell\}$ be the *set* of *unique* vote profiles, where n_1 is the number of correct classifiers. Clearly, $|U| \leq |\mathcal{D}|$; moreover, $|U| \leq n+1$ for binary classification, since a profile is determined by n_1. In practice, $|U|$ can be much smaller than $|\mathcal{D}|$ also in the multiclass case: when most classifiers are accurate, many points share their vote profile (see $|U|$ in Table 2). We now state the generalization of Theorem 3.4 in the multiclass case.

Theorem 3.5. *Let $\mathcal{C}$ be an ensemble of classifiers, $\mathcal{D}$ be a test dataset, and $G_{\mathcal{D}} = (\mathcal{C}, v_{\mathsf{MC}}(\mathcal{D},\cdot))$ be the characteristic function game over $\mathcal{D}$. For all classifiers $i \in [n]$ the Banzhaf values $\beta_i(G_{\mathcal{D}})$ can be computed exactly in time $\mathcal{O}(|U|n\ell^2 + |\mathcal{D}|n)$, and the Shapley values $\phi_i(G_{\mathcal{D}})$ can be computed exactly in time $\mathcal{O}(|U|n^3\ell + |\mathcal{D}|n)$.*

4 Methods

We present our polynomial-time algorithms for exact computation of Banzhaf and Shapley values. All missing proofs are reported in App. C.

4.1 Binary classification

Intuition. As explained in the previous section, we focus on computing the Banzhaf and Shapley values $\beta_i(G_z)$ and $\phi_i(G_z)$ for a single data point z, and then average over all data in $\mathcal{D}$ to compute $\beta_i(G_{\mathcal{D}})$ and $\phi_i(G_{\mathcal{D}})$. For each classifier $i \in [n]$ the power indices β_i and ϕ_i are given by a weighed sum of the marginal gain $v_{\mathsf{B}}(S) - v_{\mathsf{B}}(S - i)$, over all coalitions $S \subseteq [n]$ with $i \in S$. We observe that for a *correct* classifier, if $S - i \neq \emptyset$ then a non-zero marginal gain can only attain value $\frac{1}{2}$; and a symmetrical argument applies to an incorrect classifier. Furthermore, a marginal gain of $\frac{1}{2}$ is attained only in two cases, namely, (a) if the classifier i introduces a tie, in which case $v_{\mathsf{B}}(S - i) = 0$ and $v_{\mathsf{B}}(S) = \frac{1}{2}$; or ($b$) if i breaks a tie, in which case $v_{\mathsf{B}}(S - i) = \frac{1}{2}$ and $v_{\mathsf{B}}(S) = 1$. This key observation is formalized in Lemma C.1. Interestingly, given the special structure of the sets $S - i$, we can compute efficiently the number of such coalitions through combinatorial arguments. In addition, by combining with symmetry arguments we obtain our efficient algorithms from Theorem 3.4.

We now show the closed-form solutions of the power indices.

Theorem 4.1 (Banzhaf). *Let $\mathcal{C}$ be an ensemble, and $z = (x, y) \in \mathcal{D}$ be a test data point. Let $n_+ = |\{C \in \mathcal{C} : C(x) = y\}|$ be the number of classifiers in $\mathcal{C}$ that make a correct prediction for z, and $n_- = n - n_+$ the number of classifiers that make an incorrect prediction. For each $i \in [n]$ it is*

$$\beta_i(G_z) = \frac{1}{2^n}\left(\binom{n}{n_-} + 1\right) \text{ if } C_i(x) = y\,, \quad \beta_i(G_z) = -\frac{1}{2^n}\left(\binom{n}{n_+} - 1\right) \text{ if } C_i(x) \neq y\,. \tag{4}$$

Theorem 4.2 (Shapley). *Under the same notation as in Theorem 4.1, for each $i \in [n]$ it is*

$$\phi_i(G_z) = \frac{1}{n} + \frac{T_+}{2} \quad \text{if } C_i(x) = y\,, \quad \phi_i(G_z) = -\frac{T_-}{2} \quad \text{if } C_i(x) \neq y\,, \tag{5}$$

where, if we let $\sigma \in \{+, -\}$ and $\tau \in \{+, -\}$ with $\sigma \neq \tau$ then

$$T_\sigma = \frac{(n-2n_\sigma)!(2n_\sigma-1)!}{n!}\binom{n_\tau}{n_\sigma} + \sum_{h=1}^{\min\{n_\tau, n_\sigma - 1\}} \binom{n_\sigma - 1}{h-1}\binom{n_\tau}{h}\frac{(n-2h-1)!(2h-1)!}{n!}(2n_\sigma + n - 4h),$$

where $\binom{m}{k} = 0$, for any $k > m$ or $k < 0$.

The proofs of Theorems 4.1 and 4.2 are given in App. C.3. The resulting approach is shown in Algorithm 1. It receives as input the power index $\psi \in \{\phi, \beta\}$ to be computed, the trained ensemble $\mathcal{C}$, and the test dataset $\mathcal{D}$. The algorithm initializes $\psi_i = 0$, for all $i \in [n]$, and then iterates over all data points in $\mathcal{D}$. For each data point (x, y), it computes n_+ the number of classifiers that correctly predict the current label y. It uses n_+ to compute ψ_i via a closed-form solution through subroutine `Get_`ψ, the details of which are provided in Algorithm 2. The correctness of Algorithm 1 follows from Theorems 4.1 and 4.2 and the linearity of power indices.

To bound the time complexity we assume that evaluating the prediction $C(x)$ of each classifier $C \in \mathcal{C}$ can be done in $\mathcal{O}(1)$ time, and that each arithmetic operation takes $\mathcal{O}(1)$ time.[2] The correctness of Theorem 3.4 and stated complexity follows immediately as a corollary of Theorems 4.1 to 4.3.

Theorem 4.3. *Algorithm 1 computes the Banzhaf values $\psi_i = \beta_i(G_\mathcal{D})$ and the Shapley values $\psi_i = \phi_i(G_\mathcal{D})$ over the game $G_\mathcal{D}$ in time $\mathcal{O}(|\mathcal{D}|n)$. The memory required is $\mathcal{O}(n)$.*

4.2 Multiclass classification

Intuition. Let C_i be a correct classifier; an analogous argument will apply to an incorrect classifier. Consider all coalitions $S \subseteq [n]$ containing i where C_i yields non-zero marginal contribution $v_{\mathsf{MC}}(S) - v_{\mathsf{MC}}(S - i)$. These are the only coalitions that contribute to the power indices ϕ_i or β_i. We characterize the structure of these coalitions and the exact marginal gain introduced by C_i in Lemma C.2. Since the multiclass setting introduces a much more complex structure, we cannot count these coalitions directly. Instead, we leverage ordinary generating functions [13, 17] (see Def. B.10).

Fix an integer $h \geq 1$ and consider coalitions of *incorrect* classifiers where no incorrect label receives more than h votes. We design a polynomial $P_h(t)$ with variable t, where the k-th coefficient $[t^k]P_h(t)$ is the number of coalitions in $\mathcal{S}(k, h)$, i.e., coalitions where exactly k incorrect labels receive h votes, and all remaining labels receive fewer than h votes (Lemma 4.4). To form a coalition in $\mathcal{S}(k, h)$, we select, independently for each incorrect label y_j, $j \in [2, \ell]$, either: (a) fewer than h classifiers; or (b) exactly h of the n_j classifiers voting for y_j; and case (b) holds for exactly k labels. There are $A_j(h) = \sum_{q=0}^{h-1}\binom{n_j}{q}$ ways to select classifiers for case (a) and $B_j(h) = \binom{n_j}{h}$ ways to select classifiers for case (b). We multiply $B_j(h)$ by t, such that the exponent $k \geq 0$ of t^k is the number of incorrect labels for case (b). This yields $P_h(t) = \prod_{j=2}^{\ell}(A_j(h) + B_j(h)t)$ of degree at most $\ell - 1$, with $[t^k]P_h(t) = |\mathcal{S}(k, h)|$. We then show that the marginal gain of a correct classifier on a coalition formed by correct classifiers and a coalition in $\mathcal{S}(k, h)$ depends only on k and on whether there are $h - 1$ or h other correct classifiers (Lemma C.2). We use this result to obtain the Banzhaf value β_i for each correct classifier i by summing the marginal gains weighted by $[t^k]P_h(t)$ for $h \in [n_1]$ and $k \in [\ell - 1]$ (Theorem 4.6). Dividing $P_h(t)$ by the factor $(A_j(h) + B_j(h)t)$ of label y_j yields β_i for all classifiers voting for y_j, $j \in [2, \ell]$ (Corollary 4.5). Each $P_h(t)$ can be computed in $\mathcal{O}(\ell^2)$ time.

On the other hand, the Shapley formula weights each coalition S by a function of its size $|S|$, while each coefficient $[t^k]P_h(t)$ counts coalitions of different sizes together. To fix this issue we rely on a

[2] Our analysis can be trivially extended when evaluating a model on a test point has a different complexity.

bivariate polynomial $P_h(t,u)$ whose coefficient $[t^k u^m]P_h(t,u)$ is the number of coalitions in $\mathcal{S}(k,h)$ with exactly m classifiers (Lemma C.5). To do so, we track the size of the coalitions through a second variable u: (i) in case (a) above, selecting $q < h$ classifiers voting for y_j contributes u^q; while (ii) in case (b), selecting h classifiers contributes tu^h. The degrees of $P_h(t,u)$ in t and u are at most $\ell - 1$ and n, respectively. We compute $P_h(t,u)$ by dynamic programming, and obtain the Shapley values by combining the coefficients of these polynomials over $h \geq 1$ as in the Banzhaf case.

Finally, for the computation of both Banzhaf and Shapley values, we heavily exploit symmetries to avoid redundant computations, achieving a highly-optimized algorithm in practice. We proceed to state our technical results; the proofs are given in App. C.

4.2.1 Computation of Banzhaf values in the multiclass setting

We first introduce the univariate polynomial we use to compute efficiently the Banzhaf values.

Lemma 4.4. *Let $z = (x, y)$ be a test data point. For any $h \geq 1$, consider the generating function*

$$P_h(t) = \prod_{j=2}^{\ell} (A_j(h) + B_j(h)t)\,, \quad \textit{where} \quad A_j(h) = \sum_{q=0}^{h-1} \binom{n_j}{q} \quad \textit{and} \quad B_j(h) = \binom{n_j}{h}\,.$$

Fix $h \in [n]$ and $k \in [\ell-1]$. Let $\mathcal{S}(k,h)$ be the set of coalitions $S \subseteq \mathcal{C}'$ such that: (i) $\mathcal{C}' = \mathcal{C} \setminus \{C \in \mathcal{C} : C(x) = y\}$; (ii) $|\mathcal{M}(S)| = k$; (iii) $p_j(S) = h$ for each $j \in \mathcal{M}(S)$. Then $|\mathcal{S}(k,h)| = [t^k]P_h(t)$.

In other words, the coefficients of $P_h(t)$ correspond to the number of sets $|\mathcal{S}(k,h)|$ that are used to obtain a closed-form solution for $\beta_i(G_z)$, for all correct classifiers C_i. To obtain $\beta(G_z)$ for all C_i voting for an incorrect label $y_j \neq y_1$, we need to remove the contribution of classifiers voting for y_j from the sets in $\mathcal{S}(k,h)$ as specified by the next corollary.

Corollary 4.5. *Under the setting of Lemma 4.4, fix $j \in \{2, \ldots, \ell\}$, and let $\mathcal{S}_{-j}(k,h)$ be defined analogously to $\mathcal{S}(k,h)$ for coalitions $S \subseteq \mathcal{C}' \setminus \{C \in \mathcal{C} : C(x) = y_j\}$. That is, no coalition S from $\mathcal{S}_{-j}(k,h)$ contains a classifier voting for class y_j. Then it holds that*

$$|\mathcal{S}_{-j}(k,h)| = [t^k]\frac{P_h(t)}{A_j(h) + B_j(h)t}\,.$$

We show how to use the coefficients of $P_h(t)$ to obtain a closed-form solution for β_i's.

Theorem 4.6. *Let $z = (x, y_1) \in \mathcal{D}$ be a test data point. Let $f_1(k,h) = k\binom{n_1-1}{h} + \binom{n_1-1}{h-1}$ for any $k, h \geq 0$ and $f_j(k,h) = \binom{n_1}{h}\left[\binom{n_j-1}{h} + \frac{1}{k+1}\binom{n_j-1}{h-1}\right]$ for any $k, h \geq 0$ and $j \in \{2, \ldots, \ell\}$. For any correct classifier C_i with $C_i(x) = y_1$ and any incorrect classifier i' with $C_{i'}(x) = y_j$ it holds*

$$\beta_i = \frac{1}{2^{n-1}}\left(1 + \sum_{h=1}^{n_1}\sum_{k=1}^{\ell-1}\frac{f_1(k,h)}{k+1}[t^k]P_h(t)\right) \quad \textit{and} \quad \beta_{i'} = -\frac{1}{2^{n-1}}\sum_{h=1}^{n_j}\sum_{k=0}^{\ell-2}\frac{f_j(k,h)}{k+2}[t^k]\frac{P_h(t)}{A_j(h)+B_j(h)t}\,.$$

Theorem 4.6 has strong algorithmic implications: by computing the coefficients of polynomials $P_h(t)$ and $P_h(t)/(A_j + B_j(t))$, for $j \in \{2, \ldots, \ell\}$ and $h \geq 1$, we can efficiently compute $\beta_i(G_z)$ for all $i \in [n]$ and $z \in \mathcal{D}$. We detail the entire procedure in Algorithm 3, which, in addition to the closed-form solutions of Theorem 4.6, exploits *symmetries* to further reduce the time complexity.

Theorem 4.7. *Algorithm 3 computes the exact Banzhaf values $\beta_i(G_{\mathcal{D}})$ of the base classifiers of an ensemble $\mathcal{C}$ in the multiclass setting in time $\mathcal{O}(|U|n\ell^2 + |\mathcal{D}|n)$. The memory required is $\mathcal{O}(\ell(n + |U|))$.*

We can further reduce the complexity of computing $P_h(t)$ from $\mathcal{O}(\ell^2)$ to $\mathcal{O}(\ell \log \ell)$ by performing polynomial multiplication via the fast Fourier transform (FFT) algorithm, at the expense of higher constants and a more complex implementation. In practice, we find that our $\mathcal{O}(\ell^2)$ solution is highly efficient, as shown in our experimental evaluation (e.g., see Table 5).

4.2.2 Computation of Shapley values in the multiclass setting

As already mentioned, we cannot use $P_h(t)$ to obtain the values of $\phi_i(G_z)$, since the k-th coefficient $[t^k]P_h(t)$ accounts for sets of different cardinality. To fix this issue, we design a bivariate ordinary

generating function allowing us to keep track of the size of each set contributing to $\mathcal{S}(k,h)$ by partitioning it as $\mathcal{S}(k,h) = \cup_{m\geq 1}\mathcal{S}(k,h,m)$ and requiring each set of $\mathcal{S}(k,h,m)$ to have cardinality m. We introduce our main result below, while we defer the detailed technical results to App. C.5.

For $h \geq 1$ and $j \in [2,\ell]$, let $A_j(h;u) \doteq \sum_{q=0}^{h-1}\binom{n_j}{q}u^q$, $B_j(h;u) \doteq \binom{n_j}{h}u^h$, and $P_h(t,u) \doteq \prod_{j=2}^{\ell}(A_j(h;u) + B_j(h;u)t)$. The exponent of u will count the classifiers in a coalition, i.e., $[t^k u^m]P_h(t,u) = |\mathcal{S}(k,h,m)|$ (see Lemma C.5). When $u = 1$ we recover $A_j(h)$, $B_j(h)$ and $P_h(t)$ from Lemma 4.4. For ease of notation let $\eta(a) \doteq \frac{(n-a)!(a-1)!}{n!}$, for $1 \leq a \leq n$. Furthermore, let

$$f_1(k,h,m) \doteq \frac{1}{k+1}\left(k\binom{n_1-1}{h}\eta(m+h+1) + \binom{n_1-1}{h-1}\eta(m+h)\right) \text{ and}$$

$$f_j(k,h,m) \doteq \frac{1}{k+2}\binom{n_1}{h}\left[\binom{n_j-1}{h}\eta(m+2h+1) + \frac{1}{k+1}\binom{n_j-1}{h-1}\eta(m+2h)\right], \; j \in [2,\ell]\,.$$

By convention, a term with a zero binomial is 0, irrespective of the argument of $\eta(\cdot)$.

Theorem 4.8. *Let $z = (x,y) \in \mathcal{D}$ be a test data point. For any classifier C_i making a correct prediction, i.e., $C_i(x) = y_1$, it holds that its Shapley value over G_z is given by*

$$\phi_i(G_z) = \frac{1}{n} + \sum_{h=1}^{n_1}\sum_{k=1}^{\ell-1}\sum_{m=kh}^{n-n_1} f_1(k,h,m)[t^k u^m]P_h(t,u)\,.$$

Furthermore, for any classifier C_i predicting an incorrect class $C_i(x) = y_j$, $j \in \{2,\ldots,\ell\}$ it holds

$$-\phi_i(G_z) = \sum_{h=1}^{n_j}\sum_{k=0}^{\ell-2}\sum_{m=hk}^{n-(n_1+n_j)} f_j(k,h,m)[t^k u^m]\frac{P_h(t,u)}{A_j(h;u) + B_j(h;u)t}\,.$$

The above result implies that computing the coefficients of the polynomials $P_h(t,u)$, with degrees bounded by $\ell - 1$ in t and $n - n_1$ in u, enables a closed-form solution for ϕ_i. To compute the coefficients of $P_h(t,u)$ we develop a dynamic-programming algorithm presented in App. D.

Theorem 4.9. *Algorithm 4 computes the exact Shapley values $\phi_i(G_{\mathcal{D}})$ of the base classifiers of an ensemble $\mathcal{C}$ in the multiclass setting in time $\mathcal{O}(|U|n^3\ell + |\mathcal{D}|n)$. The memory required is $\mathcal{O}(\ell(n+|U|))$.*

5 Experiments

We consider the following research questions: **(Q1)** evaluate the efficiency and scalability of ENPOWER; **(Q2)** compare the rankings produced by ENPOWER and existing methods; **(Q3)** investigate the use of the rankings of ENPOWER for ensemble pruning; and **(Q4)** use ENPOWER for prompt ensembling of vision models on a zero-shot learning classification task. We defer to App. G.6 for synthetic experiments validating the tightness of our time complexity bounds.

Algorithms. Our algorithms for exact computation are denoted by ENPOWER-ϕ and ENPOWER-β.

Binary classification. We consider the following baselines, described in detail in App. F.1: `ISCG` [46]; `Tr-MLE` [34]; `Tr-PS` [27]; and `WMG` [42]. These baselines are associated with games defined in prior work. We also consider `MC-`ϕ and `MC-`β: Monte-Carlo approximations with sample size set to n^2.

Multiclass classification. We only show the scalability of ENPOWER (see Table 5), as all other methods, with the exception of `ISCG`, are not applicable to the multiclass setting, or have very high runtime. We show an application of the output of ENPOWER for the multiclass setting in Sec. 5.4. For the prompt-ensembling of Sec. 5.4 we also consider `Tr-MLE` and `Tr-PS`, which we apply through a "correct-versus-rest" reduction detailed in App. F.3.

Setup. Details on our experimental setup are in App. F.[3] on six popular datasets from the UCI repository (see App. F.1). For each dataset we obtain a 75/15/10 split of the data, where 75% is used for training, 15% for computing the power indices, and 10% as test data for the experiments on sub-ensemble selection, see also App. F. Baselines `Tr-MLE` and `Tr-PS` are from [36]. Unless otherwise stated we considered five random (seeded) splits of each dataset, processed as detailed

[3]The code to reproduce our results is available at `https://github.com/iliesarpe/enpower`.

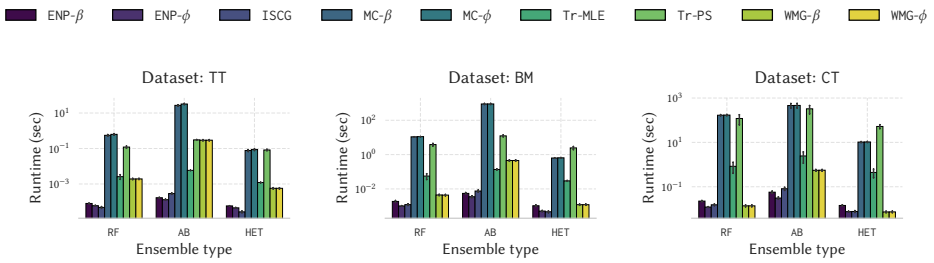


Figure 1: Runtime (mean and standard deviation) on our largest datasets, binary classification.

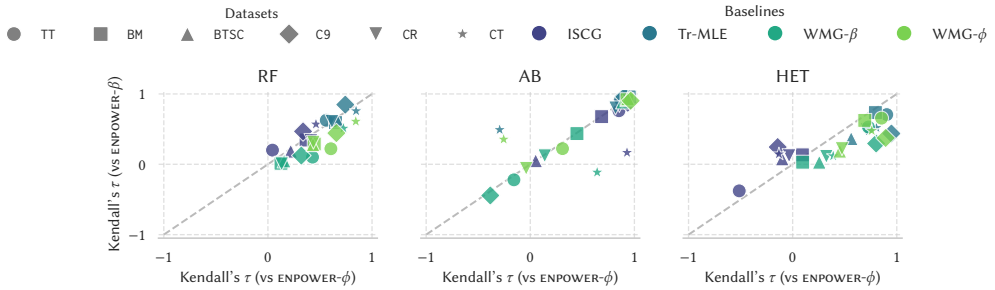


Figure 2: For each pair (dataset, baseline) encoded by a pair (shape, color), we display the Kendall's τ with respect to the exact rankings obtained by ENPOWER-ϕ and ENPOWER-β on the x-axis, and y-axis respectively, averaged over five splits. Each plot corresponds to a different ensemble.

above, and our metrics are averaged over such instances. We then execute each method once per split. Except for `MC` and `Tr-PS` all other methods are deterministic.

Ensembles. We consider three different ensembles (i) `RF`: a random forest with $n = 50$ trees; (ii) `AB`: an ensemble built with AdaBoost containing $n = 150$ models; and (iii) `HET`: and heterogeneous ensemble containing $n = 30$ models, 10 decision trees and 5 of each of the remaining types: SVMs, logistic regressors, gradient boosting, and multi layer perceptrons. See App. F for more details.

5.1 Efficiency and scalability

We start with answering **Q1**. The results are reported in Figure 1 and Figure 6. ENPOWER methods are among the fastest for binary classification. Even on the largest datasets, where the sampling methods `Tr-PS` and `MC` require up to hundreds and thousands of seconds, respectively, ENPOWER methods terminate in less than one second, as captured by Theorem 4.3. The exact method `WMG` is also very efficient, however it requires between one and three orders of magnitude more time than ENPOWER on `AB`, note that such configuration has the largest number of base classifiers ($n = 150$). In fact, the dynamic program of Uno [42] has a cost at least quadratic in n for each game of U, while Algorithm 1 evaluates a closed form. `ISCG` has a comparable runtime to ENPOWER, however it does not faithfully capture plurality voting ensembles (see Sec. 5.2 and App. F.3). Finally, `Tr-MLE` is more than one order of magnitude slower compared to ENPOWER on our largest datasets, while providing only approximate solutions. The runtimes of ENPOWER for the multiclass setting are reported in Table 5, where we observe a high runtime for computing Shapley values on our largest dataset (`CT`) on ensemble `AB` with $n = 150$ due to the $\mathcal{O}(n^3)$ term in the complexity. We would like to remind, however, that our method is the first specifically tailored to solve the multiclass case.

5.2 Comparison of rankings of base classifiers

Next, we consider **Q2**. We compare the rankings of the base classifiers produced by ENPOWER and the baselines using Kendall's τ. We recall, that rankings are used for different applications, including ensemble pruning, data validation, and interpretability. We report the results in Figure 2. We first observe that `ISCG` has low ranking correlation with ENPOWER on `RF` and `HET`, and a high

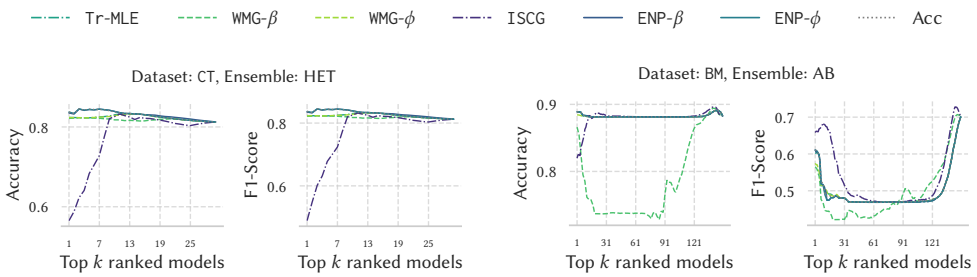


Figure 3: Generalization metrics for sub-ensemble selection, averaged over five random splits.

correlation on AB, except on BTSC. Similarly, except for few instances, the correlations of WMG-β and ENPOWER-β do not exceed 0.5. Interestingly, WMG-ϕ exhibits slightly higher correlations with ENPOWER-ϕ. Finally, there is quite high correlation (>0.5) among the rankings of ENPOWER-ϕ and Tr-MLE, despite the latter being approximate and on a different game, with the exception of CT on AB.

We note that the correlation values are particularly dependent on the baseline considered. Tr-MLE has $\tau > 0.5$ with ENPOWER-ϕ on most configurations except CT on AB. In contrast, baselines WMG and ISCG range from negative correlation to values above 0.9 for a fixed ensemble and different datasets. For example, the value of τ between WMG-β and ENPOWER-ϕ on AB is below -0.3 and above 0.8, on datasets C9 and BTSC, respectively. In summary, we observe that ENPOWER provides rankings that are not fully captured by previous methods, providing a unique characterization of plurality-voting ensembles with random tie-breaking.

5.3 Sub-ensemble selection

We now consider **Q3**. Given an ensemble $\mathcal{C}$ and a ranking of its base classifiers, we assess the generalization of the sub-ensemble obtained by retaining the top-k classifiers. We use rankings according to ENPOWER and the baselines. We additionally include a baseline ranking base classifiers by their individual accuracy on $\mathcal{D}_{\text{val}}$, denoted as ACC. For each sub-ensemble we evaluate its accuracy and (macro) F_1-score. We compute the rankings of all methods on $\mathcal{D}_{\text{val}}$, and both metrics on $\mathcal{D}_{\text{test}}$, of the same split. Classifiers with equal values are ordered by increasing index $C_i, i \in [n]$. We vary the parameter k in $[n]$. The predictions of a sub-ensemble are obtained by plurality voting with random tie-breaking, i.e., the voting rule of v_{B}. That is, for a test point where the two classes receive the same number of votes we draw the prediction uniformly at random. We compute both metrics on the resulting predictions, and report their mean over the five random splits.

We show some representative results in Figure 3 (see App. G.4 for an extended discussion). We observe that sub-ensembles produced with rankings by ENPOWER are often among those with the largest accuracy and F_1-score. For example, on CT with HET, ENPOWER matches or outperforms all baselines for all values of k; on BM with AB, ENPOWER matches ACC for all values of k, while the accuracy of WMG-β is much lower than that of all other methods for intermediate values of k. Interestingly, on the same configuration, ISCG attains a higher F_1-score for small values of k. To answer **Q3**, while our methods are not directly optimized for sub-ensemble selection, they can provide a valuable contribution in addition to existing methods, given their high practical applicability.

5.4 Large vision model prompt ensembling

Last, we consider **Q4**. Recall that prompt ensembling is an LLM-engineering technique used to improve performance on tasks such as zero-shot image classification [35]. Here, we select $n = 22$ general-purpose prompt templates, which act as our models; see Table 11. For each image in the input dataset we first compute its normalized embedding using CLIP, obtaining $\boldsymbol{x}_{\text{img}}$. Then we proceed as follows: (i) we instantiate each prompt template $i \in [n]$ with the ℓ possible classes, obtaining prompts $S_1^i, \ldots, S_\ell^i$; (ii) we use CLIP to compute the normalized embedding of each of the prompts obtaining vectors $\boldsymbol{v}_1^i, \ldots, \boldsymbol{v}_\ell^i$; (iii) we compute the dot products $\boldsymbol{x}_{\text{img}} \cdot \boldsymbol{v}_j^i$; (iv) we select the class

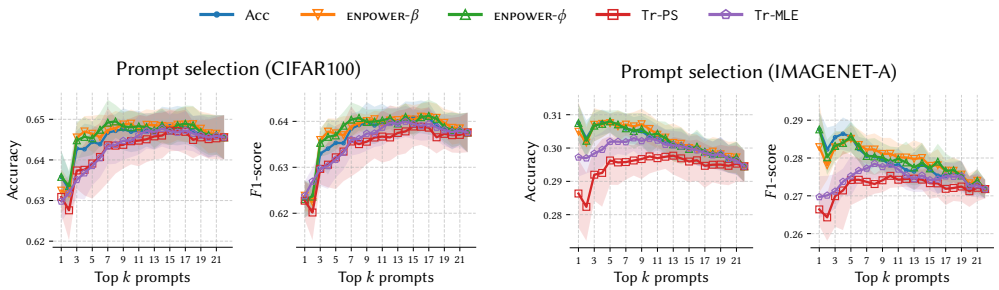


Figure 4: Accuracy and F_1-score test metrics for the prompt ensembling of CLIP on $\mathcal{D}_{\text{test}}$ (see Sec. 5.4), for varying k. We report mean and standard deviation (over multiple splits).

$j^*(i) = \arg\max_j \boldsymbol{x}_{\text{img}} \cdot \boldsymbol{v}_j^i$ as vote for model $i \in [n]$;[4] (v) the final predictions of the ensemble are evaluated according to plurality voting with random tie-breaking. We split each dataset in half obtaining $\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{test}}$ (over five random splits). We use $\mathcal{D}_{\text{val}}$ to compute the rankings of the models according to three different algorithms: ACC: ranking prompts by raw accuracy, a common practice for LLM engineering; ENPOWER-β and ENPOWER-ϕ: ranking using our definitions of β and ϕ for the multiclass setting. We also rank prompts with `Tr-MLE` and `Tr-PS` (see App. F.3). The resulting rankings are reported in Tables 9 to 11. Finally, we use $\mathcal{D}_{\text{test}}$ of each split to evaluate the accuracy and F_1-score on the sub-ensemble obtained by retaining only the top-k models, for $k \in \{1, \ldots, n\}$.

We report the result for the CIFAR and IMAGENET-A datasets in Figure 4, the missing dataset is in App. G.5, and the best and average metrics over k in Table 7. We observe that the sub-ensembles obtained through the rankings computed by ENPOWER attain a higher accuracy and F_1-score than the ones of `Tr-MLE` and `Tr-PS`, for the best k and on average over all $k \in [n]$, on all three datasets. Importantly, `Tr-MLE` and `Tr-PS` not accounting for how incorrect models for the $\ell - 1$ incorrect labels, differently from our game v_{MC} (see Apps. A and F.3). Furthermore the selection of ENPOWER matches or improves also over the baseline ACC. We also note that in general, the best performing ensembles ($k \in [1, 17]$, see Table 7) have smaller sizes than the full ensemble of $n = 22$ prompts. Highlighting that using a larger set of prompts to classify challenging images often yields less accurate predictions. Our results suggest that the Banzhaf and Shapley values computed by ENPOWER can be used to improve the predictive power of vision models under prompt ensembling. Another interesting application is enhancing vision model interpretabilty by comparing the rankings of the prompts across different methods, as shown in Tables 9 to 11, for which we defer to the discussion in App. G.5.

6 Conclusions

We introduce new characteristic function games modeling the predictions of models in plurality-voting ensembles with random tie-breaking, in binary and multiclass settings. We then design new *exact* polynomial-time algorithms, named collectively as ENPOWER, to compute the Banzhaf and Shapley values of the models in the ensemble. Our algorithms are based on non-trivial combinatorial identities, and the use of ordinary generating functions for the multiclass case. We show in practice that the indices obtained by ENPOWER: (i) are extremely efficient to compute and are not captured by previous methods; (ii) can be applied to ensemble pruning and LLM prompt ensembling with remarkable results. Interesting future directions include: (i) use our indices for data validation; (ii) study other popular power indices under our new game.

Acknowledgments and Disclosure of Funding

The authors declare no competing interests that could create a conflict of interest. This research was supported by 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. Computational resources were provided on the Berzelius system funded by the Knut and Alice Wallenberg foundation and operated by NAISS.

[4] Compared to the procedure of CLIP [35] each model selects a class, not a probability distribution.

References

[1] E. Algaba, J.M. Bilbao, J.R. Fernández García, and J.J. López. Computing power indices in weighted multiple majority games. *Mathematical Social Sciences*, 46(1):63–80, August 2003. ISSN 0165-4896. doi: 10.1016/s0165-4896(02)00086-0.

[2] R. Amer, F. Carreras, and A. Magaña. The Banzhaf–Coleman index for games with r alternatives. *Optimization*, 44(2):175–198, January 1998. ISSN 1029-4945. doi: 10.1080/02331939808844407.

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

[4] John F Banzhaf III. Weighted voting doesn't work: A mathematical analysis. *Rutgers Law Review*, 19(2):317–343, 1964.

[5] J. M. Bilbao, J. R. Fernández, A. Jiménez Losada, and J. J. López. Generating functions for computing power indices efficiently. *Top*, 8(2):191–213, December 2000. ISSN 1863-8279. doi: 10.1007/bf02628555.

[6] E. M. Bolger. A characterization of an extension of the banzhaf value to multicandidate voting games. *SIAM Journal on Discrete Mathematics*, 3(4):466–477, November 1990. ISSN 1095-7146. doi: 10.1137/0403040.

[7] Georgios Chalkiadakis, Edith Elkind, and Michael Wooldridge. *Computational Aspects of Cooperative Game Theory*. Springer International Publishing, 2012. ISBN 9783031015588. doi: 10.1007/978-3-031-01558-8.

[8] Vincent C. H. Chua, C. H. Ueng, and H. C. Huang. A method for evaluating the behavior of power indices in weighted plurality games. *Social Choice and Welfare*, 19(3):665–680, 2002. ISSN 1432-217X. doi: 10.1007/s003550100144.

[9] Shay Cohen, Gideon Dror, and Eytan Ruppin. Feature selection via coalitional game theory. *Neural Computation*, 19(7):1939–1961, 2007. ISSN 1530-888X. doi: 10.1162/neco.2007.19.7.1939.

[10] Xiaotie Deng and Christos H. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 19(2):257–266, 1994. ISSN 1526-5471.

[11] Thomas G. Dietterich. *Ensemble Methods in Machine Learning*, pages 1–15. Springer Berlin Heidelberg, 2000. ISBN 9783540450146. doi: 10.1007/3-540-45014-9_1.

[12] Alican Dogan and Derya Birant. A weighted majority voting ensemble approach for classification. In *4th International Conference on Computer Science and Engineering (UBMK)*, pages 1–6. IEEE, 2019. doi: 10.1109/ubmk.2019.8907028.

[13] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 4 edition, 2013. ISBN 9780521898065.

[14] Harris Georgiou, Michael Mavroforakis, and Sergios Theodoridis. *A Game-Theoretic Approach to Weighted Majority Voting for Combining SVM Classifiers*, pages 284–292. Springer Berlin Heidelberg, 2006.

[15] Harris V. Georgiou and Michael E. Mavroforakis. A game-theoretic framework for classifier ensembles using weighted majority voting with local accuracy estimates, 2013.

[16] 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.

[17] Naonori Kakimura and Yoshihiko Terai. Computing power indices in weighted majority games with formal power series. In *International Conference and Workshops on Algorithms and Computation*, pages 323–336, 2026.

[18] Zardad Khan, Naz Gul, Nosheen Faiz, Asma Gul, Werner Adler, and Berthold Lausen. Optimal trees selection for classification via out-of-bag assessment and sub-bagging. *IEEE Access*, 9: 28591–28607, 2021.

[19] Vrushali Y Kulkarni and Pradeep K Sinha. Pruning of random forest classifiers: A survey and future directions. In *2012 International Conference on Data Science & Engineering (ICDSE)*, pages 64–68, 2012.

[20] Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003. ISSN 1573-0565.

[21] Sascha Kurz, Alexander Mayer, and Stefan Napel. Weighted committee games. *European Journal of Operational Research*, 282(3):972–979, May 2020. ISSN 0377-2217. doi: 10.1016/j.ejor.2019.10.023.

[22] Sascha Kurz, Alexander Mayer, and Stefan Napel. Influence in weighted committees. *European Economic Review*, 132:103634, February 2021. ISSN 0014-2921. doi: 10.1016/j.euroecorev.2020.103634.

[23] Nan Li, Yang Yu, and Zhi-Hua Zhou. *Diversity Regularized Ensemble Pruning*, pages 330–345. Springer Berlin Heidelberg, 2012. ISBN 9783642334603.

[24] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3): 18–22, 2002. URL https://CRAN.R-project.org/doc/Rnews/.

[25] Zhenyu Lu, Xindong Wu, Xingquan Zhu, and Josh Bongard. Ensemble pruning via individual contribution ordering. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 871–880. ACM, 2010. doi: 10.1145/1835804.1835914.

[26] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[27] Sasan Maleki, Long Tran-Thanh, Greg Hines, Talal Rahwan, and Alex Rogers. Bounding the estimation error of sampling-based shapley value approximation, 2013.

[28] Tomomi Matsui and Yasuko Matsui. A survey of algorithms for calculating power indices of weighted majority games. *Journal of the Operations Research Society of Japan*, 43(1):71–86, 2000. ISSN 2188-8299. doi: 10.15807/jorsj.43.71.

[29] Alexander Mayer and Stefan Napel. Weighted scoring committees. *Games*, 12(4):94, December 2021. ISSN 2073-4336. doi: 10.3390/g12040094.

[30] Dang Minh, H. Xiang Wang, Y. Fen Li, and Tan N. Nguyen. Explainable artificial intelligence: a comprehensive review. *Artificial Intelligence Review*, 55(5):3503–3568, 2021. ISSN 1573-7462.

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

[32] Michael Mitzenmacher and Eli Upfal. *Probability and computing*. Cambridge University Press, 2017.

[33] Ammar Mohammed and Rania Kora. A comprehensive review on ensemble deep learning: Opportunities and challenges. *Journal of King Saud University - Computer and Information Sciences*, 35(2):757–774, 2023. ISSN 1319-1578.

[34] Guillermo Owen. Multilinear extensions of games. *Management Science*, 18(5-part-2):64–79, 1972. ISSN 1526-5501.

[35] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[36] Benedek Rozemberczki and Rik Sarkar. The Shapley value of classifiers in ensemble games. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, pages 1558–1567. ACM, October 2021. doi: 10.1145/3459637.3482302.

[37] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery*, 8(4), 2018. ISSN 1942–4795.

[38] Lloyd S. Shapley. A value for n-person games. *Princeton University Press*, 1953.

[39] Lloyd S Shapley. Cores of convex games. *International Journal of Game Theory*, 1(1):11–26, 1971.

[40] Erik Štrumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *The Journal of Machine Learning Research*, 11:1–18, 2010.

[41] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3):647–665, 2014.

[42] Takeaki Uno. *Efficient Computation of Power Indices for Weighted Majority Games*, pages 679–689. Springer Berlin Heidelberg, 2012. ISBN 9783642352614. doi: 10.1007/978-3-642-35261-4_70.

[43] 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.

[44] Danny Wood, Tingting Mu, Andrew M. Webb, Henry W. J. Reeve, Mikel Luján, and Gavin Brown. A unified theory of diversity in ensemble learning. *Journal of Machine Learning Research*, 24(1), 2023.

[45] Hadjer Ykhlef and Djamel Bouchaffra. An efficient ensemble pruning approach based on simple coalitional games. *Information Fusion*, 34:28–42, 2017. ISSN 1566-2535.

[46] Hadjer Ykhlef and Djamel Bouchaffra. Induced subgraph game for ensemble selection. *International Journal on Artificial Intelligence Tools*, 26(1), 2017. ISSN 1793-6349.

[47] Yi Zhang, Samuel Burer, W Nick Street, Kristin P Bennett, and Emilio Parrado-Hernández. Ensemble pruning via semi-definite programming. *Journal of machine learning research*, 7(7), 2006.

[48] Zhi-Hua Zhou. *Ensemble Methods, Foundations and algorithms*. Chapman and Hall/CRC machine learning and pattern recognition. Chapman and Hall CRC, second edition, 2025. ISBN 1040307663.

Limitations

We now list the major limitations and assumptions of our work.

- our methods are only applicable to ensembles constituting of base models that provide a hard classification;
- there are inputs for which our methods do not match baseline methods; we describe such behavior in Sec. 5 and discuss the related trade-offs. For example, on ensemble sub-selection, our methods may achieve high accuracy but low F_1-score, or vice-versa. To support our findings we provide additional results in App. G.
- we force each base classifier to provide a hard classification for our experiment regarding prompt ensembling in Sec. 5.4, to fit our framework. We remark that the original work of [35] used soft votes.

LLM usage

Large language models (LLMs) were used to assist portions of this research and manuscript preparation. In particular, LLMs were used to:

- assist with proofreading and improving presentation;
- help explore proof ideas and check edge cases;
- assist with implementation and debugging;
- support data analysis and visualization;
- assist with literature exploration.

All technical results, proofs, experiments, and conclusions were verified by the authors, who assume full responsibility for the content of the manuscript.

Appendix content

A Extended discussion on related work **16**

B Background **17**

B.1 Cooperative game theory 17

B.2 Connection with binary games 17

B.3 Combinatorics 18

C Proofs of main results **18**

C.1 Auxiliary technical results 18

C.2 Proofs of Sec. 3 20

C.3 Proofs of Sec. 4.1 20

C.4 Proofs of Sec. 4.2.1 23

C.5 Proofs of Sec. 4.2.2 26

D Pseudocode **30**

D.1 Binary classification 30

D.2 Multiclass classification 31

E Efficient and numerically stable implementation **33**

F Reproducibility **36**

F.1 Setup 36

F.2 Datasets 37

F.3 Baselines 38

G Additional experimental results **39**

G.1 Monte-Carlo approximation 39

G.2 Runtime of ENPOWER$_{\text{MC}}$ 40

G.3 Accuracy of the base classifiers 41

G.4 Ensemble sub-selection 41

G.5 CLIP prompt ensembling 41

G.6 Synthetic experiments 43

A Extended discussion on related work

The generalization of an ensemble is characterized by a *bias-variance-diversity* complexity trade-off that must be considered during learning [11, 20]. *Diversity*, which informally captures correlations among the models' predictions, requires a careful balancing as it directly impacts bias and variance [44]. Diversity motivates the need for ensemble *pruning* [23, 25], an active area of research [19, 47], which aims at selecting a sub-ensemble with better generalization performance, reducing significantly the size of the full ensemble and controlling the bias-variance-diversity trade-off. Our methods provide a valuable addition to existing pruning techniques (as we show in Sec. 5), especially given the efficiency of ENPOWER and the fact that we our methods are model-agnostic. We defer to dedicated textbooks and surveys for more details [37, 48].

Weighted majority games (WMGs) have been studied extensively in the literature, and under unitary weights, power indices for WMGs can be computed in polynomial time [17, 42]. However, as we discuss below, WMGs do not capture our setting. First, we recall that a WMG is a simple game, i.e., a monotone game in which coalitions are rewarded with values that are either 0 or 1, and that its quota, which is necessary for achieving a majority, is *fixed in advance* (see Def. B.6). Consider an ensemble $\mathcal{C}$ with unitary weights, as in our setting, and a quota $q \in [n]$. Given a model $i \in [n]$ and a coalition S such that $i \notin S$, the marginal contribution $v(S+i) - v(S)$ is nonzero if and only if $|S| = q-1$. That is, the Shapley and Banzhaf values of model $i \in [n]$ in a WMG depend only on the coalitions of size $q-1$. For example, under the majority quota $q = \lfloor n/2 \rfloor + 1$, no coalition with fewer than $\lfloor n/2 \rfloor$ classifiers contributes to the power indices of the models. In our experiments, we rank the models by their power indices and retain the top-k models, for every $k \in [n]$ (see Sec. 5). When $k < \lfloor n/2 \rfloor$, the power indices of a WMG ignore every coalition of size at most k. In contrast, under our game, the power indices can depend on coalitions of any size (see Lemma C.1).

Furthermore, most of the existing game formulations on which prior work is based, including the works by Rozemberczki and Sarkar [36], Uno [42] and Ykhlef and Bouchaffra [45] cannot be adapted to the multiclass setting without incurring high computational cost and loss in information. The reason is that typical approaches define $\mathcal{O}(\ell)$ binary games and aggregate the indices over all pairs retaining the correct label, or all possible pairs solving $\mathcal{O}(\ell^2)$ binary games. Despite being inefficient, this approach does not capture how the predictions for different classes interact with each other, as it is done by our definition of v_{MC}. Note that the work by Ykhlef and Bouchaffra [46] is the only method applicable for the multiclass setting, which however does not model accurately how the base models interact over the final ensembles' predictions. That is, the work of [46] assumes that classifiers correspond to nodes on a clique Q, and the characteristic function for a subset S of models is proportional to the weighted volume of the induced subgraph $Q[S]$, using the classical result from Deng and Papadimitriou [10] (see also App. F.3). This differs significantly from aggregation mechanisms used in ensemble models.

Finally, the rule underlying our characteristic function v_{MC}, i.e., plurality voting with tie-breaking, has been studied in social choice. Bolger [6] and Amer et al. [2] extend the Banzhaf index to games with r alternatives (here $r = \ell$). Their characteristic function is defined on the partitions of the players into r groups, one per alternative. The marginal contribution of a player p corresponds to the variation in the value of the characteristic function when the vote of p moves from one group to another. The resulting power index evaluates the marginal contributions over the partitions of $[n]$, of which there are up to ℓ^n, instead of over the 2^n coalitions considered in our work. Note that such indices range over all possible votes, thus in an unweighted setting all players are symmetric, and every player receives the same value, i.e., the indices cannot distinguish the classifiers. Kurz et al. [21], Mayer and Napel [29] characterize the weight vectors that yield the same outcome for every voting profile under plurality and other voting rules, and Kurz et al. [22] define new power indices for weighted committees. Chua et al. [8] compare power indices for weighted plurality, modeled as a simple game. We remark that none of the above works considers a notion of correctness for the votes of the players. In contrast, our power indices are computed for the votes of the classifiers, and vary across classifiers with different predictions even though the votes are unweighted, as shown both by our theoretical results (see Theorems 4.6 and 4.8) and in practice (see Sec. 5).

B Background

B.1 Cooperative game theory

Both the Banzhaf and Shapley values are grounded in well-established axiomatic foundations from cooperative game theory [7], which we recall below. The Shapley values is uniquely characterized by the four properties of dummy player, symmetry, linearity, and efficiency, whereas the Banzhaf value satisfies the properties of dummy player, symmetry and linearity.

Property B.1 (Dummy player)**.** Given a game $G = (N, v)$ a player $i \in N$ is said to be a *dummy player* if it holds that $v(S + i) = v(S)$ for all coalitions $S \subseteq [N]$ that do not contain i. The value $\psi_i(v)$ of a dummy player in a game G should equal to 0.

Property B.2 (Symmetry)**.** If i and j are two players in a game $G = (N, v)$ who are equivalent, in the sense that $v(S + i) = v(S + j)$ for any coalition $S \subseteq N$ that contains neither i nor j, then it should be $\psi_i(v) = \psi_j(v)$.

Property B.3 (Linearity)**.** If two coalition games $G_v = (N, v)$ and $G_w = (N, w)$ are combined, then their combined values should correspond to the values derived from v and the values derived from w, that is, it should be $\psi_i(v + w) = \psi_i(v) + \psi_i(w)$, for every $i \in N$ Also, for any real number α, it should be $\psi_i(\alpha\, v) = \alpha\, \psi_i(v)$, for every $i \in N$.

Property B.4 (Efficiency)**.** The sum of the values of all players in a game $G = (N, v)$ should be equal to the value of the grand coalition, i.e., $\sum_{i \in N} \psi_i(v) = v(N)$.

A *sum game* is the cooperative game whose characteristic function is the pointwise sum of those of a set of games defined over the same set of players.

Definition B.5 (Sum game)**.** Given an integer $k \geq 2$, and characteristic function games $G_1 = (N, v_1), \ldots, G_k = (N, v_k)$ the *sum* of $G_1, \ldots, G_k$ denoted with $G_+ = G_1 + \cdots + G_k = \sum_i G_i$ is a game $G_+ = (N, v_+)$ such that for any coalition $S \subseteq N$ it holds $v_+(S) = \sum_{i=1}^{k} v_i(S)$.

Weighted majority games (WMGs) are a classical model of voting systems in which each player carries a weight and a coalition wins precisely when its total weight reaches a fixed quota [7, 28].

Definition B.6 (Weighted majority game (WMG))**.** Let $N = \{1, \ldots, n\}$ be a finite set of players. A *weighted majority game* is a characteristic function game (N, v) for which given weights $w_1, \ldots, w_n \in \mathbb{R}_{\geq 0}$ and a quota $q > 0$ such that, for every coalition $S \subseteq N$,

$$v(S) = \begin{cases} 1 & \text{if } \sum_{i \in S} w_i \geq q, \\ 0 & \text{otherwise}\,. \end{cases}$$

B.2 Connection with binary games

The function v_{MC} can be equivalently expressed as the average of games with values in $\{0, 1\}$, where each game resolves ties according to a *fixed* total order over the labels. These games are not necessarily monotone, hence not necessarily simple (as we will show in Lemma B.7). Let Π be the set of the $\ell!$ total orders over $\mathcal{Y}$, with $\ell \geq 2$; for $\ell = 2$, v_{MC} coincides with v_{B}. Fix a data point $z = (x, y)$. For a fixed total ordering $\pi \in \Pi$, we define the game $G_\pi = (\mathcal{C}, v_\pi)$, where $v_\pi(\emptyset) \doteq 0$ and, for $S \neq \emptyset$, $v_\pi(S) \doteq 1$ if y precedes every other label of $\mathcal{M}(S)$ in π, and $v_\pi(S) \doteq 0$ otherwise.

Lemma B.7. *The following statements hold.*

(i) For every coalition $S \subseteq \mathcal{C}$, it holds $v_{\mathsf{MC}}(S) = \frac{1}{\ell!} \sum_{\pi \in \Pi} v_\pi(S)$.
(ii) For every $i \in [n]$ and $\psi \in \{\beta, \phi\}$, it holds $\psi_i(G_z) = \frac{1}{\ell!} \sum_{\pi \in \Pi} \psi_i(G_\pi)$.
(iii) There exist an ensemble $\mathcal{C}$ and a data point z such that, for every $\pi \in \Pi$, v_π is not monotone.

Proof. (i) For $S = \emptyset$ the statement follows trivially by definition. Fix $S \neq \emptyset$. If $y \notin \mathcal{M}(S)$, then $v_{\mathsf{MC}}(S) = 0$ and $v_\pi(S) = 0$ for every $\pi \in \Pi$. Let $y \in \mathcal{M}(S)$ and $k = |\mathcal{M}(S)|$. We count the orders $\pi \in \Pi$ with $v_\pi(S) = 1$, i.e., the orders in which y precedes the other $k - 1$ (incorrect) labels of $\mathcal{M}(S)$. Every order $\pi \in \Pi$ over the ℓ labels corresponds to exactly one choice of (a) the set of k positions of the labels of $\mathcal{M}(S)$, chosen in $\binom{\ell}{k}$ ways; (b) the order of the labels of $\mathcal{M}(S)$ in these positions; and (c) the order of the other $\ell - k$ labels in the remaining positions, chosen in $(\ell - k)!$

ways. It holds $v_\pi(S) = 1$ if and only if, in (b), y precedes the other $k-1$ labels of $\mathcal{M}(S)$ (i.e., there are $(k-1)!$ such orders of the tied labels yielding $v_\pi(S) = 1$). Therefore

$$\frac{1}{\ell!}\sum_{\pi\in\Pi} v_\pi(S) = \frac{1}{\ell!}\binom{\ell}{k}(k-1)!\,(\ell-k)! = \frac{1}{k} = v_{\mathsf{MC}}(S) \ .$$

(ii) From (i), it follows that G_z corresponds to the sum game $\sum_{\pi\in\Pi} G_\pi$ (Def. B.5) scaled by $\frac{1}{\ell!}$, and the claim follows from the linearity of β and ϕ (Property B.3).
(iii) Let $\mathcal{C} = \{C_1, C_2, C_3\}$, and let $C_1(x) = y$ and $C_2(x) = C_3(x) = y'$ for a label $y' \neq y$. Then $\mathcal{M}(\{C_1\}) = \{y\}$ and $\mathcal{M}(\mathcal{C}) = \{y'\}$ contain a single label, thus v_π does not depend on π on such coalitions. Hence, for every $\pi \in \Pi$, it holds $v_\pi(\{C_1\}) = 1 > 0 = v_\pi(\mathcal{C})$, while $\{C_1\} \subseteq \mathcal{C}$. That is, the set function v_π is not monotone, concluding the proof. □

Note that from the above it is immediate to observe that for any $\pi \in \Pi$ then $v_\pi(\cdot)$ is not a WMG since a WMG is monotone. Furthermore, Lemma B.7 does not yield an efficient algorithm for the computation of power indices. In fact, from (ii), we are required to obtain the power indices of $\ell!$ games, and none of such game corresponds to a WMG.

B.3 Combinatorics

We recall some standard combinatorial results used in our analysis.

Lemma B.8 (Pascal's rule)**.** *For any $n, k \in \mathbb{N}_{>0}$ it holds*

$$\binom{n-1}{k} + \binom{n-1}{k-1} = \binom{n}{k} ,$$

where we assume $\binom{n}{k} = 0$ whenever $n < k$.

Lemma B.9 (Vandermonde's identity)**.** *For any non-negative integers $a, b, c \in \mathbb{N}_{>0}$ it holds*

$$\sum_{j=0}^{c}\binom{a}{j}\binom{b}{c-j} = \binom{a+b}{c} .$$

Definition B.10 (Flajolet and Sedgewick [13], Chapter 1)**.** The *ordinary generating function* of a sequence $\{A_j\}$ is the power series

$$P(x) = \sum_{j=0}^{\infty} A_j x^j .$$

Variable x is denoted as the ordinary generating function *indeterminate*. We denote with $[x^\ell]P(x)$ the operation of extracting the coefficient of x^ℓ in $P(x)$.

C Proofs of main results

C.1 Auxiliary technical results

Lemma C.1. *Let (x, y) be a data point and consider the characteristic function v_{B}. Without loss of generality let $y = y_1 \in \mathcal{Y}$. For any classifier $C_i \in \mathcal{C}$, and any coalition $S \subseteq \mathcal{C}$ with $i \in S$ and $S - i \neq \emptyset$, the marginal gain $v_{\mathsf{B}}(S) - v_{\mathsf{B}}(S-i)$ takes values*

$$v_{\mathsf{B}}(S) - v_{\mathsf{B}}(S-i) = \begin{cases} \frac{1}{2}, & \text{if } C_i(x) = y, (p_1(S-i) = p_2(S-i)) \text{ or } (p_1(S-i) = p_2(S-i) - 1) , \\ -\frac{1}{2}, & \text{if } C_i(x) \neq y, (p_1(S-i) = p_2(S-i)) \text{ or } (p_1(S-i) - 1 = p_2(S-i)) , \\ 0, & \text{otherwise} . \end{cases}$$

If $S - i = \emptyset$ then $v_{\mathsf{B}}(S) - v_{\mathsf{B}}(S-i)$ is 1 if $C_i(x) = y$, and 0 if $C_i(x) \neq y$.

Proof of Lemma C.1. We fix a data point $(x, y = y_1)$ and a classifier $C_i \in \mathcal{C}$. To compute the marginal gain $v_{\mathsf{B}}(S) - v_{\mathsf{B}}(S-i)$, for any $S \subseteq \mathcal{C}$ with $i \in S$, first note that since $v_{\mathsf{B}}(\emptyset) = 0$, it follows that if $S - i = \emptyset$ then $v_{\mathsf{B}}(S) - v_{\mathsf{B}}(S-i)$ is 1 when $C_i(x) = y$, and it is 0 when $C_i(x) \neq y$. Hence we focus on the case $S - i \neq \emptyset$.

Correct classifier. $(C_i(x) = y)$ We distinguish the following cases, which follow from the definition of $v_{\mathsf{B}}(\cdot)$, see Eq. (2):

(*i*) $v_{\mathsf{B}}(S-i)=0$, hence $p_1(S-i)<p_2(S-i)$;
(*ii*) $v_{\mathsf{B}}(S-i)=\frac{1}{2}$, hence $p_1(S-i)=p_2(S-i)$; and
(*iii*) $v_{\mathsf{B}}(S-i)=1$, hence $p_1(S-i)>p_2(S-i)$.

Clearly for case (*iii*) it follows that $v_{\mathsf{B}}(S)=1$, and therefore $v_{\mathsf{B}}(S)-v_{\mathsf{B}}(S-i)=0$. For case (*i*) it holds that if $p_1(S-i)=p_2(S-i)-1$ then $v_{\mathsf{B}}(S)=\frac{1}{2}$, otherwise if $p_1(S-i)<p_2(S-i)-1$ then clearly $v_{\mathsf{B}}(S)=0$. It follows that for case (*i*) it is $v_{\mathsf{B}}(S)-v_{\mathsf{B}}(S-i)=\frac{1}{2}$ if and only if $p_1(S-i)=p_2(S-i)-1$. Finally, for case (*ii*) if $v_{\mathsf{B}}(S-i)=\frac{1}{2}$ it holds that $v_{\mathsf{B}}(S)=1$, proving the first part of the claim.

Incorrect classifier. ($C_i(x)\neq y$) The claim follows by a symmetrical argument noting that any classifier $i\in\mathcal{C}$ making an incorrect prediction can only decrease the value of $v_{\mathsf{B}}(S-i)$, for any $S-i\subseteq\mathcal{C}$. Specifically,

(*i*) if $v_{\mathsf{B}}(S-i)=1$ and $p_1(S-i)-1=p_2(S-i)$, then $v_{\mathsf{B}}(S)=\frac{1}{2}$, thus $v_{\mathsf{B}}(S)-v_{\mathsf{B}}(S-i)=-\frac{1}{2}$.
(*ii*) If $v_{\mathsf{B}}(S-i)=\frac{1}{2}$ then $p_1(S-i)=p_2(S-i)$. Hence, a classifier i making an incorrect prediction breaks the tie negatively, so $v_{\mathsf{B}}(S)=0$, yielding a marginal contribution of $-\frac{1}{2}$.
(*iii*) In all other cases, the characteristic function value does not change, yielding a marginal contribution of 0.

□

Lemma C.2. *Let (x,y) be a data point and consider the characteristic function v_{MC}. Without loss of generality let $y=y_1\in\mathcal{Y}$. For any classifier $C_i\in\mathcal{C}$ with $C_i(x)=y$, and any coalition $S\subseteq\mathcal{C}$ with $i\in S$ and $S-i\neq\emptyset$, the marginal gain $v_{\mathsf{MC}}(S)-v_{\mathsf{MC}}(S-i)$ takes values*

$$\begin{cases}\frac{1}{|\mathcal{M}(S-i)|+1}, & \text{if } |\mathcal{M}(S-i)|>0 \text{ and } p_1(S-i)=p_j(S-i)-1, \text{ for each } j\in\mathcal{M}(S-i)\,,\\ \frac{|\mathcal{M}(S-i)|-1}{|\mathcal{M}(S-i)|}, & \text{if } |\mathcal{M}(S-i)|>1 \text{ and } p_1(S-i)=p_j(S-i), \text{ for each } j\in\mathcal{M}(S-i)\,,\\ 0, & \text{otherwise}\,.\end{cases}$$

On the other hand, for any $C_i\in\mathcal{C}$ with $C_i(x)=y_j$, where $y_j\neq y_1$, and any coalition $S\subseteq\mathcal{C}$ with $i\in S$ and $S-i\neq\emptyset$, the marginal gain $v_{\mathsf{MC}}(S)-v_{\mathsf{MC}}(S-i)$ takes values

$$\begin{cases}-\frac{1}{|\mathcal{M}(S-i)|}, & \text{if } p_j(S-i)=p_{j'}(S-i), \text{ for each } j'\in\mathcal{M}(S-i) \text{ and } 1\in\mathcal{M}(S-i)\,,\\ -\frac{1}{|\mathcal{M}(S-i)|(|\mathcal{M}(S-i)|+1)}, & \text{if } p_j(S-i)=p_{j'}(S-i)-1, \text{ for each } j'\in\mathcal{M}(S-i) \text{ and } 1\in\mathcal{M}(S-i)\,,\\ 0, & \text{otherwise}\,.\end{cases}$$

Finally, if $S-i=\emptyset$, then $v_{\mathsf{MC}}(S)-v_{\mathsf{MC}}(S-i)$ is 1 if $C_i(x)=y$, and it is 0 if $C_i(x)\neq y$.

Proof of Lemma C.2. We fix a data point $(x,y=y_1)$ and a classifier $C_i\in\mathcal{C}$. We compute the marginal gain $v_{\mathsf{MC}}(S)-v_{\mathsf{MC}}(S-i)$ for any $S\subseteq\mathcal{C}$ with $i\in S$. First note that since $v_{\mathsf{MC}}(\emptyset)=0$, if $S-i=\emptyset$ then $v_{\mathsf{MC}}(S)-v_{\mathsf{MC}}(S-i)$ is 1 when $C_i(x)=y$, and it is 0 when $C_i(x)\neq y$. Hence we focus on the case $S-i\neq\emptyset$.

We distinguish the following cases, which follow from the definition of $v_{\mathsf{MC}}(\cdot)$, see Eq. (3):

(*i*) $v_{\mathsf{MC}}(S-i)=0$, hence $p_1(S-i)<p_j(S-i)$ for $j\in\mathcal{M}(S-i)$;
(*ii*) $v_{\mathsf{MC}}(S-i)=1/|\mathcal{M}(S-i)|$, hence $p_1(S-i)=p_j(S-i)$ for $j\in\mathcal{M}(S-i)$, $|\mathcal{M}(S-i)|\geq 1$;

Correct classifier. ($C_i(x)=y$) We first consider case (*i*) for which $1\notin\mathcal{M}(S-i)$. Note that $p_1(S)=p_1(S-i)+1$ and hence if $p_1(S-i)+1<p_j(S-i)=p_j(S)$ then $v_{\mathsf{MC}}(S)=0$ and therefore $v_{\mathsf{MC}}(S)-v_{\mathsf{MC}}(S-i)=0$. Thus, it holds $v_{\mathsf{MC}}(S)-v_{\mathsf{MC}}(S-i)\neq 0$ if and only if $p_1(S-i)+1=p_j(S-i)$ for each $j\in\mathcal{M}(S-i)$. Therefore $v_{\mathsf{MC}}(S)=1/(|\mathcal{M}(S-i)|+1)$ and in turn $v_{\mathsf{MC}}(S)-v_{\mathsf{MC}}(S-i)=1/(|\mathcal{M}(S-i)|+1)$.

For case (*ii*) note first that $p_j(S)=p_j(S-i)$ for each $j\geq 2$, since $C_i(x)=y$. Hence, for each $j\in\mathcal{M}(S-i), j\geq 2$, it holds $p_1(S)=p_1(S-i)+1=p_j(S-i)+1>p_j(S)$, and this also holds for each $j\notin\mathcal{M}(S-i), j\geq 2$, that is $p_j(S-i)<p_1(S-i)$. Therefore $\mathcal{M}(S)=\{1\}$ and $v_{\mathsf{MC}}(S)=1$. Note that it must be that $|\mathcal{M}(S-i)|>1$ otherwise if $|\mathcal{M}(S-i)|=1$ then $v_{\mathsf{MC}}(S)=v_{\mathsf{MC}}(S-i)$, yielding no marginal gain for the addition of i to $S-i$. Therefore, we have that $v_{\mathsf{MC}}(S)-v_{\mathsf{MC}}(S-i)=1-1/|\mathcal{M}(S-i)|$. This yields the claim for correct classifiers.

Incorrect classifier. ($C_i(x)=y_j, j\geq 2$) For case (*i*) we have that $p_1(S)=p_1(S-i)$, since $C_i(x)\neq y_1$, while $\max_{j'\in[\ell]}p_{j'}(S)\geq\max_{j'\in[\ell]}p_{j'}(S-i)>p_1(S-i)$, where the last inequality

is the defining property of case (i). Hence $1 \notin \mathcal{M}(S)$, and therefore $v_{\mathsf{MC}}(S) = 0 = v_{\mathsf{MC}}(S-i)$. That is, for case (i) it holds that $v_{\mathsf{MC}}(S) - v_{\mathsf{MC}}(S-i) = 0$.

For case (ii) we distinguish the following two sub-cases

$(ii.a)$ $p_j(S-i) = p_{j'}(S-i)$ for each $j' \in \mathcal{M}(S-i)$ and $1 \in \mathcal{M}(S-i)$;
$(ii.b)$ $p_j(S-i) = p_{j'}(S-i) - 1$ for each $j' \in \mathcal{M}(S-i)$ and $1 \in \mathcal{M}(S-i)$.

For $(ii.a)$ first note that $v_{\mathsf{MC}}(S-i) = 1/(|\mathcal{M}(S-i)|)$ by Eq. (3). Thus, $p_j(S) = p_j(S-i) + 1 > p_{j'}(S-i)$ for each $j' \in \mathcal{M}(S-i)$, implying $v_{\mathsf{MC}}(S) = 0$ and therefore $v_{\mathsf{MC}}(S) - v_{\mathsf{MC}}(S-i) = 0 - 1/(|\mathcal{M}(S-i)|) = -1/|\mathcal{M}(S-i)|$.

For $(ii.b)$ it holds $v_{\mathsf{MC}}(S-i) = 1/(|\mathcal{M}(S-i)|)$. Since $p_j(S-i) = p_{j'}(S-i) - 1$ we have that $p_j(S) = p_j(S-i) + 1 = p_{j'}(S-i)$ and therefore $j \notin \mathcal{M}(S-i)$ while $j \in \mathcal{M}(S)$. This implies that $v_{\mathsf{MC}}(S) = 1/(1 + |\mathcal{M}(S-i)|)$ and therefore $v_{\mathsf{MC}}(S) - v_{\mathsf{MC}}(S-i) = 1/(1+|\mathcal{M}(S-i)|) - 1/(|\mathcal{M}(S-i)|) = -1/[(|\mathcal{M}(S-i)|)(1+|\mathcal{M}(S-i)|)]$.

Finally, it is sufficient to observe that in all other cases where $1 \notin \mathcal{M}(S-i)$ the addition of i can never alter the value of $v_{\mathsf{MC}}(S-i)$. Similarly, if $1 \in \mathcal{M}(S-i)$ and $p_{j'}(S-i) - p_j(S-i) \geq 2$ for $j' \in \mathcal{M}(S-i)$, then $p_j(S) = p_j(S-i)+1 < p_{j'}(S-i)$, thus $\mathcal{M}(S) = \mathcal{M}(S-i)$ and the addition of i yields $v_{\mathsf{MC}}(S) = v_{\mathsf{MC}}(S-i)$. Since $p_{j'}(S-i) = \max_{h\in[\ell]} p_h(S-i)$ for $j' \in \mathcal{M}(S-i)$, our case split for $p_{j'}(S-i) - p_j(S-i)$ in $\{0, 1, \geq 2\}$ is exhaustive, which proves the claim for an incorrect classifier and concludes the proof. □

C.2 Proofs of Sec. 3

Lemma C.3 (Restatement of Lemma 3.3)**.** *The characteristic function $v_{\mathsf{B}} : 2^{\mathcal{C}} \to \{0, \frac{1}{2}, 1\}$ is*

(i) neither supermodular;
(ii) nor superadditive;
(iii) nor monotone.

Proof. We prove the lemma by providing counterexamples. Consider an ensemble $\mathcal{C} = \{C_1, \ldots, C_5\}$ with 5 classifiers. Fix a data-point $z = (x, 1)$ and let $C_1(x) = C_2(x) = C_5(x) = 1$ and $C_3(x) = C_4(x) = 0$. Fix the coalition $S = \{C_1, \ldots, C_4\}$. Clearly, $v_{\mathsf{B}}(S) = \frac{1}{2}$. Let $S_1 = \{C_1, C_2\}$ and $S_2 = \{C_3, C_4\}$, and note that $S_1 \cap S_2 = \emptyset$. Further, let $\{C_3\} = S_3 \subseteq S_4 = \{C_1, C_2, C_3\}$. We have:

(i) $1 - 1 = v_{\mathsf{B}}(S_4 \cup \{C_5\}) - v_{\mathsf{B}}(S_4) < v_{\mathsf{B}}(S_3 \cup \{C_5\}) - v_{\mathsf{B}}(S_3) = \frac{1}{2} - 0$;
(ii) $\frac{1}{2} = v_{\mathsf{B}}(S) = v_{\mathsf{B}}(S_1 \cup S_2) < v_{\mathsf{B}}(S_1) + v_{\mathsf{B}}(S_2) = 1 + 0$;
(iii) $\frac{1}{2} = v_{\mathsf{B}}(S) < v_{\mathsf{B}}(S_1) = 1$.

□

Corollary C.4. *The characteristic function game $G = (\mathcal{C}, v_{\mathsf{B}})$ is non-convex.*

Proof of Corollary C.4. It is a well-known result in cooperative game theory that a game $G = (N, v)$ is convex if and only if its characteristic function v is supermodular [39]. □

C.3 Proofs of Sec. 4.1

Proof of Theorem 4.1. Fix a test data point $z = (x, y) \in \mathcal{D}$ and, without loss of generality, let $y = y_1$. Expressing the summation in Def. 3.2 over the subsets of $\mathcal{C} - i$, the Banzhaf value of a classifier $i \in [n]$ corresponds to

$$\beta_i(G_z) = \frac{1}{2^{n-1}} \sum_{S \subseteq \mathcal{C}-i} (v_{\mathsf{B}}(S+i) - v_{\mathsf{B}}(S)) = \mathbb{E}_S \left[v_{\mathsf{B}}(S+i) - v_{\mathsf{B}}(S) \right] \ ,$$

where S is a uniformly random subset of $\mathcal{C} - i$. That is, each classifier $j \in \mathcal{C} - i$ is included in S independently with probability $1/2$. Let $n_+(S) \doteq |\{j \in S : C_j(x) = y\}|$ and $n_-(S) \doteq |\{j \in S : C_j(x) \neq y\}|$ be the number of correct and of incorrect classifiers in S. Recall n_+ and n_- corresponds to the number of correct and incorrect classifiers in $\mathcal{C}$. Furthermore, we have $p_1(S) = n_+(S)$ and $p_2(S) = n_-(S)$. Note that the random variables $n_+(S)$ and $n_-(S)$ are independent, since a model is either correct or not. Recall that we use the convention $\binom{m}{k} = 0$ for $k < 0$ or $k > m$.

Correct classifier. ($C_i(x) = y$) The set $\mathcal{C} - i$ contains $n_+ - 1$ correct and n_- incorrect classifiers, thus $n_+(S) \sim \mathrm{Bin}(n_+ - 1, 1/2)$ and $n_-(S) \sim \mathrm{Bin}(n_-, 1/2)$. By Lemma C.1, for $S \neq \emptyset$ the marginal gain of i is $\frac{1}{2}$ if $n_-(S) - n_+(S) \in \{0, 1\}$ and it is 0 otherwise, while for $S = \emptyset$ it is 1. Since, conditioning on $S = \emptyset$ it holds $n_-(S) - n_+(S) = 0$, we have that for every $S \subseteq \mathcal{C} - i$,

$$v_{\mathsf{B}}(S + i) - v_{\mathsf{B}}(S) = \frac{1}{2}\mathbb{1}[n_-(S) - n_+(S) \in \{0, 1\}] + \frac{1}{2}\mathbb{1}[S = \emptyset] \ .$$

Thus,

$$\beta_i(G_z) = \frac{1}{2}\Pr[n_-(S) = n_+(S)] + \frac{1}{2}\Pr[n_-(S) = n_+(S) + 1] + \frac{1}{2}\Pr[S = \emptyset] \ , \tag{6}$$

by taking the expectation over S. We now compute the three terms separately.

$$\begin{aligned}\Pr[n_-(S) = n_+(S)] &= \sum_{k\geq 0}\Pr[n_+(S) = k]\Pr[n_-(S) = k] = \sum_{k\geq 0}\binom{n_+ - 1}{k}\binom{n_-}{k}\left(\frac{1}{2}\right)^{n_+ - 1 + n_-} \\ &= \frac{1}{2^{n-1}}\sum_{k\geq 0}\binom{n_+ - 1}{k}\binom{n_-}{n_- - k} = \frac{1}{2^{n-1}}\binom{n-1}{n_-} \ .\end{aligned}$$

The first equality uses the independence of $n_+(S)$ and $n_-(S)$. The second one uses that both $n_+(S)$ and $n_-(S)$ are binomial random variables with parameter $1/2$. The third equality follows by $n_+ + n_- = n$ and the symmetry of the binomial coefficients, i.e., $\binom{a}{k} = \binom{a}{a-k}$. Finally, the last equality uses the Vandermonde's identity (Lemma B.9) with $a = n_+ - 1$, $b = n_-$ and $c = n_-$. The second term of Eq. (6) is obtained analogously,

$$\begin{aligned}\Pr[n_-(S) = n_+(S) + 1] &= \sum_{k\geq 0}\Pr[n_+(S) = k]\Pr[n_-(S) = k + 1] \\ &= \frac{1}{2^{n-1}}\sum_{k\geq 0}\binom{n_+ - 1}{k}\binom{n_-}{k+1} \\ &= \frac{1}{2^{n-1}}\sum_{k\geq 0}\binom{n_+ - 1}{k}\binom{n_-}{n_- - 1 - k} = \frac{1}{2^{n-1}}\binom{n-1}{n_- - 1} \ ,\end{aligned}$$

where the last equality follows by Vandermonde's identity with $a = n_+ - 1$, $b = n_-$ and $c = n_- - 1$. The third term of Eq. (6) corresponds to $\Pr[S = \emptyset] = 2^{-(n-1)}$, since S is a uniformly random subset of the $n - 1$ classifiers in $\mathcal{C} - i$. Combining all three terms above with Eq. (6) we have

$$\beta_i(G_z) = \frac{1}{2^n}\left(\binom{n-1}{n_-} + \binom{n-1}{n_- - 1}\right) + \frac{1}{2^n} = \frac{1}{2^n}\left(\binom{n}{n_-} + 1\right) \ ,$$

where we used Pascal's rule (Lemma B.8), which yields the first part of the claim.

Incorrect classifier. ($C_i(x) \neq y$) The set $\mathcal{C} - i$ contains n_+ correct and $n_- - 1$ incorrect classifiers, hence $n_+(S) \sim \mathrm{Bin}(n_+, 1/2)$ and $n_-(S) \sim \mathrm{Bin}(n_- - 1, 1/2)$. By Lemma C.1, for $S \neq \emptyset$ the marginal gain of i is $-\frac{1}{2}$ if $n_+(S) - n_-(S) \in \{0, 1\}$ and it is 0 otherwise, while for $S = \emptyset$ it is 0. Hence, for every $S \subseteq \mathcal{C} - i$,

$$v_{\mathsf{B}}(S + i) - v_{\mathsf{B}}(S) = -\frac{1}{2}\mathbb{1}[n_+(S) - n_-(S) \in \{0, 1\}] + \frac{1}{2}\mathbb{1}[S = \emptyset] \ ,$$

where the last term accounts for the correction of the marginal gain over $S = \emptyset$. Thus

$$\beta_i(G_z) = -\frac{1}{2}\Pr[n_+(S) = n_-(S)] - \frac{1}{2}\Pr[n_+(S) = n_-(S) + 1] + \frac{1}{2^n} \ ,$$

by taking the expectation over S. The two probabilities in the above sum follow from the computations of the previous case, by simply exchanging the roles of correct and of incorrect classifiers. That is, it is sufficient to replace $n_+ - 1$ with $n_- - 1$ and n_- with n_+. Therefore

$$\Pr[n_+(S) = n_-(S)] = \frac{1}{2^{n-1}}\binom{n-1}{n_+} \ , \qquad \Pr[n_+(S) = n_-(S) + 1] = \frac{1}{2^{n-1}}\binom{n-1}{n_+ - 1} \ .$$

Combining the above we have

$$\beta_i(G_z) = -\frac{1}{2^n}\binom{n}{n_+} + \frac{1}{2^n} = -\frac{1}{2^n}\left(\binom{n}{n_+} - 1\right) \ ,$$

by Pascal's rule (Lemma B.8), which yields the claim and concludes the proof. □

Proof of Theorem 4.2. Similar to the proof of Theorem 4.1 we prove the claim by distinguishing between correct and incorrect classifiers.

Correct classifier. ($C_i(x) = y$). Without loss of generality we fix $y = y_1$. Hence, $n_+ = p_1(\mathcal{C})$. From Def. 3.2 and Lemma C.1 it follows that the coalitions $i \in S, S \subseteq \mathcal{C}$ yielding a non-zero term for the difference $v_{\mathsf{B}}(S) - v_{\mathsf{B}}(S - i)$ in the computation of the Shapley value ϕ_i are such that, (1) $S - i = \emptyset$; (2) $p_1(S - i) = p_2(S - i)$; and (3) $p_1(S - i) = p_2(S - i) - 1$. Clearly in case (1) $v_{\mathsf{B}}(\{i\}) - v_{\mathsf{B}}(\emptyset) = 1$. Next, in cases (2) and (3) it holds that $v_{\mathsf{B}}(S) - v_{\mathsf{B}}(S - i) = \frac{1}{2}$, since $C_i(x) = y$. Recall that we let $n_+ = |\{C \in \mathcal{C} : C(x) = y\}|$, and $n_- = n - n_+$. We can therefore express the Shapley values of Eq. (1) as

$$\phi_i = \frac{1}{n} + \frac{1}{2}\left(T_1 + T_2\right),$$

where

$$T_1 = \sum_{h=1}^{\min\{n_+ - 1, n_-\}} \frac{(n - (2h+1))!(2h)!}{n!} \binom{n_+ - 1}{h} \binom{n_-}{h}$$

corresponds to the contribution for the Shapley value ϕ_i provided by coalitions of the type $S - i$ for which it holds $p_1(S - i) = p_2(S - i)$. Furthermore,

$$T_2 = \sum_{h=1}^{\min\{n_+, n_-\}} \frac{(n - 2h)!(2h - 1)!}{n!} \binom{n_+ - 1}{h - 1} \binom{n_-}{h}$$

corresponds to the contribution for the Shapley value ϕ_i provided by coalitions $S - i$ such that $p_1(S - i) = p_2(S - i) - 1$. Finally, the term $1/n$ accounts for $S - i = \emptyset$. Hence,

$$\begin{aligned} T_1 + T_2 = {} & \frac{(n - 2n_+)!(2n_+ - 1)!}{n!} \binom{n_-}{n_+} \\ & + \sum_{h=1}^{\min\{n_+ - 1, n_-\}} \binom{n_-}{h} \frac{(n - 2h - 1)!(2h - 1)!}{n!} \left[(n - 2h) \binom{n_+ - 1}{h - 1} + 2h \binom{n_+ - 1}{h} \right] \end{aligned}$$

By simple manipulations we obtain the first part of the claim

$$\begin{aligned} (n - 2h) \binom{n_+ - 1}{h - 1} + 2h \binom{n_+ - 1}{h} &\overset{(a)}{=} (n - 2h) \binom{n_+ - 1}{h - 1} + 2(n_+ - h) \binom{n_+ - 1}{h - 1} \\ &= \binom{n_+ - 1}{h - 1} (n + 2n_+ - 4h), \end{aligned}$$

where in (a) we used the known identity $k\binom{n}{k} = n\binom{n}{k-1}$.

Incorrect classifier. ($C_i(x) \neq y$). Following the analysis of the case above, by noticing the analogy in the argument for incorrect classifiers, with respect to correct classifiers, we can express

$$\begin{aligned} \phi_i = -\frac{1}{2} \Bigg(& \sum_{h=1}^{\min\{n_- - 1, n_+\}} \frac{(n - (2h+1))!(2h)!}{n!} \binom{n_- - 1}{h} \binom{n_+}{h} \\ & + \sum_{h=1}^{\min\{n_-, n_+\}} \frac{(n - 2h)!(2h - 1)!}{n!} \binom{n_- - 1}{h - 1} \binom{n_+}{h} \Bigg). \end{aligned}$$

Following the same manipulations for the case of correct classifiers we obtain the claim. □

Proof of Theorem 4.3. First we note that by the *symmetry* property, it is a trivial fact that $\phi_i = \phi_{i'}$ and $\beta_i = \beta_{i'}$ for any two classifiers $i \neq i'$ and $i, i' \in [n]$ such that $C_i = C_{i'}$. Furthermore, recall that symmetry holds for both ϕ and β. This property allows us to compute the closed-form solution of ϕ_i and β_i only *once* for each game G_z, and assign such values to all classifier according to their predictions, as detailed below.

Banzhaf power index. The values of $\beta_i, i \in [n]$ for all classifiers can be computed by evaluating the closed-form solution of Theorem 4.1, in $\mathcal{O}(1)$ time. Furthermore, note that the value of $\beta_i(G_z), i \in$

$[n]$ will be equal for all correct classifiers, and incorrect classifiers respectively over a fixed $(x, y) \in \mathcal{D}$ by symmetry. Hence, we can compute the two terms of Theorem 4.1, a single time for each test data point. Therefore, the complexity of obtaining $\beta_i(G_{(x,y)})$ is bounded by $\sum_{i=1}^{n} \mathcal{O}(1) = \mathcal{O}(n)$. To obtain $\beta_i(G_{\mathcal{D}})$ we use the linearity property of the Banzhaf power index, performing $\mathcal{O}(|\mathcal{D}|)$ computations, one on each test data point in $\mathcal{D}$. Yielding as final time complexity

$$\sum_{(x,y)\in\mathcal{D}} \sum_{i=1}^{n} \mathcal{O}(1) = \mathcal{O}(|\mathcal{D}|n)\,,$$

which yields the first part of the claim.

Shapley power index. By Theorem 4.2 the values of $\phi_i, i \in [n]$ for all classifiers can be computed through the evaluation of the summation from Eq. (5). The summation can be evaluated once per each game G_z efficiently in linear time $\mathcal{O}(n)$ (see App. E). Similar to the Banzhaf case above, using the *symmetry* of Shapley values for a game G_z then all correct classifiers will have the same Shapley value. Similarly, all incorrect classifiers will also attain the same Shapley value, hence we only need to compute the summation in Eq. (5) at one time. That is we have to obtain ϕ_i for a classifier $i \in [n]$ such that $C_i(x) = y$. Then we know, thanks to the *efficiency* property, that

$$v_{\mathsf{B}}([n]) = \sum_{i\in[n]} \phi_i = \sum_{i:C_i(x)=y} \phi_i + \sum_{i:C_i(x)\neq y} \phi_i\,.$$

It follows that the value ϕ_i, for any i such that $C_i(x) \neq y$, can be computed by $(v_{\mathsf{B}}([n]) - \sum_{i:C_i(x)=y} \phi_i)/n_-$, where $n_- = |\{i : C_i(x) \neq y\}|$ provided that $n_- > 0$. Therefore, the final complexity of the algorithm is bounded by

$$\sum_{(x,y)\in\mathcal{D}} \left(\mathcal{O}(n) + \sum_{i=1}^{n} \mathcal{O}(1)\right) = \mathcal{O}(|\mathcal{D}|n)\,.$$

Concerning the memory complexity, note that by symmetry, we have to compute the values of β_i (or ϕ_i) only once per vote profile $\boldsymbol{p} \in U$. Storing the values of the power indices (for all vote profiles) requires $\mathcal{O}(|U|)$ memory and retaining the final output uses $\mathcal{O}(n)$ memory. Since $|U| \leq n + 1$ in the binary classification setting, the total memory is $\mathcal{O}(|U| + n) = \mathcal{O}(n)$, concluding the proof. □

C.4 Proofs of Sec. 4.2.1

Proof of Lemma 4.4. Consider the expansion of the k-th order coefficient in $P_h(t)$ for $k \in [1, \ell - 1]$

$$\begin{aligned}
[t^k]P_h(t) &= [t^k] \prod_{j=2}^{\ell} (A_j(h) + B_j(h)t) \\
&= \sum_{\substack{I\subseteq\{2,\dots,\ell\} \\ |I|=k}} \left(\prod_{j\in I} B_j(h) \prod_{j\notin I} A_j(h)\right) \\
&= \sum_{\substack{I\subseteq\{2,\dots,\ell\} \\ |I|=k}} \left(\prod_{j\in I} \binom{n_j}{h} \prod_{j\notin I} \sum_{q=0}^{h-1} \binom{n_j}{q}\right)\,.
\end{aligned}$$

The first equality follows from the definition of $P_h(t)$, the second from the expansion of a product of binomial terms, and the third by the definitions of $A_j(h)$ and $B_j(h)$.

We now show that $|\mathcal{S}(k, h)|$ equals the above term via a combinatorial argument. By definition, $\mathcal{S}(k, h)$ contains subsets of classifiers $S \subseteq \mathcal{C}'$ that exclude the correct label y_1. For any $S \in \mathcal{S}(k, h)$, exactly k labels must have h votes, while all other labels $j \in \{2, \dots, \ell\}$ must have strictly fewer than h votes.

We can construct such coalitions by first selecting a subset of labels $I \subseteq \{2, \dots, \ell\}$ of size $|I| = k$ to attain h votes. For a fixed I, we calculate the number of choices T_I,

- **forcing $p_j(S) = h$ for $j \in I$:** we must select exactly h classifiers for each label in I. There are $\prod_{j \in I} \binom{n_j}{h}$ ways to do this;
- **forcing $p_j(S) < h$ for $j \notin I$:** for each label $j \in \{2, \dots, \ell\} \setminus I$, we must select $q \in \{0, \dots, h-1\}$ classifiers. Summing over possible values of q for a single label j gives $\sum_{q=0}^{h-1} \binom{n_j}{q} = A_j(h)$. Applying the product rule across all labels not in I gives $\prod_{j \notin I} A_j(h)$.

Thus, $T_I = \prod_{j \in I} \binom{n_j}{h} \prod_{j \notin I} A_j(h)$. Summing T_I over all possible subsets I of size k yields,

$$|\mathcal{S}(k,h)| = \sum_{\substack{I \subseteq \{2,\dots,\ell\} \\ |I|=k}} \left(\prod_{j \in I} \binom{n_j}{h} \prod_{j \notin I} \sum_{q=0}^{h-1} \binom{n_j}{q} \right) = [t^k] P_h(t)\,,$$

which concludes the proof. □

Proof of Corollary 4.5. By the definition of $P_h(t)$, factoring out the term for label y_j yields

$$\frac{P_h(t)}{A_j(h) + B_j(h)t} = \prod_{\substack{j'=2 \\ j' \neq j}}^{\ell} (A_{j'}(h) + B_{j'}(h)t)\,.$$

This expression is exactly the generating function from Lemma 4.4 applied to the ensemble $\mathcal{C}' \setminus \{C \in \mathcal{C} : C(x) = y_j\}$ and the label set $\mathcal{Y} \setminus \{y_1, y_j\}$. Since coalitions $S \in \mathcal{S}_{-j}(k,h)$ contain no classifiers with predictions for y_j, it follows that $p_j(S) = 0$. Applying an identical combinatorial counting argument as in Lemma 4.4 to the new product above yields the claim. □

Proof of Theorem 4.6. Let us first prove the claim for a correct classifier.

Correct classifier $(C_i(x) = y)$**.** Let $\mathcal{S}_1 = \{S \subseteq [n] : i \in S, S - i \neq \emptyset, |\mathcal{M}(S-i)| > 1, p_1(S-i) = p_j(S-i), j \in \mathcal{M}(S-i)\}$, and let $\mathcal{S}_2 = \{S \subseteq [n] : i \in S, S - i \neq \emptyset, |\mathcal{M}(S-i)| > 0, p_1(S-i) = p_j(S-i) - 1, j \in \mathcal{M}(S-i)\}$. Furthermore let $\mathcal{S}_1(k) = \{S \in \mathcal{S}_1 : |\mathcal{M}(S-i)| = k\}$ for $k \in [2, \ell]$ and $\mathcal{S}_2(k) = \{S \in \mathcal{S}_2 : |\mathcal{M}(S-i)| = k\}$ for $k \in [1, \ell-1]$. By Lemma C.2 we have that

$$2^{n-1}\beta_i = \sum_{i \in S \subseteq [n]} v_{\mathsf{MC}}(S) - v_{\mathsf{MC}}(S-i) \tag{7}$$

$$= 1 + \sum_{S \in \mathcal{S}_1} \frac{|\mathcal{M}(S-i)| - 1}{|\mathcal{M}(S-i)|} + \sum_{S \in \mathcal{S}_2} \frac{1}{|\mathcal{M}(S-i)| + 1} \tag{8}$$

$$= 1 + \sum_{k=2}^{\ell} \sum_{S \in \mathcal{S}_1(k)} \frac{k-1}{k} + \sum_{k=1}^{\ell-1} \sum_{S \in \mathcal{S}_2(k)} \frac{1}{k+1} \tag{9}$$

$$= 1 + \sum_{k=2}^{\ell} \sum_{h=1}^{n_1 - 1} \frac{k-1}{k} \binom{n_1 - 1}{h} |\mathcal{S}(k-1,h)| + \sum_{k=1}^{\ell-1} \sum_{h=0}^{n_1-1} \frac{1}{k+1} \binom{n_1-1}{h} |\mathcal{S}(k,h+1)| \tag{10}$$

$$= 1 + \sum_{h=1}^{n_1-1} \binom{n_1-1}{h} \sum_{k=1}^{\ell-1} \frac{k}{k+1} |\mathcal{S}(k,h)| + \sum_{h=1}^{n_1} \binom{n_1-1}{h-1} \sum_{k=1}^{\ell-1} \frac{1}{k+1} |\mathcal{S}(k,h)| \tag{11}$$

$$= 1 + \sum_{h=1}^{n_1-1} \sum_{k=1}^{\ell-1} \frac{|\mathcal{S}(k,h)|}{k+1} \left[k \binom{n_1-1}{h} + \binom{n_1-1}{h-1} \right] + \sum_{k=1}^{\ell-1} \frac{1}{k+1} |\mathcal{S}(k,n_1)| \tag{12}$$

$$= 1 + \sum_{h=1}^{n_1-1} \sum_{k=1}^{\ell-1} \frac{[x^k]P_h(x)}{k+1} \left[k \binom{n_1-1}{h} + \binom{n_1-1}{h-1} \right] + \sum_{k=1}^{\ell-1} \frac{[x^k]P_{n_1}(x)}{k+1} \tag{13}$$

$$= 1 + \sum_{h=1}^{n_1} \sum_{k=1}^{\ell-1} \frac{[x^k]P_h(x)}{k+1} \left[k \binom{n_1-1}{h} + \binom{n_1-1}{h-1} \right]. \tag{14}$$

In the above derivation, Eq. (7) holds is by definition of Banzhaf value and Eq. (8) follows by Lemma C.2. Eq. (9) holds by the definition of $\mathcal{S}_1(k)$ and $\mathcal{S}_2(k)$, noting that they form a partition of $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. Eq. (10) follows from a combinatorial selection to count sets from $\mathcal{S}_1(k)$ and $\mathcal{S}_2(k)$: we first choose the number of correct classifiers from $\{C : C(x) = y\}$ and combine it with the definition of $\mathcal{S}(k, h)$. Eqs (11), (12) and (13) follow from simple manipulations, including re-indexing and carrying out boundary terms. Finally, the claim in Eq. (14) follows from the fact that $f(k, n_1) = 1$ since $\binom{n_1-1}{n_1} = 0$.

We now prove the remaining part of the claim for an incorrect classifier.

Incorrect classifier ($C_i(x) = y_j, j \neq 1$). We now let $\mathcal{S}_3 = \{S \subseteq [n] : i \in S, S - i \neq \emptyset, |\mathcal{M}(S - i)| \geq 2, p_1(S - i) = p_j(S - i) = p_{j'}(S - i), j' \in \mathcal{M}(S - i)\}$, and $\mathcal{S}_4 = \{S \subseteq [n] : i \in S, S - i \neq \emptyset, |\mathcal{M}(S - i)| \geq 1, p_j(S - i) + 1 = p_1(S - i) = p_{j'}(S - i), j' \in \mathcal{M}(S - i)\}$. In analogy with the previous case we partition the two sets above as $\mathcal{S}_3(k) = \{S \in \mathcal{S}_3 : |\mathcal{M}(S - i)| = k\}$ for $k \in [2, \ell]$ and $\mathcal{S}_4(k) = \{S \in \mathcal{S}_4 : |\mathcal{M}(S - i)| = k\}$ for $k \in [1, \ell - 1]$. We can compute the Banzhaf value as

$$-2^{n-1}\beta_i = -\sum_{i \in S \subseteq [n]} v_{\mathsf{MC}}(S) - v_{\mathsf{MC}}(S - i) \tag{15}$$

$$= \sum_{S \in \mathcal{S}_3} \frac{1}{|\mathcal{M}(S - i)|} + \sum_{S \in \mathcal{S}_4} \frac{1}{|\mathcal{M}(S - i)|(|\mathcal{M}(S - i)| + 1)} \tag{16}$$

$$= \sum_{k=2}^{\ell} \sum_{S \in \mathcal{S}_3(k)} \frac{1}{k} + \sum_{k=1}^{\ell-1} \sum_{S \in \mathcal{S}_4(k)} \frac{1}{k(k+1)} \tag{17}$$

$$= \sum_{k=2}^{\ell} \sum_{h=1}^{n_j - 1} \frac{1}{k} \binom{n_j - 1}{h} \binom{n_1}{h} |\mathcal{S}_{-j}(k - 2, h)|$$

$$+ \sum_{k=1}^{\ell-1} \sum_{h=0}^{n_j - 1} \binom{n_j - 1}{h} \binom{n_1}{h+1} |\mathcal{S}_{-j}(k - 1, h + 1)| \frac{1}{k(k+1)} \tag{18}$$

$$= \sum_{h=1}^{n_j} \sum_{k=0}^{\ell-2} \binom{n_j - 1}{h} \binom{n_1}{h} \frac{|\mathcal{S}_{-j}(k, h)|}{k + 2} + \sum_{h=1}^{n_j} \sum_{k=0}^{\ell-2} \binom{n_j - 1}{h - 1} \binom{n_1}{h} \frac{|\mathcal{S}_{-j}(k, h)|}{(k + 1)(k + 2)} \tag{19}$$

$$= \sum_{h=1}^{n_j} \sum_{k=0}^{\ell-2} \frac{|\mathcal{S}_{-j}(k, h)|}{k + 2} \binom{n_1}{h} \left[\binom{n_j - 1}{h} + \frac{1}{k + 1} \binom{n_j - 1}{h - 1} \right] \tag{20}$$

$$= \sum_{h=1}^{n_j} \sum_{k=0}^{\ell-2} \frac{f_j(k, h)}{k + 2} [t^k] \frac{P_h(t)}{A_j(h) + B_j(h)t} \,. \tag{21}$$

In the above derivations, Eq. (15) follows by Def. 3.2. Eq. (16) follows by Lemma C.2. Eq. (17) follows by noting that $\mathcal{S}_3(k)$ and $\mathcal{S}_4(k)$ form a partition of $\mathcal{S}_3$ and $\mathcal{S}_4$ respectively. Eq. (18) follows by counting the sets in $\mathcal{S}_3(k)$ and $\mathcal{S}_4(k)$ with a combinatorial argument. We now discuss how to build sets from $\mathcal{S}_3(k)$. That is, (i) we first select the $h \geq 1$ classifiers voting for y_j for which we have $\binom{n_j - 1}{h}$ possible choice; (ii) we then select the number of correct classifiers to match the h classifiers in $S - i$ selected, for which we have $\binom{n_1}{h}$ possible choices (iii) for any $k \geq 2$ we have to fix $k - 2$ labels to tie the h votes for y_j and y_1, and all remaining labels (with possibly no votes) to have strictly less than h votes. This corresponds exactly to the definition of $|\mathcal{S}_{-j}(k - 2, h)|$. Summing over all possible values of h for fixed k yields the first part of Eq. (18). The sets of $\mathcal{S}_4(k)$ follow an analogous construction yielding Eq. (18). Finally, Eqs (19) and (20) follow by simple algebraic manipulations and summation re-indexing. Finally, Eq. (21) follows by the definition of $f_j(k, h)$. This concludes the proof. □

Proof of Theorem 4.7. Consider two data points $d_1 = (x_1, y_1)$ and $d_2 = (x_2, y_2)$ with identical vote profiles. By the symmetry axiom, and as captured by our Theorem 4.6, the values of $\beta_i(G_{d_1})$ and $\beta_i(G_{d_2})$ depend strictly on the values n_j for $j \in [\ell]$. That is, under two identical vote profiles $\boldsymbol{p}^{d_1} = \boldsymbol{p}^{d_2}$, it holds that $\beta_i(G_{d_1}) = \beta_{i'}(G_{d_2})$ if either both i and i' vote for the correct label, or

if $C_i(x_1) = y_j$ and $C_{i'}(x_2) = y_{j'}$ where $j, j' \neq 1$ and $n_j = n_{j'}$. This observation allows us to compute the Banzhaf values exactly once per unique vote profile, caching the result and reusing it in $\mathcal{O}(1)$ time for any two points with the same vote profile.

For a unique vote profile of (x, y), the algorithm first obtains $n_j, j \in [\ell]$ in $\mathcal{O}(n)$ time. Then, the algorithm iterates over $h \in [n_1]$. Inside this loop, it computes the coefficients of the polynomial $P_h(t)$ in $\mathcal{O}(\ell^2)$ time. The contribution to β_i for any correct classifier provided by $P_h(t)$ is computed in $\mathcal{O}(\ell)$ time. Next, for each incorrect class $j \in [2, \ell]$, the algorithm obtains the polynomial $Q_h^j(t) \doteq P_h(t)/(A_j(h) + B_j(h)t)$ in $\mathcal{O}(\ell)$ time. Evaluating the contribution to β_i for any classifier voting for y_j in $\mathcal{O}(\ell)$ time. By caching $P_h(t)$ the algorithm can process all incorrect classes in $\sum_{j=2}^{\ell} \mathcal{O}(\ell)$ time. After processing each unique profile of U, we iterate through all classifiers correcting for the factor 2^{n-1} as shown by Theorem 4.6, which takes $\mathcal{O}(n)$ per vote profile. Finally, the output corrects for the factor $|\mathcal{D}|$, using the linearity property of power indices.

The overall time complexity is bounded by

$$\sum_{u=1}^{|U|} \left(\mathcal{O}(n) + \sum_{h=1}^{n_1} \left[\mathcal{O}(\ell^2) + \mathcal{O}(\ell) + \sum_{j=2}^{\ell} \mathcal{O}(\ell) \right] \right) + \mathcal{O}(|\mathcal{D}|n) = \mathcal{O}(|U|n\ell^2 + |\mathcal{D}|n)\,,$$

where the $\mathcal{O}(|\mathcal{D}|n)$ is required to evaluate the predictions of all classifiers of $\mathcal{C}$. If predictions are provided in input, e.g., through a preprocessing, then such term can be dropped.

Finally, the algorithm requires no more than $\mathcal{O}(|U|\ell + n + \ell n) = \mathcal{O}(\ell(n + |U|))$ auxiliary space. That is, 1. $\mathcal{O}(|U|\ell)$ is required to store the dictionaries of unique vote profiles and the Banzhaf values of their game; 2. $\mathcal{O}(n)$ is necessary to accumulate the global output for the ensemble; 3. finally $\mathcal{O}(\ell)$ is required to store the coefficients of the polynomial $P_h(t)$ for any iteration $h \geq 1$, which concludes the proof. □

C.5 Proofs of Sec. 4.2.2

Lemma C.5. *Let $(x, y) \in \mathcal{D}$ be a test data point. Without loss of generality, let $y = y_1 \in \mathcal{Y}$. For any $h \geq 1$ and $j \in [2, \ell]$, we define the two polynomials*

$$A_j(h; u) = \sum_{q=0}^{h-1} \binom{n_j}{q} u^q \quad \textit{and} \quad B_j(h; u) = \binom{n_j}{h} u^h\,.$$

Consider the ordinary generating function $P_h(t, u) = \prod_{j=2}^{\ell} (A_j(h; u) + tB_j(h; u))$.

Fix $h \in [n]$, $k \in \{1, \ldots, \ell - 1\}$, and $m \geq 0$. Let $\mathcal{S}(k, h, m)$ be the set of coalitions $S \subseteq \mathcal{C}'$ such that: (i) $\mathcal{C}' = \mathcal{C} \setminus \{C \in \mathcal{C} : C(x) = y\}$; (ii) $|\mathcal{M}(S)| = k$; (iii) $p_j(S) = h$ for each $j \in \mathcal{M}(S)$; and (iv) the number of classifiers in S is exactly $|S| = m$. Then it holds that

$$|\mathcal{S}(k, h, m)| = [t^k u^m] P_h(t, u)\,.$$

Proof. We now expand $P_h(t, u)$. To do so we first extract the k-th coefficient in t. That is, analogously to the proof of Lemma 4.4 it holds that

$$\begin{aligned}
[t^k u^m] P_h(t, u) &= [u^m] \sum_{\substack{I \subseteq \{2, \ldots, \ell\} \\ |I| = k}} \left(\prod_{j \in I} B_j(h; u) \prod_{j \notin I} A_j(h; u) \right) \\
&= [u^m] \sum_{\substack{I \subseteq \{2, \ldots, \ell\} \\ |I| = k}} \left(\prod_{j \in I} \binom{n_j}{h} u^h \prod_{j \notin I} \sum_{q_j = 0}^{h-1} \binom{n_j}{q_j} u^{q_j} \right) \\
&= \sum_{\substack{I \subseteq \{2, \ldots, \ell\} \\ |I| = k}} \left(\prod_{j \in I} \binom{n_j}{h} [u^{m-kh}] \prod_{j \notin I} \sum_{q_j = 0}^{h-1} \binom{n_j}{q_j} u^{q_j} \right)
\end{aligned}$$

$$= \sum_{\substack{I \subseteq \{2,\dots,\ell\} \\ |I|=k}} \left(\prod_{j \in I} \binom{n_j}{h} \sum_{\substack{q_j \in \{0,\dots,h-1\} \text{ for all } j \notin I \\ \sum_{j \notin I} q_j = m - kh}} \prod_{j \notin I} \binom{n_j}{q_j} \right).$$

The first equality extracts the coefficient of $[t^k]$, and the second equality expands terms A and B by their definition. The third equality follows by noting that $\prod_{j \in I} u^h = u^{kh}$. Therefore, the extraction of $[u^m]$ corresponds to extracting $[u^{m-kh}]$ from the polynomial $\prod_{j \notin I} \sum_{q_j=0}^{h-1} \binom{n_j}{q_j} u^{q_j}$. Finally, the last equality follows by the definition of polynomial multiplication (Cauchy product). We now show that $|\mathcal{S}(k,h,m)|$ corresponds to the above term by a combinatorial argument. By definition, $\mathcal{S}(k,h,m)$ contains subsets $S \subseteq \mathcal{C}'$, where classifiers voting for the correct label are excluded from $\mathcal{C}'$. For any $S \in \mathcal{S}(k,h,m)$, exactly k labels must receive exactly h votes, all other incorrect labels must receive strictly fewer than h votes, and the total number of classifiers in S must sum to m.

We now construct a coalition from $\mathcal{S}(k,h,m)$ as follows. We first select a subset of labels $I \subseteq \{2,\dots,\ell\}$ of size $|I| = k$ corresponding to the k labels receiving exactly h votes. For a fixed I, we compute the number of sets that belong to $\mathcal{S}(k,h,m)$ as follows

- **forcing $p_j(S) = h$ for $j \in I$:** we select exactly h classifiers for each label in I. This yields $\prod_{j \in I} \binom{n_j}{h}$ possible combinations. Note that, the selection of such classifiers collectively contributes to kh classifiers in the final set $S \in \mathcal{S}(k,h,m)$.
- **forcing $p_j(S) < h$ for $j \notin I$:** for each remaining incorrect label $j \in \{2,\dots,\ell\} \setminus I$, we must select $q_j \in \{0,\dots,h-1\}$ classifiers. For a specific sequence of choices $(q_j)_{j \notin I}$, there are $\prod_{j \notin I} \binom{n_j}{q_j}$ ways to select the classifiers. Note that, given the additional constrain of $|S| = m$, then it must hold that a valid selection yields $\sum_{j \notin I} q_j = m - \sum_{j \in I} k = m - kh$.

Therefore, by summing over all possible subsets I we get

$$|\mathcal{S}(k,h,m)| = \sum_{\substack{I \subseteq \{2,\dots,\ell\} \\ |I|=k}} \left(\prod_{j \in I} \binom{n_j}{h} \sum_{\substack{q_j \in \{0,\dots,h-1\} \text{ for all } j \notin I \\ \sum_{j \notin I} q_j = m - kh}} \prod_{j \notin I} \binom{n_j}{q_j} \right) = [t^k u^m] P_h(t,u)\,,$$

concluding the proof. □

Corollary C.6. *Under the setting of Lemma C.5, fix $j \in \{2,\dots,\ell\}$. Let $\mathcal{S}_{-j}(k,h,m)$ be defined analogously to $\mathcal{S}(k,h,m)$ for coalitions $S \subseteq \mathcal{C}' \setminus \{C \in \mathcal{C} : C(x) = y_j\}$. That is, a coalition S from $\mathcal{S}_{-j}(k,h,m)$ contains no classifier voting for class y_j. Then it holds that*

$$|\mathcal{S}_{-j}(k,h,m)| = [t^k u^m] \frac{P_h(t,u)}{A_j(h;u) + B_j(h;u)t}\,.$$

Proof. The proof follows an analogous argument to that of Corollary 4.5. By the definition of $P_h(t,u)$, factoring out the term for label $y_j, j \in [2,\ell]$ yields

$$\frac{P_h(t,u)}{A_j(h;u) + B_j(h;u)t} = \prod_{\substack{j'=2 \\ j' \neq j}}^{\ell} (A_{j'}(h;u) + B_{j'}(h;u)t)\,.$$

This expression is exactly the generating function from Lemma C.5 applied to the ensemble $\mathcal{C}' \setminus \{C \in \mathcal{C} : C(x) = y_j\}$ and the label set $\mathcal{Y} \setminus \{y_1, y_j\}$. Since coalitions $S \in \mathcal{S}_{-j}(k,h,m)$ contain no classifiers with predictions for y_j, it follows that $p_j(S) = 0$. Applying an identical combinatorial counting argument as in Lemma C.5 to the new product above yields the claim. □

Proof of Theorem 4.8. Let us first prove the claim for a correct classifier.

Correct classifier $(C_i(x) = y)$**.** Similarly to the proof of Theorem 4.6. Let $\mathcal{S}_1 = \{S \subseteq [n] : i \in S, S - i \neq \emptyset, |\mathcal{M}(S-i)| > 1, p_1(S-i) = p_j(S-i), j \in \mathcal{M}(S-i)\}$, and let $\mathcal{S}_2 = \{S \subseteq [n] : i \in S, S - i \neq \emptyset, |\mathcal{M}(S-i)| > 0, p_1(S-i) = p_j(S-i) - 1, j \in \mathcal{M}(S-i)\}$. We have that,

$$\phi_i = \sum_{i \in S \subseteq \mathcal{C}} \eta(|S|)(v(S) - v(S-i)) \tag{22}$$

$$= \frac{1}{n} + \sum_{S \in \mathcal{S}_1} \eta(|S|) \frac{|\mathcal{M}(S-i)| - 1}{|\mathcal{M}(S-i)|} + \sum_{S \in \mathcal{S}_2} \eta(|S|) \frac{1}{|\mathcal{M}(S-i)| + 1} \tag{23}$$

$$= \frac{1}{n} + \sum_{k=2}^{\ell} \sum_{h=1}^{n_1 - 1} \sum_{m=(k-1)h}^{n-n_1} \frac{k-1}{k} \eta(m+h+1) \binom{n_1 - 1}{h} |\mathcal{S}(k-1, h, m)|$$
$$+ \sum_{k=1}^{\ell-1} \sum_{h=0}^{n_1 - 1} \sum_{m=k(h+1)}^{n-n_1} \frac{1}{k+1} \eta(m+h+1) \binom{n_1 - 1}{h} |\mathcal{S}(k, h+1, m)| \tag{24}$$

$$= \frac{1}{n} + \sum_{k=1}^{\ell-1} \sum_{h=1}^{n_1 - 1} \sum_{m=kh}^{n-n_1} \frac{k}{k+1} \eta(m+h+1) \binom{n_1 - 1}{h} |\mathcal{S}(k, h, m)|$$
$$+ \sum_{k=1}^{\ell-1} \sum_{h=1}^{n_1} \sum_{m=kh}^{n-n_1} \frac{1}{k+1} \eta(m+h) \binom{n_1 - 1}{h-1} |\mathcal{S}(k, h, m)| \tag{25}$$

$$= \frac{1}{n} + \sum_{k=1}^{\ell-1} \sum_{h=1}^{n_1} \sum_{m=kh}^{n-n_1} \frac{1}{k+1} \left(k\eta(m+h+1) \binom{n_1 - 1}{h} + \eta(m+h) \binom{n_1 - 1}{h-1} \right) |\mathcal{S}(k, h, m)| \tag{26}$$

$$= \frac{1}{n} + \sum_{k=1}^{\ell-1} \sum_{h=1}^{n_1} \sum_{m=kh}^{n-n_1} f_1(k, h, m)[t^k u^m] P_h(t, u)\,, \tag{27}$$

where Eq. (22) follows by the definition of Shapley (Eq. (1)), Eq. (23) follow by Lemma C.2, and the definition of $\mathcal{S}_1$ and $\mathcal{S}_2$. Eq. (24) follows by summing over all sets of $\mathcal{S}_1$ and $\mathcal{S}_2$. We show how to account for the summation over the sets in $\mathcal{S}_1$, the summation accounting for sets in $\mathcal{S}_2$ follows an analogous formulation. That is, (i) first we fix the number of classes in $\mathcal{M}(S-i)$ that vote for an incorrect label (i.e., k); (ii) for each incorrect label we fix the cardinality of the classifiers voting for it (i.e., h) (iii) we then weight the number of sets $S \in \mathcal{S}_1$ with $i \in S$ with the product rule, by first selecting $\binom{n_1 - 1}{h}$ classifiers voting for the correct label, excluding i and multiplying it by $|\mathcal{S}(k-1, h, m)|$. Therefore, this corresponds to sets $S \in \mathcal{S}_1$ where there are h classifiers voting for the correct label, k labels attaining h votes over $\boldsymbol{p}(S)$ and all remaining labels having strictly less than h votes. Finally, by the definition of $|\mathcal{S}(k-1, h, m)|$ each set has exactly cardinality $(m+h+1)$. Combining all the above each set must be weighted $\eta(m+h+1)(k-1)/k$ to retain correctness. Eq. (25) follows by re-indexing the summations. Eq. (26) follows by grouping the two summations. Finally, Eq. (27) follows by the definition of $f_1(k, h, m)$ combined with Lemma C.5.

Incorrect classifier $(C_i(x) = y_j, j \neq 1)$**.** We now let $\mathcal{S}_3 = \{S \subseteq [n] : i \in S, S - i \neq \emptyset, |\mathcal{M}(S-i)| \geq 2, p_1(S-i) = p_j(S-i) = p_{j'}(S-i), j' \in \mathcal{M}(S-i)\}$, and $\mathcal{S}_4 = \{S \subseteq [n] : i \in S, S - i \neq \emptyset, |\mathcal{M}(S-i)| \geq 1, p_j(S-i) + 1 = p_1(S-i) = p_{j'}(S-i), j' \in \mathcal{M}(S-i)\}$.

$$-\phi_i = -\sum_{i \in S \subseteq \mathcal{C}} \eta(|S|)(v(S) - v(S-i)) \tag{28}$$

$$= \sum_{S \in \mathcal{S}_3} \frac{\eta(|S|)}{|\mathcal{M}(S-i)|} + \sum_{S \in \mathcal{S}_4} \frac{\eta(|S|)}{|\mathcal{M}(S-i)|(|\mathcal{M}(S-i)| + 1)} \tag{29}$$

$$= \sum_{k=2}^{\ell} \sum_{h=1}^{n_j - 1} \sum_{m=h(k-2)}^{n-(n_1+n_j)} \frac{1}{k} \eta(m+2h+1) \binom{n_1}{h} \binom{n_j - 1}{h} |\mathcal{S}_{-j}(k-2, h, m)|$$

$$+\sum_{k=1}^{\ell-1}\sum_{h=0}^{n_j-1}\sum_{m=(k-1)(h+1)}^{n-(n_1+n_j)}\frac{1}{k(k+1)}\eta(m+2(h+1))\binom{n_1}{h+1}\binom{n_j-1}{h}|\mathcal{S}_{-j}(k-1,h+1,m)| \tag{30}$$

$$=\sum_{k=0}^{\ell-2}\sum_{h=1}^{n_j-1}\sum_{m=hk}^{n-(n_1+n_j)}\frac{1}{k+2}\eta(m+2h+1)\binom{n_1}{h}\binom{n_j-1}{h}|\mathcal{S}_{-j}(k,h,m)|$$
$$+\sum_{k=0}^{\ell-2}\sum_{h=1}^{n_j}\sum_{m=hk}^{n-(n_1+n_j)}\frac{1}{(k+1)(k+2)}\eta(m+2h)\binom{n_1}{h}\binom{n_j-1}{h-1}|\mathcal{S}_{-j}(k,h,m)| \tag{31}$$

$$=\sum_{k=0}^{\ell-2}\sum_{h=1}^{n_j}\sum_{m=hk}^{n-(n_1+n_j)}f_j(k,h,m)[t^k u^m]\frac{P_h(t,u)}{A_j(h;u)+B_j(h;u)t}, \tag{32}$$

$$\tag{33}$$

where Eq. (28) follows by Eq. (1). Eq. (29) follows by Lemma C.2 and the definitions of $\mathcal{S}_3$ and $\mathcal{S}_4$. Eq. (30) follows by a combinatorial selection of the sets in $\mathcal{S}_3$ and $\mathcal{S}_4$, with an analogous argument detailed for the case of correct classifiers, combined with the definition of $|\mathcal{S}_{-j}(k,h,m)|$. Eq. (31) follows by re-indexing the summations. Finally, Eq. (32) follows by grouping the two summations noting that $\binom{n_j-1}{n_j}=0$, the definition of $f_j, j\in[2,\ell]$ and Corollary C.6. □

Proof of Theorem 4.9. As shown in the proof of Theorem 4.7 we only need to compute the values of ϕ_i for all classifiers $i\in[n]$ yielding a unique vote profile $\boldsymbol{p}\in U$, by symmetry.

Fix a data point (x,y) the our Algorithm 4 first evaluates the predictions of all classifiers in $\mathcal{O}(n)$ time, and applies the mapping such that $y=y_1$ in $\mathcal{O}(\ell)$ time. Note that any label y_j with $n_j=0$ can be ignored, since $A_j(h;u)=1$ and $B_j(h;u)=0$ for $h\geq 1$, i.e., its factor in $P_h(t,u)$ is 1. Thus, we assume $n_j\geq 1$ for $j\in[2,\ell]$. Algorithm 4 then iterates over $h\in[n_1]$ initializing a matrix P_h of size $\ell\times(n-n_1+1)$, which contains the coefficient of the polynomial $P_h(t,u)$. All matrices $P_h, h\geq 1$ can be computed within a time complexity bounded by

$$\sum_{h=1}^{n_1}\sum_{j=2}^{\ell}\sum_{i=0}^{j-2}\sum_{m=0}^{\sum_{j'=2}^{j-1}n_{j'}}\mathcal{O}(\min(n_j,h))\leq\sum_{h=1}^{n_1}\sum_{j=2}^{\ell}\mathcal{O}(\ell(n-n_1)n_j)$$
$$=\mathcal{O}(n_1\ell(n-n_1))\sum_{j=2}^{\ell}n_j$$
$$=\mathcal{O}(n_1\ell(n-n_1)^2)=\mathcal{O}(\ell n^3)\,,$$

where the first step follows by the independence of the summations on m and i, and by the facts that $j-2\leq\ell$; $\sum_{j'=2}^{j-1}n_{j'}\leq\sum_{j'=2}^{\ell}n_{j'}=n-n_1$; and that $\min(a,b)\leq a$. The first equality follows by the linearity of the sum, and the second equality follows again by $\sum_{j'=2}^{\ell}n_{j'}=n-n_1$. Clearly the last step bounds $(n-n_1)\leq n$ and $n_1\leq n$. For each matrix P_h the algorithm computes the Shapley values of a correct classifier in $\mathcal{O}(\ell n)$ time, using the result of Theorem 4.8. For each incorrect label, the algorithm computes the coefficients of the polynomial $Q_j(t,u)\doteq P_h(t,u)/(A_j(h;u)+B_j(h;u)t)$ updating the matrix P_h in place with a time complexity bounded by $\mathcal{O}(\ell(n-n_1)n_j)$, it then computes the related Shapley value in $\mathcal{O}(\ell n)=\mathcal{O}(\ell n n_j)$ time, and it then restores the matrix P_h adding class $j\in[2,\ell]$ in $\mathcal{O}(\ell(n-n_1)n_j)$ time. Therefore, summing over $j\in[2,\ell]$ and using $\sum_{j=2}^{\ell}n_j=n-n_1$, we obtain a time complexity bounded by $\mathcal{O}(\ell n^2)$ time for each h, and $\mathcal{O}(\ell n^3)$ over all $h\in[n_1]$. Therefore, the time complexity of Algorithm 4 is bounded by $\mathcal{O}(|U|n^3\ell+|\mathcal{D}|n)$. The memory complexity is bounded by $\mathcal{O}(|U|\ell+\ell n)=\mathcal{O}(\ell(|U|+n))$. □

Algorithm 1: ENPOWER$_{\mathsf{B}}$: exact computation of Banzhaf and Shapley values for binary classifiers

Input: $\psi \in \{\beta, \phi\}$; ensemble $\mathcal{C} = \{C_1, \ldots, C_n\}$; dataset $\mathcal{D}$

1 **for** $i \in [n]$ **do** $\psi_i(G_{\mathcal{D}}) \leftarrow 0$;

2 **foreach** $(x, y) \in \mathcal{D}$ **do**

3 $\quad n_+ \leftarrow \sum_{C \in \mathcal{C}} \mathbb{1}[C(x) = y]$;

4 $\quad$ **for** $i \leftarrow 1$ **to** n **do**

5 $\quad\quad \psi_i(G_{\mathcal{D}}) \leftarrow \psi_i(G_{\mathcal{D}}) + \texttt{Get_}\psi(n, n_+, C_i(x), y)$;

6 **return** $\left\{\frac{\psi_i(G_{\mathcal{D}})}{|\mathcal{D}|}\right\}_{i \in [n]}$

Algorithm 2: `Get-`$\beta(n, n_+, C_i(x), y)$ and `Get-`$\phi(n, n_+, C_i(x), y)$

Input: number of classifiers n, number of correct predictions n_+, prediction $C_i(x)$, label y

Procedure `Get-`$\beta(n, n_+, C_i(x), y)$

$\quad n_- \leftarrow n - n_+$;

$\quad$ **if** $C_i(x) = y$ **then**

$\quad\quad$ **return** $\left(\binom{n}{n_-} + 1\right) 2^{-n}$;

$\quad$ **else**

$\quad\quad$ **return** $-\left(\binom{n}{n_+} - 1\right) 2^{-n}$;

Procedure `Get-`$\phi(n, n_+, C_i(x), y)$

$\quad a \leftarrow 0$; $b \leftarrow 0$; $\phi_i \leftarrow 0$;

$\quad$ **if** $C_i(x) = y$ **then**

$\quad\quad a \leftarrow \dfrac{(n-2n_+)!(2n_+-1)!}{n!}\binom{n_-}{n_+}$;

$\quad\quad b \leftarrow \min\{n_-, n_+ - 1\}$;

$\quad\quad \phi_i \leftarrow \dfrac{1}{n} + \dfrac{1}{2}\left(a + \sum_{h=1}^{b}\binom{n_+-1}{h-1}\binom{n_-}{h}\dfrac{(n-2h-1)!(2h-1)!}{n!}(2n_+ + n - 4h)\right)$;

$\quad$ **else**

$\quad\quad a \leftarrow \dfrac{(n-2n_-)!(2n_--1)!}{n!}\binom{n_+}{n_-}$;

$\quad\quad b \leftarrow \min\{n_+, n_- - 1\}$;

$\quad\quad \phi_i \leftarrow -\dfrac{1}{2}\left(a + \sum_{h=1}^{b}\binom{n_--1}{h-1}\binom{n_+}{h}\dfrac{(n-2h-1)!(2h-1)!}{n!}(2n_- + n - 4h)\right)$;

$\quad$ **return** ϕ_i;

D Pseudocode

D.1 Binary classification

Computation of Banzhaf and Shapley values. Pseudocode for our main algorithm, which is presented in Sec. 4.1, is shown in Algorithm 1,

We detail the missing subroutine `Get-`ψ of Algorithm 1 in Algorithm 2. That is, Algorithm 2 simply computes the Banzhaf (when $\psi = \beta$) or Shapley (when $\psi = \phi$) according to Theorem 4.1 and Theorem 4.2 respectively.

We remark that in practice we do not execute the loop in Line 4 of Algorithm 1. That is, we need to execute `Get-`ψ exactly twice for any point $(x, y) \in \mathcal{D}$ of the dataset $\mathcal{D}$: the first execution is to obtain the values of the power index on $v_{\mathsf{B}}(x)$ for all correct models and the second execution is to obtain the remaining values. That is, by symmetry all correct models will have the same power index on the game $G = (\mathcal{C}, v_{\mathsf{B}}(x))$, and similarly for all the incorrect models. We discuss how to implement efficiently our method in App. E.

D.2 Multiclass classification

Banzhaf computation over v_{MC} (ENPOWER$_{\mathsf{MC}}$-β). We detail the pseudocode of our approach that computes the values of $\beta_i(G_{\mathcal{D}})$ in Algorithm 3. The algorithm proceeds by processing each element independently in the loop from Line 2. It applies the relabeling discussed in Sec. 4.2 from Line 4 to Line 7, setting class 1 as the correct one and all the remaining classes as incorrect ones, we use $D[\cdot]$ to denote a dictionary data structure. The main loop of the algorithm from Line 10 simply computes the power indices as dictated by Theorem 4.6. Note that, while obtaining the coefficients of P_h, with our approach, requires $\mathcal{O}(\ell^2)$ (Line 12), we compute the term $A_j(h)$ (Line 11) using $A_{j-1}(h)$, i.e., avoiding to recompute the summation for each value of h. The loop in Line 14 computes the Banzhaf values for all incorrect models, according to the class they voted to. We then update the Banzhaf values of each model in Line 16 according to the class for which each model voted. The final output in Line 20 reports the average of the values as captured by the definition of $G_{\mathcal{D}}$.

Note that our actual implementation does not compute the power indices β_i all points in $\mathcal{D}$ as we apply the symmetry property to process only the points in $\mathcal{D}$ yileding *unique* vote profiles as discussed in Sec. 3. We detail how to process only the unique vote profiles in the pseudocode of ENPOWER$_{\mathsf{MC}}$-ϕ that we will discuss next.

Shapley computation over v_{MC} (ENPOWER$_{\mathsf{MC}}$-ϕ). Algorithm 4 displays the pseudocode of ENPOWER$_{\mathsf{MC}}$-ϕ computing the Shapley values for the multiclass case implementing our theoretical results of Theorem 4.8.

The algorithm is similar to Algorithm 3, while slightly more complex due to the introduction of the bivariate polynomial from Theorem 4.8. We now detail the main differences with respect to Algorithm 3. First, as mentioned previously, and as captured by our theoretical results we need to compute the Shapley values only for the points yileding unique vote profiles. To do so, we use a dictionary J_{unique} (Line 2) which caches the vote profiles seen so far and the Shapley values for the specific game (Line 27). If the vote profile has been observed we can simply use the cached values, and update the global Shapley values of all points (Line 10 and Line 28). Otherwise the algorithm processes the vote profile, by implementing Theorem 4.8. In Line 16 we compute P_h with a dynamic programming approach storing the coefficients of the bivariate polynomial necessary to compute the Shapley values for all models.

The dynamic programming approach is detailed in Algorithm 5. Such approach computes a matrix $P_h[k,m]$ containing all coefficients of the bivariate polynomial $P_h(t,u)$. That is, each row $k \in [0,\ell-1]$ will correspond to the coefficients of the polynomial in u after the extraction of the k-th coefficient in t (i.e., $[t^k]P_h(t,u)$). In other words, the entry $P_h[k,m]$ contains the counts of all coalitions of $\mathcal{S}(k,h,m)$. The algorithm processes one incorrect class $j \in [2,\ell]$ at a time, computing the number of sets that such class contributes towards the coefficients of $P_h(t,u)$. To detail the recursion used by the algorithm, for $j \in [\ell]$ let[5]

$$P_h^{(j)}(t,u) \doteq \prod_{j'=2}^{j} \left(A_{j'}(h;u) + B_{j'}(h;u)t\right) \ ,$$

be the partial product over the first $j-1$ incorrect classes (arbitrarily labelled). We define $P_h^{(1)}(t,u) = 1$ to correspond to the empty product. Note that $P_h^{(\ell)}(t,u) = P_h(t,u)$. We also let $n^{(j)} \doteq \sum_{j'=2}^{j} n_{j'}$ be the number of classifiers voting for one of such classes. The next lemma formalizes the recursion used in Algorithm 5, and proves its correctness.

Lemma D.1. *Fix $h \geq 1$, for $j \in [2,\ell]$, $k \in [0,\ell-1]$ and $m \in [0,n^{(j)}]$, let*

$$P_h^{(j)}[k,m] = \sum_{q=0}^{\min(n_j,h-1,m)} \binom{n_j}{q} P_h^{(j-1)}[k,m-q] + \mathbb{1}[k \geq 1 \wedge m \geq h] \binom{n_j}{h} P_h^{(j-1)}[k-1,m-h] \ , \tag{34}$$

where $P_h^{(1)}[k,m] \doteq \mathbb{1}[k=0 \wedge m=0]$, and $P_h^{(j)}[k,m] \doteq 0$ for $m > n^{(j)}$, as $P_h^{(j)}(t,u)$ has degree at most $n^{(j)}$ in u. Then, for each $j = 1,\dots,\ell$, $k \in [0,\ell-1]$ and $m \in [0,n^{(j)}]$ it holds that

$$P_h^{(j)}[k,m] = [t^k u^m] P_h^{(j)}(t,u) \ .$$

[5]We use the convention that a product $\prod_{i=a}^{b} x_i$ for $b < a$ is empty and evaluates to 1.

Proof. We proceed by induction on $j \geq 1$, i.e., the number of incorrect classes evaluated by the recursion.

Base case ($j = 1$). By definition $P_h^{(1)}(t,u)$ is the empty product, i.e., $P_h^{(1)}(t,u) = 1 = t^0u^0$. Therefore, for each $k \in [0, \ell-1]$ and $m \geq 0$ it holds that $[t^k u^m]P_h^{(1)}(t,u) = \mathbb{1}[k = 0 \wedge m = 0] = P_h^{(1)}[k,m]$.

Inductive step. Fix $j \in [2, \ell]$ and assume that $P_h^{(j-1)}[k,m] = [t^k u^m]P_h^{(j-1)}(t,u)$ holds for each $k \in [0, \ell-1]$ and $m \geq 0$, where both sides equal 0 for $m > n^{(j-1)}$, since $P_h^{(j-1)}(t,u)$ has degree at most $n^{(j-1)}$ in u. By definition of $P_h^{(j)}(t,u)$ we have $P_h^{(j)}(t,u) = P_h^{(j-1)}(t,u)\,(A_j(h;u) + B_j(h;u)t)$. Note that terms $A_j(h;u)$ and $B_j(h;u)$ do not depend on t, thus for $k \in [0, \ell-1]$ extracting the coefficient of t^k yields

$$[t^k]P_h^{(j)}(t,u) = A_j(h;u)\,[t^k]P_h^{(j-1)}(t,u) + B_j(h;u)\,[t^{k-1}]P_h^{(j-1)}(t,u) \ ,$$

where the second term is 0 when $k = 0$. First note that each of the above terms is a polynomial in u. Thus, let $m \in [0, n^{(j)}]$, we now extract the coefficient of u^m from each of the two terms above. Considering the first term, we have

$$\begin{aligned}[u^m]\left(A_j(h;u)[t^k]P_h^{(j-1)}(t,u)\right) &= \sum_{q=0}^{h-1}\binom{n_j}{q}[t^k u^{m-q}]P_h^{(j-1)}(t,u) \\ &= \sum_{q=0}^{\min(n_j,h-1,m)}\binom{n_j}{q}P_h^{(j-1)}[k,m-q] \ ,\end{aligned}$$

the first equality follows by the definition of $A_j(h;u)$ and the Cauchy product. The last equality uses the inductive hypothesis and ignores zero terms, that is: (a) terms with $q > m$, since $m - q < 0$ and the polynomial $P_h^{(j-1)}(t,u)$ has only monomials of non-negative degree in u; and (b) terms with $q > n_j$, since $\binom{n_j}{q} = 0$. For the second term, since $B_j(h;u) = \binom{n_j}{h}u^h$, the multiplication by u^h increases by exactly h the degree in u of each monomial of $P_h^{(j-1)}(t,u)$. Therefore, the extraction of $[u^m]$ corresponds to extracting $[u^{m-h}]$ from $P_h^{(j-1)}(t,u)$, hence

$$\begin{aligned}[u^m]\left(B_j(h;u)[t^{k-1}]P_h^{(j-1)}(t,u)\right) &= \binom{n_j}{h}[t^{k-1}u^{m-h}]P_h^{(j-1)}(t,u) \\ &= \binom{n_j}{h}P_h^{(j-1)}[k-1,m-h] \ ,\end{aligned}$$

using the inductive hypothesis. Note that the above term equals 0 for $m < h$, and for $n_j < h$ since $\binom{n_j}{h} = 0$. Summing the two terms yields the right-hand side of Eq. (34), that is

$$[t^k u^m]P_h^{(j)}(t,u) = P_h^{(j)}[k,m] \ ,$$

concluding the induction, and yielding the claim. □

Note that by the above result we have $P_h^{(\ell)}[k,m] = |\mathcal{S}(k,h,m)|$, since $P_h^{(\ell)}(t,u) = P_h(t,u)$ and by Lemma C.5.

Algorithm 6 implements Eq. (34) in place, i.e., using a single matrix P_h, initialized to $P_h^{(1)}$ by Algorithm 5. The term for $q = 0$ in Eq. (34) equals $P_h^{(j-1)}[k,m]$, i.e., the current value of the entry, which is maintained in place. By Eq. (34), the value of an entry $[k,m]$ is obtained from the entries $[k,m-q]$ for $q \in [1, \min(n_j, h-1)]$ and $[k-1, m-h]$, i.e., from entries in the same row with a smaller column, or in a smaller row. Since the algorithm processes both rows and columns in decreasing order, these entries are processed after $[k,m]$, and thus each entry (when processed) holds its value for $P_h^{(j-1)}$ when read, preserving the correctness of our result. Note that by reversing the recursion (e.g., see [42]) we can remove any class in-place, as we do in Algorithm 7. We update the Shapley values ϕ_i to be reported in output in Line 31, and finally output the average values over $\mathcal{D}$ (Line 32).

Algorithm 3: ENPOWER$_{\mathsf{MC}}$-β: exact computation of Banzhaf values for multiclass classification

Input: dataset $\mathcal{D}$, ensemble $\mathcal{C} = \{C_1, \ldots, C_n\}$, labels $\mathcal{Y}$.

$\beta_i \leftarrow 0$ for $i \in [n]$;
for $(x, y) \in \mathcal{D}$ **do**
  $n_1 \leftarrow \sum_{C \in \mathcal{C}} \mathbb{1}[C = y]$;
  $D[y] \leftarrow 1$; $j \leftarrow 2$;
  for $r \leftarrow 1$ **to** ℓ **do**
    if $y_r \neq y$ **then**
      $n_j \leftarrow \sum_{C \in \mathcal{C}} \mathbb{1}[C = y_r]$; $D[y_r] \leftarrow j$; $j \leftarrow j + 1$;
  for $j \leftarrow 1$ **to** ℓ **do** $\beta'_j \leftarrow 0$;
  $\beta'_1 \leftarrow 1$;
  for $h \leftarrow 1$ **to** n_1 **do**
    for $j \leftarrow 2$ **to** ℓ **do** $A_j(h) \leftarrow \sum_{q=0}^{h-1} \binom{n_j}{q}$, $B_j(h) \leftarrow \binom{n_j}{h}$;
    $P_h(t) \leftarrow \prod_{j=2}^{\ell} (A_j(h) + B_j(h)t)$;
    $\beta'_1 \leftarrow \beta'_1 + \sum_{k=1}^{\ell-1} \frac{1}{k+1} \left[k\binom{n_1-1}{h} + \binom{n_1-1}{h-1} \right] [t^k] P_h(t)$;
    for $j \leftarrow 2$ **to** ℓ **do**
      $\beta'_j \leftarrow \beta'_j - \sum_{k=0}^{\ell-2} \frac{1}{k+2} \binom{n_1}{h} \left[\binom{n_j-1}{h} + \frac{1}{k+1} \binom{n_j-1}{h-1} \right] [t^k] \frac{P_h(t)}{A_j(h) + B_j(h)t}$;
  for $i \leftarrow 1$ **to** n **do**
    $y_C \leftarrow C_i(x)$;
    $j \leftarrow D[y_C]$;
    $\beta_i \leftarrow \beta_i + \beta'_j / 2^{n-1}$;
return $\{\beta_i / |\mathcal{D}|\}_{i \in [n]}$;

That is upon the computation of the coefficients of the polynomial $P_h(t, u)$ our Algorithm 4 simply implements the result from Theorem 4.8.

Note that in the loop computing the Shapley values for the incorrect classifiers (Line 21), we modify in place the matrix $P_h[k, m]$, for a space efficient implementation, by safely removing classes as mentioned above using Algorithm 7.

We discuss in App. E how to implement our methods without incurring in numerical issues due to the presence of large binomial and factorial terms.

Monte-Carlo sampling baseline. We present our baseline `MC` in Algorithm 8. The algorithm takes in input a sample size s, the dataset $\mathcal{D}$ and the ensemble $\mathcal{C}$. The algorithm process each model $i \in [n]$ independently by sampling s coalitions S such that $\mathbb{P}[S] = 1/2^{n-1}$ for the Banzhaf value and $\mathbb{P}[S] = \frac{(n-|S|)!(|S|-1)!}{n!}$ for the Shapley index. That is, such probability accounts for the weight of each marginal gain $v(S + i) - v(S)$ in the respective power index, that is, the estimate is unbiased. The output averages across samples, and over the entire dataset $\mathcal{D}$ the marginal contribution of $v(S + i) - v(S)$ for all the s samples over each point. Since evaluating $v(S)$ on a point requires computing all the votes of the sampled coalition with size bounded by n, the final complexity is $\mathcal{O}(sn^2|\mathcal{D}|)$. We remark that more advanced and sample efficient techniques exists, e.g., [31]. However, our baseline serves to show that a simple sampling algorithm is significantly more inaccurate and inefficient compared to our algorithms in ENPOWER.

E Efficient and numerically stable implementation

In this section we briefly details about our practical implementation of ENPOWER. We highlight how we addressed specific issues regarding numerical stability and efficiency.

Banzhaf power index (ENPOWER$_{\mathsf{B}}$). For simplicity fix $i \in [n]$ to be a correct classifier, an analogous discussion applies to incorrect classifiers. From Theorem 4.1 we can express the Banzhaf index as $\binom{n}{n_-}/2^n + (\frac{1}{2}^n)$. Therefore, computing the Banzhaf index evaluating each individual term will yield to numerical issues, given by the large values attained by 2^n and by $\binom{n}{n_-}$. However, we observe that $\binom{n}{n_-}/2^n$ corresponds to the probability that a Binomial random variable $\mathrm{Bin}(n, \frac{1}{2})$ equals n_-.

Algorithm 4: ENPOWER$_{\mathsf{MC}}$-ϕ: exact computation of Shapley values for multiclass classification

Input: dataset $\mathcal{D}$, ensemble $\mathcal{C} = \{C_1, \ldots, C_n\}$, labels $\mathcal{Y}$.

$\phi_i \leftarrow 0$ for $i \in [n]$
$J_{\text{unique}} \leftarrow$ empty dictionary
for $(x, y) \in \mathcal{D}$ **do**
  $n_1 \leftarrow \sum_{C \in \mathcal{C}} \mathbb{1}[C = y]$
  $D[y] \leftarrow 1$; $j \leftarrow 2$
  $\boldsymbol{p} \leftarrow (n_1, 0, \ldots, 0) \in \mathbb{R}^{\ell}$
  for $r \leftarrow 1$ **to** ℓ **do**
    if $y_r \neq y$ **then**
      $n_j \leftarrow \sum_{C \in \mathcal{C}} \mathbb{1}[C = y_r]$; $p_j \leftarrow n_j$; $D[y_r] \leftarrow j$; $j \leftarrow j + 1$
  if $\boldsymbol{p} \in J_{\textit{unique}}$ **then**
    $(\phi'_1, \ldots, \phi'_{\ell}) \leftarrow J_{\text{unique}}[\boldsymbol{p}]$
  else
    for $j \leftarrow 1$ **to** ℓ **do** $\phi'_j \leftarrow 0$
    $\phi'_1 \leftarrow 1/n$
    for $h \leftarrow 1$ **to** n_1 **do**
      $P_h \leftarrow$`ComputeCoeffsDP`$(n_j$ for $j \in [2, \ell], h)$
      for $k \leftarrow 1$ **to** $\ell - 1$ **do**
        for $m \leftarrow kh$ **to** $n - n_1$ **do**
          $\phi'_1 \leftarrow \phi'_1 + f_1(k, h, m) P_h[k, m]$
      for $n_j \in J_{\text{unique}}$.keys() **do**
        $P_h \leftarrow$ `RemoveClass`(P_h, n_j, h)
        $\phi_{\text{curr}} \leftarrow 0$
        for $k \leftarrow 0$ **to** $\ell - 2$ **do**
          for $m \leftarrow kh$ **to** $n - (n_1 + n_j)$ **do**
            $\phi_{\text{curr}} \leftarrow \phi_{\text{curr}} + f_j(k, h, m) P_h[k, m]$
        $P_h \leftarrow$ `AddClass`$(P_h, n_j, h, \ell - 2, \sum_{j' \geq 2} n_{j'} - n_j)$
      $J_{\text{unique}}[\boldsymbol{p}] \leftarrow (\phi'_1, \ldots, \phi'_{\ell})$
  for $i \leftarrow 1$ **to** n **do**
    $y_C \leftarrow C_i(x)$
    $j \leftarrow D[y_C]$
    $\phi_i \leftarrow \phi_i + \phi'_j$
return $\{\phi_i / |\mathcal{D}|\}_{i \in [n]}$

Algorithm 5: `ComputeCoeffsDP`$(n_j$ for $j \in [2, \ell], h)$

Input: Number of classifiers n_j for each incorrect class $j \in [2, \ell]$, threshold h

$n' \leftarrow 0$; $P_h \leftarrow \mathbf{0} \in \mathbb{R}^{\ell \times (\sum_{j \geq 2} n_j + 1)}$;
$P_h[0, 0] \leftarrow 1$;
for $j \leftarrow 2$ **to** ℓ **do**
  $P_h \leftarrow$ `AddClass`$(P_h, n_j, h, j - 2, n')$;
  $n' \leftarrow n' + n_j$;
return P_h;

Therefore, we employ such simple results to directly compute the ratio $\binom{n}{n_-}/2^n$, and we account for the missing term in β_i using a numerically stable implementation to evaluate the value $\frac{1}{2}^n$.

Shapley power index (ENPOWER$_{\mathsf{B}}$). The computation of Eq. (5) by its definition has several drawbacks. First it is not numerically stable, due to the large values of the terms involved in the summation, leading to numerical issues. Second, there are several repeated calculations, involving factorial and binomial terms. To address the above issues we evaluate $\phi_i, i \in [n]$ through a recursive

Algorithm 6: `AddClass`(P_h, n_j, h, u, m')

Input: Matrix P_h, counts n_j, threshold h, max current label ties u, max incorrect votes m'

$q_{\max} \leftarrow \min(n_j, h-1)$
for $k \leftarrow u$ **downto** 0 **do**
  for $m \leftarrow m'$ **downto** 0 **do**
    if $P_h[k,m] > 0$ **then**
      if $n_j \geq h$ **then**
        $P_h[k+1, m+h] \leftarrow P_h[k+1, m+h] + P_h[k,m] \cdot \binom{n_j}{h}$;
      for $q \leftarrow q_{\max}$ **downto** 1 **do**
        $P_h[k, m+q] \leftarrow P_h[k, m+q] + P_h[k,m] \cdot \binom{n_j}{q}$;
return P_h;

Algorithm 7: `RemoveClass`(P_h, n_j, h)

Input: Matrix P_h, classifier count n_j, threshold h

$q_{\max} \leftarrow \min(n_j, h-1)$
$n' \leftarrow \sum_{i=2}^{\ell} n_i$
for $k \leftarrow 0$ **to** $\ell - 2$ **do**
  for $m \leftarrow 0$ **to** n' **do**
    if $n_j \geq h$ **and** $k \geq 1$ **and** $m \geq h$ **then**
      $P_h[k,m] \leftarrow P_h[k,m] - P_h[k-1, m-h] \cdot \binom{n_j}{h}$;
    for $q \leftarrow 1$ **to** $q_{\max}$ **do**
      if $m \geq q$ **then**
        $P_h[k,m] \leftarrow P_h[k,m] - P_h[k, m-q] \cdot \binom{n_j}{q}$;
return P_h;

formula. That is, if we set $w_0 = (n + 2n_\sigma - 4)$, $T_0 = \frac{n_\tau}{n(n-1)(n-2)}$ and

$$w_j = w_{j-1} - 4 \quad \text{and} \quad T_h = \frac{2(n_\sigma - h)(n_\tau - h)(2h+1)}{(h+1)(n-2h-1)(n-2h-2)} T_{h-1}\,,$$

for $h \geq 1$, then we can express ϕ_i with the summation

$$\phi_i = \sum_{h \geq 0} T_h w_h\,.$$

The above expression allows us to compute the Shapley values from Eq. (5) efficiently in linear time $\mathcal{O}(n)$. That is, we use a constant number of operations to evaluate each term $T_h w_h$. Moreover, the above summation is numerically stable since it avoids evaluating factorials and binomials.

Algorithm 8: Monte-Carlo estimation of $\psi \in \{\phi, \beta\}$

Input: Dataset $\mathcal{D}$, ensemble $\mathcal{C}$, sample size s.

$\hat{\psi}_i \leftarrow 0, i \in [n]$;
for $i \leftarrow 1$ **to** n **do**
  for $k \leftarrow 1$ **to** s **do**
    $S \leftarrow \emptyset$;
    if $\psi = \beta$ **then**
      $S \leftarrow \{j \neq i : \texttt{FairCoin}_j = \texttt{H}\}$.
    else
      $\sigma \leftarrow \texttt{RandomPermutation}(n)$;
      $S \leftarrow \{j : \sigma_j < \sigma_i\}$;
    for $(x, y) \in \mathcal{D}$ **do** $\hat{\psi}_i \leftarrow \hat{\psi}_i + (v(S+i) - v(S))$;
  return $\left\{\hat{\psi}_i / (s|\mathcal{D}|)\right\}_{i \in [n]}$;

Table 1: Datasets used for binary and multiclass classification settings in Sec. 5. Classes (ℓ), denotes the original number of labels, which are also retained for the multiclass experiments. The last two columns display the number of points in $\mathcal{D}_{\text{val}} \cup \mathcal{D}_{\text{test}}$ with label $y = 1$ and $y = 0$ after binarization.

Dataset (Label)	$\lvert\mathcal{D}_{\text{val}}\rvert + \lvert\mathcal{D}_{\text{test}}\rvert$	Classes (ℓ)	$y = 1$	$y = 0$
Blood Transfusion (BTSC)	187	2	143	44
Credit-G (CR)	250	2	179	71
CNAE-9 (C9)	270	9	38	232
Titanic (TT)	328	2	210	118
Bank Marketing (BM)	11 303	2	9 994	1 309
Covertype (CT)	145 253	7	70 628	74 625

Table 2: Dataset statistics for the zero-shot multiclass classification experiment. For each dataset we display its total number of images, and the number of possible classes. We also show $|\mathcal{D}_{\text{val}}|$, used to compute the ranking metrics, and $|\mathcal{D}_{\text{test}}|$ used to evaluate the generalization of model selection. Column $|U|$ reports the number of unique vote profiles observed on $\mathcal{D}_{\text{val}}$ (the ensemble has $n = 22$ base predictors), averaged over five splits. Other columns do not depend on the split.

Dataset	Total images	Classes (ℓ)	$\lvert\mathcal{D}_{\text{val}}\rvert$	$\lvert\mathcal{D}_{\text{test}}\rvert$	$\lvert U\rvert$
CIFAR (Cifar-100)	10 000	100	5 000	5 000	675.2 ± 6.6
PETS (Oxford-IIIT Pets)	3 669	37	1 834	1 835	123.6 ± 5.4
IMAGENET-A (ImageNet-A)	7 500	200	3 750	3 750	861.0 ± 11.6

Banzhaf and Shapley power indices for the multiclass case (ENPOWER$_{\text{MC}}$). We note that a naive implementation of Algorithms 3 and 4 would incur in numerical issues (such as integer overflows) due to the computation of large binomial and factorials. To fix such issue our C++ implementation represents the coefficients of the dynamic programming matrices as exact integers. Specifically, for each vote profile we use the smallest fixed-width unsigned integer to represent the numbers used in the evaluations of Theorem 4.6 and Theorem 4.8, ranging the length from 128 up to 1024 bits, which supports ensembles with $n \leq 1023$. We then only evaluate in floating point the final weighted sum corresponding to the power indices. Note that long integers increase the cost of arithmetical operations such as multiplications, which is however necessary to enable the practical usability of our methods, i.e., retaining the accuracy of the exact computation of our methods. As a concrete example, the above discussion explains the high runtime of ENPOWER-ϕ on the CT dataset and the AdaBoost ensemble from Table 5.

F Reproducibility

F.1 Setup

Environment. The code was executed on a machine with four Intel Xeon Gold 6252N CPU @ 2.30/3.60GHz and 3TB of RAM memory. All experiments are executed in the same container based on Debian Linux, obtained through Apptainer,[6] and that we provide with our source code.[7] Our container builds the necessary environment specifying: the C++ compiler and the Python versions and packages. Each run is sequential: it uses a single thread on a dedicated CPU core. All our reported time measurements do not include the training of the ensembles or the evaluation of the predictions of base models.

All our methods use either Python or C++: (i) we use C++17, compiled with g++ and optimization flags for all algorithms of ENPOWER, i.e., ENPOWER-β and ENPOWER-ϕ for the binary and multiclass settings; and our baselines WMG and ISCG, and the sampling baseline MC. (ii) we use Python 3.10 for the baselines Tr-MLE and Tr-PS, i.e., using the code of Rozemberczki and Sarkar [36].[8] We also use Python to train all ensembles through Scikit-learn, and computing the predictions of CLIP by using PyTorch.

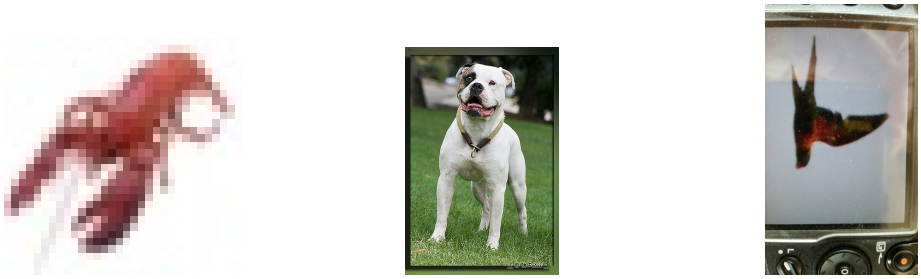
[6] https://apptainer.org/

[7] https://github.com/iliesarpe/enpower

[8] https://github.com/benedekrozemberczki/shapley

Table 3: Class-imbalance statistics for the multiclass datasets of Sec. 5.4 (computed on $\mathcal{D}_{\text{val}} \cup \mathcal{D}_{\text{test}}$, hence equal on every split). ℓ is the number of classes. "min/max" are the smallest/largest number of points (images) with the same class; $IR = \max / \min$ is the imbalance ratio; CV is the coefficient of variation computed over the cardinality of all classes; and $H_{\text{norm}} \in (0, 1]$ is the normalized Shannon entropy of the class distribution (1 corresponds to a uniform distribution).

Dataset	Classes (ℓ)	min	max	IR	CV (%)	H_{norm}
CIFAR (Cifar-100)	100	100	100	1.00	0.0	1.000
PETS (Oxford-IIIT Pets)	37	88	100	1.14	2.6	1.000
IMAGENET-A (ImageNet-A)	200	3	100	33.33	78.2	0.946



(a) CIFAR (b) PETS (c) IMAGENET-A

Figure 5: Sample images from the three datasets considered in Sec. 5.4. CIFAR consists of tiny, low resolution (32×32 pixel) images. PETS consists of high quality images of different pets with fine-grained classes, and IMAGENET-A is a notorious challenging dataset for image classification.

CLIP. We use CLIP's version ViT-B/32.[9]

Ensembles training. As mentioned in Sec. 5 for our experiments comparing with sota (state-of-the-art) approaches we use the 75% of data points of each dataset to train three different ensembles. More in detail we use the default classifiers available from Scikit-learn. In particular,

1. For both `RF` and `AB` we do not set any custom parameter other than the number of classifiers required;
2. For `HET` we train each classifier with slightly different parameters to enforce heterogeneity.

All the details of the parameter settings are easily accessed in our code repository `https://github.com/iliesarpe/enpower`.

F.2 Datasets

We describe the datasets used in our experiments of Sec. 5. The scripts to replicate the processing of our datasets is in our code repository `https://github.com/iliesarpe/enpower`.

Binary and multiclass classification. The datasets used in our experiments for the binary and multiclass classification experiments of Sec. 5 are reported in Table 1. We additionally report the distribution of the classes of each dataset in Table 1. All datasets are publicly available.[10] Each dataset was processed with a standard pipeline as follows,

- for the binary classification scenario we remapped multiple classes by retaining the most frequent class as positive ($y = 1$) and all other classes collectively as negative ($y = 0$);
- numerical features were standardized to zero mean and unit variance;
- missing values were set to the median for the respective feature;
- categorical features were mapped to integers;
- we split each dataset with a 75/15/10 split. The 75% split of data points is used for training. The 15% spit constitutes $\mathcal{D}_{\text{val}}$ used to compute the rankings of the power indices according

[9] `https://github.com/openai/CLIP`
[10] `https://archive.ics.uci.edu/`.

to the various methods. Finally, the remaining 10% split is used for testing, constituting $\mathcal{D}_{\text{test}}$. We first draw the 75% training split at random, and then split the remaining 25% of the points at random into $\mathcal{D}_{\text{val}}$ (60% of the remaining) and $\mathcal{D}_{\text{test}}$ (40% of remaining points).

LLM prompt ensembling. For our experiments of Sec. 5.4 we rely on publicly available datasets for image classification. Their statistics are reported in Table 2, and some sample images are show in Figure 5. We further report the statistics for class distribution in Table 3. We observe that our datasets are heterogeneous: CIFAR is perfectly balanced (imbalance ratio (IR) 1.0), PETS is nearly balanced (IR, 1.14), and IMAGENET-A is highly imbalanced (IR, 33.3), showing that our methods are not affected by class distribution as captured by our theoretical results. Each dataset was split retaining 50% of its points for $\mathcal{D}_{\text{val}}$ and the rest for $\mathcal{D}_{\text{test}}$.

F.3 Baselines

Baselines for binary classification. Here we provide more details for the baselines of our experiments.

`ISCG` [46] computes the Shapley values of each base classifier through a game on a graph capturing pairwise relationship between base classifier (see Sec. 2) and their diversity across predictions. Note that the game of `ISCG` does *not* capture how the actual ensemble aggregates their votes. The game is an induced subgraph game [10] on the complete graph where each vertex corresponds to a base classifier, with the addition of self-loops. An edge $\{i, j\}$ has weight w_{ij}, and the self-loop at vertex i has weight w_{ii}. The value v of a coalition S corresponds to the sum of the weights of the edges and self-loops with both endpoints in S. The Shapley value of C_i corresponds to $w_{ii} + \frac{1}{2}\sum_{j\neq i} w_{ij}$ [10]. The weights depend only on which classifiers are correct. For a point drawn uniformly at random from $\mathcal{D}_{\text{val}}$, let $Z_i = 1$ if C_i classifies the point correctly, and $Z_i = 0$ otherwise. Following Ykhlef and Bouchaffra [46, Eq. (5) to (7)], we set

$$w_{ii} = \Pr[Z_i = 1], \qquad w_{ij} = \tfrac{1}{2}\big(\Pr[Z_i = 1 \mid Z_j = 0] + \Pr[Z_j = 1 \mid Z_i = 0]\big) \quad \text{for } i \neq j.$$

Hence w_{ii} corresponds to the accuracy of C_i on $\mathcal{D}_{\text{val}}$. While the term $\Pr[Z_i = 1 \mid Z_j = 0]$ corresponds to the fraction of the points misclassified by C_j that C_i classifies correctly. When C_j performs no classifications, we set the fraction to 0. We compute the weights on $\mathcal{D}_{\text{val}}$, as for all other methods.

`Tr-MLE` and `Tr-PS` implement the algorithms of Owen [34] and Maleki et al. [27], respectively. Both methods compute the Shapley values under the definition of game introduced in Rozemberczki and Sarkar [36], and both methods are approximate. In our setting, for each point $z = (x, y)$ in $\mathcal{D}_{\text{val}}$ `Troupe` considers a simple weighted majority game, in which base classifier C_i has weight $w_i(x) = \frac{1}{n}\mathbb{1}[C_i(x) = y]$. And for a fixed coalition $S \subseteq \mathcal{C}$ it holds

$$v_{\text{Tr}}(z, S) = \mathbb{1}\Big[\sum_{i\in S} w_i(x) > q\Big], \qquad q = \tfrac{1}{2} \ .$$

That is, a coalition S attains a value of 1 for the game over z it contains at least $n/2$ base classifiers that are correct on x. Note that the quota is relative to the size n of the ensemble, not to $|S|$, and must be fixed in advance. The power-index that we report for $C_i, i \in [n]$ corresponds to average of its Shapley value over the games of the points in $\mathcal{D}_{\text{val}}$. We use the code of Rozemberczki and Sarkar [36] with the above weights and quota. `Tr-PS` samples 1 000 permutations, and we execute it five times. While, `Tr-MLE` is deterministic. We note that on a point with at most $n/2$ correct base classifiers the grand coalition has value 0, and every Shapley value of that game is 0. The code of the authors does not return 0 on such a point, and we use their code unaltered.[11]

`WMG` implements the algorithms of [42] for computing the Banzhaf and Shapley values under WMGs, for which we set the weight of a classifier on a point to 1 if it predicts the label of the point correctly and to 0 otherwise, and the quota to $q = \lfloor n/2 \rfloor + 1$. Each point of $\mathcal{D}_{\text{val}}$ thus defines one WMG. For each model C_i, its power index Banzhaf (resp. Shapley) value that we report under `WMG` is obtained as the average over the points of $\mathcal{D}_{\text{val}}$. Note that in a WMG for a fixed point, the power index depends only on the number of correct base classifiers, i.e., n_+. We therefore run the dynamic program of Uno [42] once per distinct value of n_+, and reuse its output for all points with the same values of n_+.

[11] `https://github.com/benedekrozemberczki/shapley`

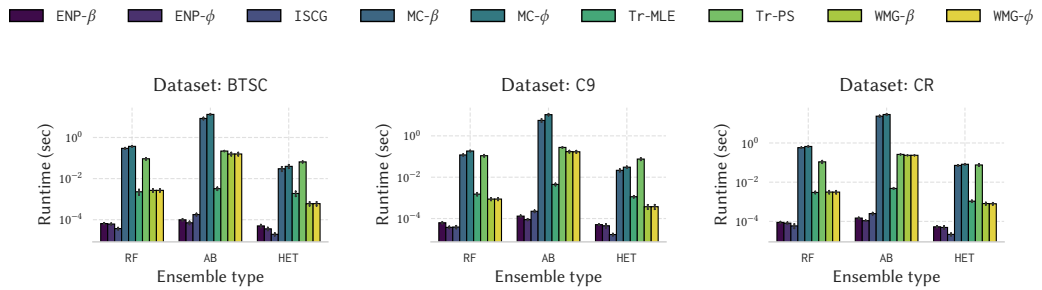


Figure 6: Runtime comparison (mean and standard deviation) on the missing datasets from Table 1 not reported in Sec. 5 due to space constraints.

We have also implemented two simple Monte-Carlo algorithms for our game, denoted as MC-ϕ and MC-β. Given that these algorithms require a high runtime to converge, and have poor estimation quality (see Table 4) we did not consider such baselines in our discussion. On every configuration, we draw n^2 samples for each base classifier. We remark that obtaining efficient sampling algorithms for our formulation remains an interesting and challenging open direction, especially for the multiclass case.

Baselines for multiclass classification. All our baselines based on WMGs do not directly extend to the multiclass setting (see App. A). Possible ways to adapt such baselines for a fixed point $z = (x, y)$ are to average the power indices obtained over (i) one binary game for each pair of labels, i.e., $\mathcal{O}(\ell^2)$ games for each point; or (ii) one binary game between y and each of the $\ell - 1$ incorrect labels, i.e., $\mathcal{O}(\ell)$ games for each point. The only method admitting a multiclass formulation, by Ykhlef and Bouchaffra [46], has no public implementation, and it already correlates poorly with our indices in the binary case in our implementation (specifically on RF and HET, see Sec. 5.2). We therefore consider Tr-MLE and Tr-PS as our multiclass baselines, with a simpler reduction than (i) and (ii): a single binary game for each point, between y and all other incorrect labels. The game v_{Tr} of Troupe is a simple weighted game where a coalition (i.e., a sub-ensemble) either wins or loses. Hence, such formulation cannot capture which of the $\ell - 1$ incorrect classes an incorrect model votes for. For each point $z = (x, y)$ of $\mathcal{D}_{\text{val}}$ we evaluate Troupe as follows: we retain only the correctness of the predictions of model C_i on the data point x, i.e., $\mathbb{1}[C_i(x) = y]$; and we assign to each base classifier a weight $w_i(x) = \frac{1}{n}\mathbb{1}[C_i(x) = y]$, as for the binary case. That is, a coalition S wins, i.e., $v_{\mathsf{Tr}}(z, S) = 1$, if it contains more than $n/2$ correct base classifiers (quota $q = \frac{1}{2}$). We then execute Tr-MLE and Tr-PS on such game with the parameters of the binary case, and average the values of each $C_i, i \in [n]$ over the points of $\mathcal{D}_{\text{val}}$. The above game v_{Tr} differs substantially from v_{MC}: (i) each incorrect base classifier has weight 0, i.e., models voting for different incorrect labels cannot be distinguished; (ii) v_{Tr} requires more than $n/2$ correct classifiers to evaluate to 1, while v_{MC} does not require a fixed quota; and (iii) v_{Tr} does not account for ties (see Eq. (3)), as it requires a strict majority. For example, let $n = 22$, and consider a point z with 8 correct votes and 2 votes on each of the remaining 7 incorrect labels. According to the plurality voting rule, the ensemble classifies x correctly, thus $v_{\mathsf{MC}}(z, \mathcal{C}) = 1$, while $v_{\mathsf{Tr}}(z, \mathcal{C}) = 0$, and every Shapley value of v_{Tr} for the game over z is 0, since $v_{\mathsf{Tr}}(z, S) = 0$ for every $S \subseteq \mathcal{C}$. Data points with less than $n/2$ base correct classifiers are common in our experiments, e.g., on IMAGENET-A (see Table 11).

G Additional experimental results

G.1 Monte-Carlo approximation

We show the estimation quality achieved by our simple sampling baseline from Algorithm 8 in Table 4. We computed, for each model the absolute error $\varepsilon_{\text{abs}} = |\psi_i - \hat{\psi}_i|$ and the relative error $\varepsilon_{\text{rel}} = |\psi_i - \hat{\psi}_i|/|\psi_i|$ where $\hat{\psi}_i$ corresponds to the estimate provided by MC-ψ for $\psi \in \{\beta, \phi\}$. Finally we also report the Kendall's τ and Spearman's correlation metrics. We recall that the results refer to the binary classification in Sec. 4.1, where the sample size is $s = n^2$; note that the number of possible samples is exponential in n.

Table 4: Approximation error of the MC baseline. For each dataset and power index we report the maximum absolute (ε_{abs}) and the maximum relative approximation error (ε_{rel}), over all models from $\mathcal{C}$. We further report the Kendall's (τ) and Spearman (ρ) ranking coefficients, with the exact scores computed by ENPOWER. The results are the average and standard deviation over five random splits.

Dataset	Ensemble	Index	ε_{abs}	ε_{rel}	Kendall's (τ)	Spearman (ρ)
BTSC	RF	ϕ	0.0045 ± 0.0007	0.42 ± 0.15	0.720 ± 0.036	0.889 ± 0.027
		β	0.0003 ± 0.0000	3.34 ± 4.51	0.967 ± 0.004	0.997 ± 0.001
	AB	ϕ	0.0013 ± 0.0002	0.26 ± 0.18	0.819 ± 0.030	0.930 ± 0.017
		β	0.0007 ± 0.0001	0.07 ± 0.02	0.794 ± 0.036	0.917 ± 0.023
	HET	ϕ	0.0111 ± 0.0021	0.51 ± 0.10	0.515 ± 0.094	0.676 ± 0.105
		β	0.0005 ± 0.0002	13.72 ± 25.07	0.886 ± 0.073	0.964 ± 0.034
CR	RF	ϕ	0.0043 ± 0.0005	0.38 ± 0.06	0.780 ± 0.026	0.930 ± 0.015
		β	0.0004 ± 0.0001	1.61 ± 0.96	0.968 ± 0.005	0.997 ± 0.001
	AB	ϕ	0.0010 ± 0.0002	0.28 ± 0.20	0.882 ± 0.014	0.964 ± 0.006
		β	0.0004 ± 0.0001	0.07 ± 0.08	0.863 ± 0.021	0.953 ± 0.010
	HET	ϕ	0.0106 ± 0.0022	0.46 ± 0.14	0.558 ± 0.170	0.720 ± 0.194
		β	0.0007 ± 0.0001	0.96 ± 0.87	0.931 ± 0.037	0.986 ± 0.010
C9	RF	ϕ	0.0063 ± 0.0009	0.31 ± 0.04	0.288 ± 0.089	0.406 ± 0.112
		β	0.0001 ± 0.0000	0.60 ± 0.35	0.791 ± 0.043	0.938 ± 0.021
	AB	ϕ	0.0023 ± 0.0002	0.10 ± 0.05	0.800 ± 0.017	0.929 ± 0.008
		β	0.0015 ± 0.0003	0.06 ± 0.03	0.764 ± 0.041	0.910 ± 0.022
	HET	ϕ	0.0148 ± 0.0029	0.42 ± 0.09	0.457 ± 0.139	0.609 ± 0.152
		β	0.0003 ± 0.0001	3.42 ± 2.97	0.816 ± 0.065	0.935 ± 0.032
TT	RF	ϕ	0.0056 ± 0.0009	0.30 ± 0.03	0.444 ± 0.070	0.620 ± 0.082
		β	0.0001 ± 0.0000	5.38 ± 3.58	0.963 ± 0.008	0.996 ± 0.001
	AB	ϕ	0.0010 ± 0.0000	0.12 ± 0.06	0.936 ± 0.015	0.989 ± 0.005
		β	0.0006 ± 0.0001	0.06 ± 0.02	0.891 ± 0.020	0.970 ± 0.008
	HET	ϕ	0.0136 ± 0.0028	0.51 ± 0.08	0.706 ± 0.047	0.872 ± 0.032
		β	0.0006 ± 0.0001	0.86 ± 0.63	0.913 ± 0.022	0.980 ± 0.007
BM	RF	ϕ	0.0056 ± 0.0007	0.31 ± 0.04	0.081 ± 0.075	0.126 ± 0.117
		β	0.0000 ± 0.0000	3.14 ± 2.79	0.971 ± 0.006	0.997 ± 0.001
	AB	ϕ	0.0015 ± 0.0002	0.90 ± 0.47	0.929 ± 0.020	0.984 ± 0.009
		β	0.0004 ± 0.0002	0.46 ± 0.29	0.941 ± 0.029	0.987 ± 0.012
	HET	ϕ	0.0134 ± 0.0026	0.43 ± 0.07	0.187 ± 0.107	0.259 ± 0.132
		β	0.0001 ± 0.0000	0.51 ± 0.47	0.970 ± 0.018	0.996 ± 0.003
CT	RF	ϕ	0.0056 ± 0.0007	0.29 ± 0.04	0.053 ± 0.072	0.077 ± 0.097
		β	0.0000 ± 0.0000	0.53 ± 0.54	0.970 ± 0.008	0.997 ± 0.001
	AB	ϕ	0.0010 ± 0.0002	1.68 ± 2.00	0.843 ± 0.020	0.945 ± 0.011
		β	0.0004 ± 0.0001	6.28 ± 4.13	0.825 ± 0.037	0.929 ± 0.031
	HET	ϕ	0.0119 ± 0.0024	0.40 ± 0.05	0.666 ± 0.036	0.848 ± 0.026
		β	0.0002 ± 0.0001	0.28 ± 0.20	0.980 ± 0.005	0.998 ± 0.000

We observe from Table 4, that overall the performances of the sampling approach are quite remarkable in recovering most of the ranking metrics. However, we also note that on some configurations (e.g., BM with RF and HET, and CT with RF) the ranking metrics with the exact Shapley values are below 0.3. We also observe a non-negligible relative error in some configurations (e.g., BTSC with HET), likely due to the small values of the values of ϕ_i and β_i for $i \in [n]$. This connects to a well known phenomena for sampling algorithms, which require a very high sample size to provide high quality approximations of extremely small values [32].

We remark that we leave as an open direction designing efficient sampling algorithms, especially for the multiclass case, e.g., by reusing samples across different models.

G.2 Runtime of ENPOWER$_{\text{MC}}$

Table 5 reports the runtime of our methods on the two datasets of Table 1 with $\ell \geq 3$ labels, i.e., datasetsC9 and CT. ENPOWER-β requires at most 2 seconds in all configurations, including CT, our largest dataset. ENPOWER-ϕ requires at most 8 seconds on C9, and at most 3 seconds on CT with RF and HET. The only exception corresponds to CT with AB, our largest ensemble ($n = 150$), where ENPOWER-ϕ requires approximately 4 minutes, due to the $\mathcal{O}(n^3)$ term in its complexity and to the exact integer arithmetic (see App. E).

Table 5: Runtime (seconds) of ENPOWER on $\mathcal{D}_{\text{val}}$ in the multiclass setting. The results are the average and standard deviation over five random splits.

Ensemble	Method	C9	CT
RF ($n = 50$)	ENPOWER-β	$(3.7 \pm 0.4) \cdot 10^{-3}$	0.10 ± 0.01
	ENPOWER-ϕ	0.14 ± 0.02	2.6 ± 0.2
AB ($n = 150$)	ENPOWER-β	$(5.0 \pm 0.9) \cdot 10^{-2}$	1.9 ± 0.4
	ENPOWER-ϕ	7.8 ± 1.4	265 ± 50
HET ($n = 30$)	ENPOWER-β	$(1.5 \pm 0.2) \cdot 10^{-3}$	$(4.0 \pm 0.1) \cdot 10^{-2}$
	ENPOWER-ϕ	$(3.3 \pm 0.4) \cdot 10^{-2}$	0.42 ± 0.01

G.3 Accuracy of the base classifiers

Table 6 reports the mean and the standard deviation of the accuracy of the base classifiers on $\mathcal{D}_{\text{val}}$, for each combination of dataset and ensemble.

Table 6: Accuracy on $\mathcal{D}_{\text{val}}$ of the base classifiers of each ensemble, binary classification. Each entry is the mean $\pm$ the standard deviation over the base classifiers of the ensemble, both averaged over five random splits.

Ensemble	BTSC	CR	C9	TT	BM	CT
RF	0.697 ± 0.033	0.648 ± 0.035	0.979 ± 0.014	0.884 ± 0.032	0.868 ± 0.003	0.886 ± 0.005
AB	0.510 ± 0.251	0.505 ± 0.176	0.534 ± 0.376	0.535 ± 0.156	0.545 ± 0.311	0.519 ± 0.025
HET	0.745 ± 0.032	0.700 ± 0.036	0.940 ± 0.062	0.834 ± 0.121	0.895 ± 0.008	0.768 ± 0.068

G.4 Ensemble sub-selection

We present additional experimental results for the sub-ensemble selection task in Figure 7. To complement our discussion, we observe that a selection of the top-k models with the rankings provided by ENPOWER yields a sub-ensemble consistently ranking among the ones with best generalization for the accuracy and F_1-score metrics. There are, however few exceptions. On CT with AB, ACC attains higher accuracy and F_1-score for intermediate values of k. On CR with HET, WMG-ϕ and WMG-β attain higher metrics for small values of k. On both configurations the base classifiers have similar accuracy (see Table 6), thus the rankings of the methods depend on how each formulation weights the classifiers on points with tied votes. The indices of ENPOWER capture the contribution of each base classifier to the full ensemble with random tie-breaking, and such contributions may differ on a sub-ensemble of size $k < n$. We believe that such behavior may also be related to other specific properties of the datasets and of the base models. Connecting properties of the input dataset with the use of the indices of ENPOWER for sub-ensemble selection is an interesting and important direction for our work. We note that when performing sub-ensemble selection, a different approach is to recompute the indices of the remaining players after selecting a model, i.e., see [9].

G.5 CLIP prompt ensembling

The missing results of Sec. 5.4 on the PETS dataset are reported in Figure 8. Table 7 reports, for each ranking, the best accuracy and F_1-score over k, the value of k attaining it, and the average of each metric over $k \in [n]$. The best $k \in [n]$ corresponds to the value that maximizes the mean metric over the five $\mathcal{D}_{\text{test}}$ (i.e., chosen independently for the accuracy and F_1-score). Each entry of the table corresponds to the mean and its the standard deviation over the splits. As discussed in our main Sec. 5.4, a selection according to the power indices yields metrics comparable to ACC and higher than Tr-MLE and Tr-PS.

Table 8 reports the runtime of all methods on $\mathcal{D}_{\text{val}}$. ENPOWER-β terminates in less than 0.02 seconds and ENPOWER-ϕ in less than one second on all three datasets, while Tr-PS requires up to 1.3 seconds.

In addition we also show the variation in the rankings produced by accuracy and our methods on a fixed split in Tables 9 to 11. An analysis of such tables provides some interesting results about the

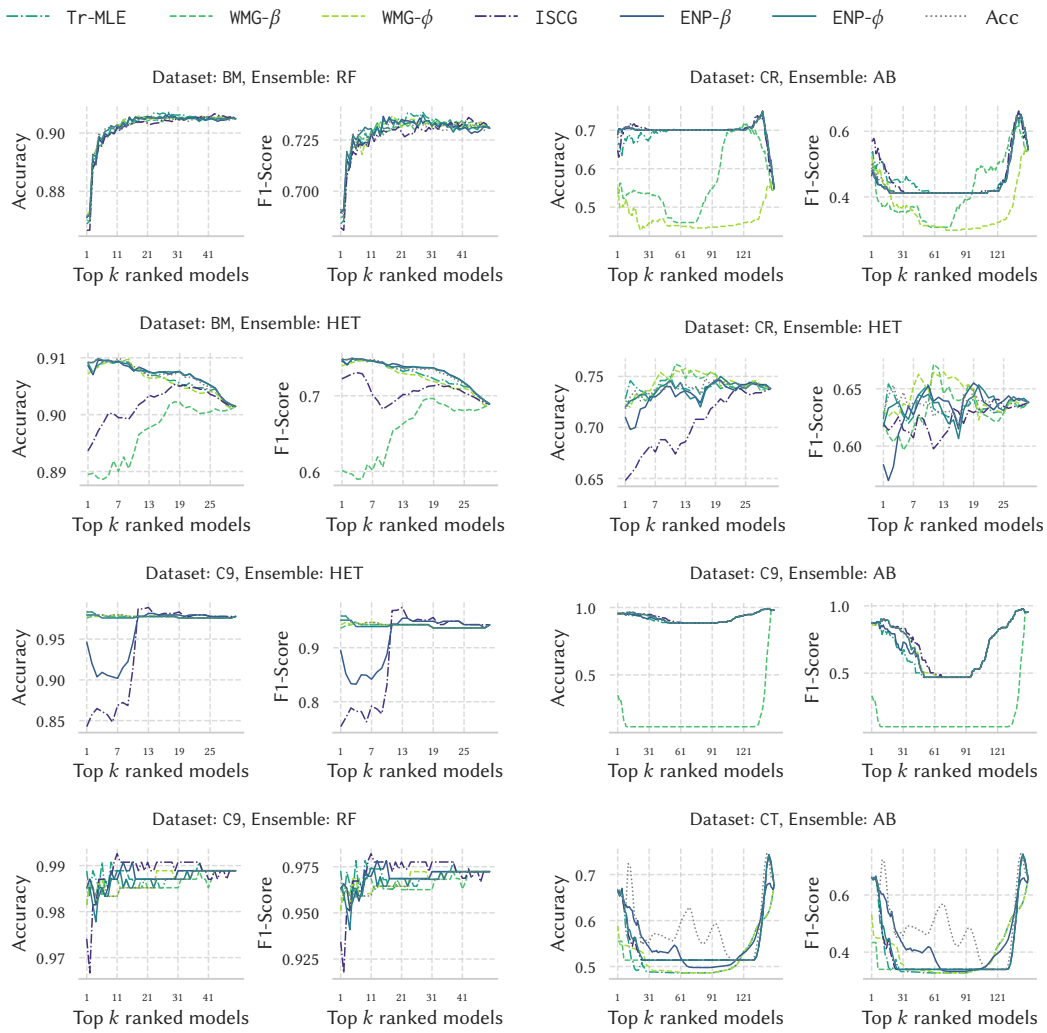


Figure 7: Generalization metrics for sub-ensemble selection, missing results from Sec. 5.3.

interaction between different prompts. We note that, for example, on dataset CIFAR the prompt "A cropped photo of a {}.", which is highly ranked by accuracy, has a poor ranking according to power indices. That is, such prompt is affecting the overall performances of the ensemble, as it can be seen by its negative Banzhaf value.

Other interesting trends can be observed over Tables 9 to 11 for example

- the rankings of prompt "A macro shot of a {}" in Table 10. That is such prompt that has poor accuracy ranking but high synergy with the overall ensemble as captured by power indices.
- the rankings of prompt "A photo of a {}, with a watermark." on the IMAGENET-A dataset in Table 11. That prompt is ranked high by accuracy, but is has significantly smaller ranks according to power indices.

Overall our experiments for prompt ensembling show that ENPOWER for multiclass classification can provide a valuable addition for LLM prompt engineering, and to enhance the interpretability of LLM predictions under different prompts.

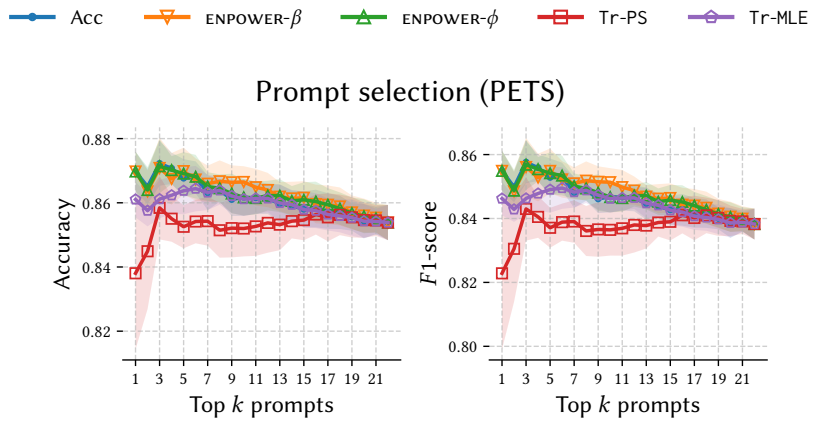


Figure 8: Accuracy and F_1-score test metrics for the prompt ensembling of CLIP on $\mathcal{D}_{\text{test}}$ (see Sec. 5.4), as a function of k. Missing results from Sec. 5.4.

Table 7: Prompt ensembling of CLIP, metrics on $\mathcal{D}_{\text{test}}$ of the sub-ensemble for varying k (see Sec. 5.4). Columns report, best: the maximum over k of the average over the five random splits, and the k attaining it. Avg.: the average over $k \in [n]$. Bold marks the best value for each configuration.

Dataset	Ranking	best acc.	k	avg. acc.	best F_1	k	avg. F_1
CIFAR100	Acc	0.648 ± 0.005	14	0.645 ± 0.004	$\mathbf{0.641 \pm 0.004}$	16	0.637 ± 0.003
	ENPOWER-β	$\mathbf{0.649 \pm 0.004}$	10	$\mathbf{0.646 \pm 0.004}$	$\mathbf{0.641 \pm 0.003}$	16	$\mathbf{0.638 \pm 0.003}$
	ENPOWER-ϕ	$\mathbf{0.649 \pm 0.004}$	8	$\mathbf{0.646 \pm 0.004}$	$\mathbf{0.641 \pm 0.004}$	17	$\mathbf{0.638 \pm 0.003}$
	Tr-PS	0.647 ± 0.005	15	0.643 ± 0.006	0.639 ± 0.004	15	0.635 ± 0.004
	Tr-MLE	0.648 ± 0.004	14	0.643 ± 0.005	0.640 ± 0.003	14	0.636 ± 0.003
IMAGENET-A	Acc	$\mathbf{0.308 \pm 0.004}$	4	0.302 ± 0.004	$\mathbf{0.288 \pm 0.005}$	1	$\mathbf{0.279 \pm 0.003}$
	ENPOWER-β	0.307 ± 0.003	5	$\mathbf{0.303 \pm 0.004}$	0.285 ± 0.004	5	$\mathbf{0.279 \pm 0.003}$
	ENPOWER-ϕ	$\mathbf{0.308 \pm 0.003}$	5	0.302 ± 0.004	$\mathbf{0.288 \pm 0.005}$	1	$\mathbf{0.279 \pm 0.003}$
	Tr-PS	0.298 ± 0.006	13	0.295 ± 0.004	0.275 ± 0.003	10	0.272 ± 0.002
	Tr-MLE	0.303 ± 0.004	8	0.300 ± 0.004	0.279 ± 0.004	8	0.275 ± 0.003
PETS	Acc	$\mathbf{0.872 \pm 0.007}$	3	0.862 ± 0.005	$\mathbf{0.858 \pm 0.007}$	3	0.846 ± 0.004
	ENPOWER-β	0.871 ± 0.009	3	$\mathbf{0.863 \pm 0.006}$	0.856 ± 0.009	3	$\mathbf{0.848 \pm 0.005}$
	ENPOWER-ϕ	0.871 ± 0.008	3	0.862 ± 0.005	0.857 ± 0.008	3	0.847 ± 0.005
	Tr-PS	0.858 ± 0.010	3	0.853 ± 0.007	0.843 ± 0.010	3	0.838 ± 0.006
	Tr-MLE	0.864 ± 0.007	6	0.859 ± 0.005	0.850 ± 0.007	6	0.844 ± 0.004

G.6 Synthetic experiments

Real data does not allow us to control the parameters that determine the runtime of ENPOWER. We therefore generate synthetic instances in which we vary the number $|U|$ of unique vote profiles, the ensemble size n and the number of classes ℓ. Recall that the runtime of our algorithms depends on $|U|$ and not on $|\mathcal{D}|$.

Data generation. A vote profile is determined by two integers: the number n_1 of classifiers predicting the correct class, and the number r of incorrect labels receiving at least one vote. We allocate the $n - n_1$ incorrect votes evenly over r labels, i.e., $(n - n_1) \bmod r$ labels receive $\lceil (n - n_1)/r \rceil$ votes and the other ones receive $\lfloor (n - n_1)/r \rfloor$ votes. There are $\ell - 1$ incorrect labels and $n - n_1$ incorrect votes, thus $r \leq r_{\max}(n_1) \doteq \min\{\ell - 1, n - n_1\}$. We consider $1 \leq r \leq r_{\max}(n_1)$ if $n_1 < n$, and $r = 0$ if $n_1 = n$. Each distinct pair (n_1, r) yields a distinct profile. Given $p \in (0, 1)$, we build an instance of $|U|$ profiles as follows.

1. Let $n_1^\star \doteq \lfloor pn + 1/2 \rfloor$, i.e., pn rounded to the nearest integer.
2. Order the values $n_1 \in \{0, \ldots, n\}$ as the sequence $n_1^\star,\ n_1^\star - 1,\ n_1^\star + 1,\ n_1^\star - 2,\ n_1^\star + 2, \ldots$, i.e., in increasing order of $|n_1 - n_1^\star|$ and, breaking ties in favor of the smallest value.
3. Build a list $\mathcal{U}$ of profiles by replacing each n_1 in this sequence with the profiles (n_1, r) in decreasing order of r, i.e., $(n_1, r_{\max}(n_1)), (n_1, r_{\max}(n_1) - 1), \ldots, (n_1, 1)$. The value $n_1 = n$ is replaced with the single profile $(n, 0)$.

Table 8: Runtime (seconds) of the ranking methods on $\mathcal{D}_{\text{val}}$ for the prompt ensembling of CLIP, average $\pm$ standard deviation over five random splits.

Method	CIFAR100	IMAGENET-A	PETS
ENPOWER-β	$(1.17 \pm 0.01) \cdot 10^{-2}$	$(1.37 \pm 0.04) \cdot 10^{-2}$	$(1.64 \pm 0.09) \cdot 10^{-3}$
ENPOWER-ϕ	$(3.61 \pm 0.07) \cdot 10^{-1}$	$(5.14 \pm 0.18) \cdot 10^{-1}$	$(3.45 \pm 0.20) \cdot 10^{-2}$
`Tr-PS`	$(1.33 \pm 0.13) \cdot 10^{0}$	$(8.64 \pm 0.44) \cdot 10^{-1}$	$(3.77 \pm 0.04) \cdot 10^{-1}$
`Tr-MLE`	$(2.37 \pm 0.05) \cdot 10^{-2}$	$(1.84 \pm 0.08) \cdot 10^{-2}$	$(1.26 \pm 0.02) \cdot 10^{-2}$

Table 9: Prompt ensemble evaluation for LLM zero-shot multiclass image classification (CIFAR100). Each prompt corresponds to a different model of $\mathcal{C}$. All values and rankings are computed on the validation split $\mathcal{D}_{\text{val}}$. Columns Δ_β and Δ_ϕ display the difference in the validation rankings induced by the power indices $\beta(G_{\mathsf{MC}})$ and $\phi(G_{\mathsf{MC}})$ with respect to validation accuracy.

Prompt template	**Accuracy**		**Banzhaf**		**Shapley**		Δ_β	Δ_ϕ
	Value	Rank	Value	Rank	Value	Rank		
A pixelated photo of a {}.	0.6338	**1**	0.00296	**1**	0.03285	**1**	–	–
A heavily compressed JPEG photo of a {}.	0.6308	**2**	0.00262	**2**	0.03226	**2**	–	–
A high-resolution, crisp photo of a {}.	0.6248	**3**	0.00113	5	0.03047	4	↓ **2**	↓ **1**
A cropped photo of a {}.	0.6242	4	-0.00003	15	0.02941	10	↓ **11**	↓ **6**
A photo of a {}, cluttered background.	0.6238	5	0.00161	**3**	0.03077	**3**	↑ **2**	↑ **2**
An unusual, hard to recognize photo of a {}.	0.6236	6	0.00095	8	0.03024	6	↓ **2**	–
A photo of a {}, with a watermark.	0.6232	7	0.00075	9	0.02991	9	↓ **2**	↓ **2**
A hand-drawn illustration of a {}.	0.6212	8	0.00070	10	0.02997	7	↓ **2**	↑ **1**
A black and white photo of a {}.	0.6206	9	0.00153	4	0.03046	5	↑ **5**	↑ **4**
A painting of a {}.	0.6198	10	0.00102	7	0.02992	8	↑ **3**	↑ **2**
A blurry photo of a {}.	0.6196	11	0.00032	13	0.02934	11	↓ **2**	–
A photo of a {}.	0.6184	12	-0.00006	16	0.02884	12	↓ **4**	–
A poor quality, grainy photo of a {}.	0.6112	13	0.00036	12	0.02866	14	↑ **1**	↓ **1**
A photo focusing on the shape of a {}.	0.6100	14	-0.00041	18	0.02776	17	↓ **4**	↓ **3**
A photo focusing on the texture of a {}.	0.6098	15	-0.00027	17	0.02787	16	↓ **2**	↓ **1**
A photo of a {}, with text over it.	0.6076	16	0.00041	11	0.02847	15	↑ **5**	↑ **1**
A zoomed-out photo of a {}.	0.6046	17	-0.00084	20	0.02699	19	↓ **3**	↓ **2**
A photo of a {} in its natural environment.	0.6046	18	0.00113	6	0.02871	13	↑ **12**	↑ **5**
A photo focusing closely on the details of a {}.	0.6032	19	-0.00064	19	0.02688	20	–	↓ **1**
A close-up photo of a {}.	0.6002	20	-0.00107	22	0.02633	21	↓ **2**	↓ **1**
A sketch of a {}.	0.5992	21	0.00010	14	0.02741	18	↑ **7**	↑ **3**
A macro shot of a {}.	0.5872	22	-0.00092	21	0.02536	22	↑ **1**	–

4. An instance consists of the first $|U|$ profiles of such a list (i.e., of $\mathcal{U}$).

Hence the fraction of correct models n_1/n is close to the parameter p. Furthermore, for each n_1 the profiles that assign the incorrect votes over most of the incorrect labels are generated first. Note that when the incorrect votes are distributed evenly, many coalitions result in a tie, increasing the runtime of our algorithms for $v_{\mathsf{MC}}(\cdot)$. For the binary case it holds $r_{\max}(n_1) \leq 1$, i.e., a profile is determined by n_1 uniquely. We vary $p \in \{0.1, 0.5, 0.9\}$, i.e., representing ensembles of base classifiers with low, medium and high accuracy. In the binary case we vary $n \in \{100, 300, 1\,000, 3\,000, 10\,000\}$ and $|U| \in \{10, 20, 40, 80, 160\}$. In the multiclass case we vary $n \in \{10, 25, 50, 100\}$, $\ell \in \{2, 5, 10, 20, 50, 100, 1000\}$ and $|U| \in \{10, 10^2, 10^3\}$. We do not consider configurations with less than the fixed number $|U|$ of profiles. We run each instance five times and divide its median runtime by $|U|$. For each (p, n, ℓ) we report the median of these values over $|U|$, which we denote as the runtime per unique profile. The time complexity of our methods is linear in $|U|$, thus dividing by $|U|$ isolates the influence of all other parameters on the runtime.

Results. Figure 9 reports the runtime per unique profile in log-log scale, where each row corresponds to a different value of p. In each panel the horizontal axis corresponds to the bound on the time complexity of the corresponding theorem (i.e., according to the setting and power index considered). That is

- the left column reports ENPOWER-β and ENPOWER-ϕ for the binary classification setting ($\ell = 2$) as a function of n (Theorem 4.3);

Table 10: Prompt ensemble evaluation for LLM zero-shot multiclass image classification (PETS). Each prompt corresponds to a different model of $\mathcal{C}$. All values and rankings are computed on the validation split $\mathcal{D}_{\text{val}}$. Columns Δ_β and Δ_ϕ display the difference in the validation rankings induced by the power indices $\beta(G_{\mathsf{MC}})$ and $\phi(G_{\mathsf{MC}})$ with respect to validation accuracy.

Prompt template	**Accuracy**		**Banzhaf**		**Shapley**		Δ_β	Δ_ϕ
	Value	Rank	Value	Rank	Value	Rank		
A poor quality, grainy photo of a {}.	0.8648	**1**	0.00434	**1**	0.04529	**1**	–	–
A blurry photo of a {}.	0.8539	**2**	0.00328	**2**	0.04335	**2**	–	–
A close-up photo of a {}.	0.8522	**3**	0.00327	**3**	0.04312	**3**	–	–
A cropped photo of a {}.	0.8506	4	0.00188	4	0.04175	4	–	–
A heavily compressed JPEG photo of a {}.	0.8462	5	0.00095	9	0.04059	7	↓ **4**	↓ **2**
A photo focusing on the texture of a {}.	0.8451	6	0.00163	6	0.04088	6	–	–
A photo of a {}.	0.8446	7	0.00019	10	0.03962	8	↓ **3**	↓ **1**
A pixelated photo of a {}.	0.8435	8	0.00180	5	0.04099	5	↑ **3**	↑ **3**
A high-resolution, crisp photo of a {}.	0.8375	9	-0.00002	13	0.03878	9	↓ **4**	–
A photo of a {}, with text over it.	0.8337	10	-0.00045	15	0.03811	12	↓ **5**	↓ **2**
A zoomed-out photo of a {}.	0.8310	11	0.00009	11	0.03839	11	–	–
A photo of a {} in its natural environment.	0.8261	12	-0.00033	14	0.03759	14	↓ **2**	↓ **2**
A photo of a {}, with a watermark.	0.8250	13	-0.00142	17	0.03657	16	↓ **4**	↓ **3**
A sketch of a {}.	0.8233	14	-0.00001	12	0.03768	13	↑ **2**	↑ **1**
A photo focusing closely on the details of a {}.	0.8201	15	-0.00092	16	0.03642	17	↓ **1**	↓ **2**
A macro shot of a {}.	0.8195	16	0.00126	7	0.03869	10	↑ **9**	↑ **6**
A photo focusing on the shape of a {}.	0.8168	17	-0.00225	19	0.03494	18	↓ **2**	↓ **1**
A painting of a {}.	0.8141	18	-0.00247	21	0.03464	19	↓ **3**	↓ **1**
A photo of a {}, cluttered background.	0.8108	19	-0.00236	20	0.03443	20	↓ **1**	↓ **1**
An unusual, hard to recognize photo of a {}.	0.8086	20	-0.00344	22	0.03313	21	↓ **2**	↓ **1**
A black and white photo of a {}.	0.7999	21	0.00097	8	0.03676	15	↑ **13**	↑ **6**
A hand-drawn illustration of a {}.	0.7863	22	-0.00198	18	0.03286	22	↑ **4**	–

- the middle column reports ENPOWER-β in the multiclass classification setting as a function of $n\ell^2$ (Theorem 4.7); and
- the right column reports ENPOWER-ϕ as a function of $n^3\ell$ (Theorem 4.9).

Regarding the figures concerning the multiclass case each line connects the values of n for a fixed ℓ, colored by the value of ℓ. The dotted line in each panel corresponds to theoretical bound. In the binary classification setting, the runtime per unique profile of ENPOWER grows linearly with n (with a constant slightly smaller than 1), as predicted by Theorem 4.3. In the multiclass setting for ENPOWER-β for a fixed value of ℓ the runtime scales with n almost linearly as from the bound in Theorem 4.7. The only setting where the runtime appears to be slightly larger than linear in n, is for $p = 0.1$ and a high value of ℓ, where the number of incorrect labels approaches n. Interestingly, we observe that increasing ℓ for a fixed n does not result in the $\mathcal{O}(\ell^2)$ of our bound, that is our bounds can be likely improved. Similarly, the theoretical runtime of ENPOWER-ϕ for large values of $n^3\ell$ is likely pessimistic given that in practice our methods are much more efficient.

Table 11: Prompt ensemble evaluation for LLM zero-shot multiclass image classification (IMAGENET-A). Each prompt corresponds to a different model of $\mathcal{C}$. All values and rankings are computed on the validation split $\mathcal{D}_{\text{val}}$. Columns Δ_β and Δ_ϕ display the difference in the validation rankings induced by the power indices $\beta(G_{\text{MC}})$ and $\phi(G_{\text{MC}})$ with respect to validation accuracy.

Prompt template	**Accuracy**		**Banzhaf**		**Shapley**		Δ_β	Δ_ϕ
	Value	Rank	Value	Rank	Value	Rank		
A poor quality, grainy photo of a {}.	0.3136	**1**	0.00359	**1**	0.01936	**1**	–	–
A pixelated photo of a {}.	0.3029	**2**	0.00242	**3**	0.01738	**3**	↓ **1**	↓ **1**
A photo of a {}, with a watermark.	0.2989	**3**	0.00096	8	0.01581	6	↓ **5**	↓ **3**
A blurry photo of a {}.	0.2989	4	0.00184	4	0.01656	4	–	–
An unusual, hard to recognize photo of a {}.	0.2981	5	0.00295	**2**	0.01750	**2**	↑ **3**	↑ **3**
A heavily compressed JPEG photo of a {}.	0.2949	6	0.00154	5	0.01583	5	↑ **1**	↑ **1**
A zoomed-out photo of a {}.	0.2893	7	0.00066	10	0.01468	8	↓ **3**	↓ **1**
A cropped photo of a {}.	0.2888	8	0.00071	9	0.01463	9	↓ **1**	↓ **1**
A hand-drawn illustration of a {}.	0.2872	9	0.00137	6	0.01501	7	↑ **3**	↑ **2**
A photo of a {}, with text over it.	0.2869	10	0.00049	11	0.01428	10	↓ **1**	–
A photo of a {}.	0.2853	11	-0.00027	15	0.01353	12	↓ **4**	↓ **1**
A photo of a {}, cluttered background.	0.2840	12	-0.00035	16	0.01337	13	↓ **4**	↓ **1**
A photo of a {} in its natural environment.	0.2797	13	0.00116	7	0.01420	11	↑ **6**	↑ **2**
A painting of a {}.	0.2779	14	-0.00000	13	0.01309	14	↑ **1**	–
A photo focusing on the shape of a {}.	0.2776	15	-0.00043	17	0.01261	16	↓ **2**	↓ **1**
A high-resolution, crisp photo of a {}.	0.2765	16	-0.00136	20	0.01162	18	↓ **4**	↓ **2**
A black and white photo of a {}.	0.2739	17	0.00024	12	0.01289	15	↑ **5**	↑ **2**
A sketch of a {}.	0.2704	18	-0.00004	14	0.01240	17	↑ **4**	↑ **1**
A photo focusing on the texture of a {}.	0.2579	19	-0.00080	18	0.01048	19	↑ **1**	–
A photo focusing closely on the details of a {}.	0.2544	20	-0.00180	21	0.00934	20	↓ **1**	–
A close-up photo of a {}.	0.2365	21	-0.00218	22	0.00727	22	↓ **1**	↓ **1**
A macro shot of a {}.	0.2349	22	-0.00134	19	0.00808	21	↑ **3**	↑ **1**

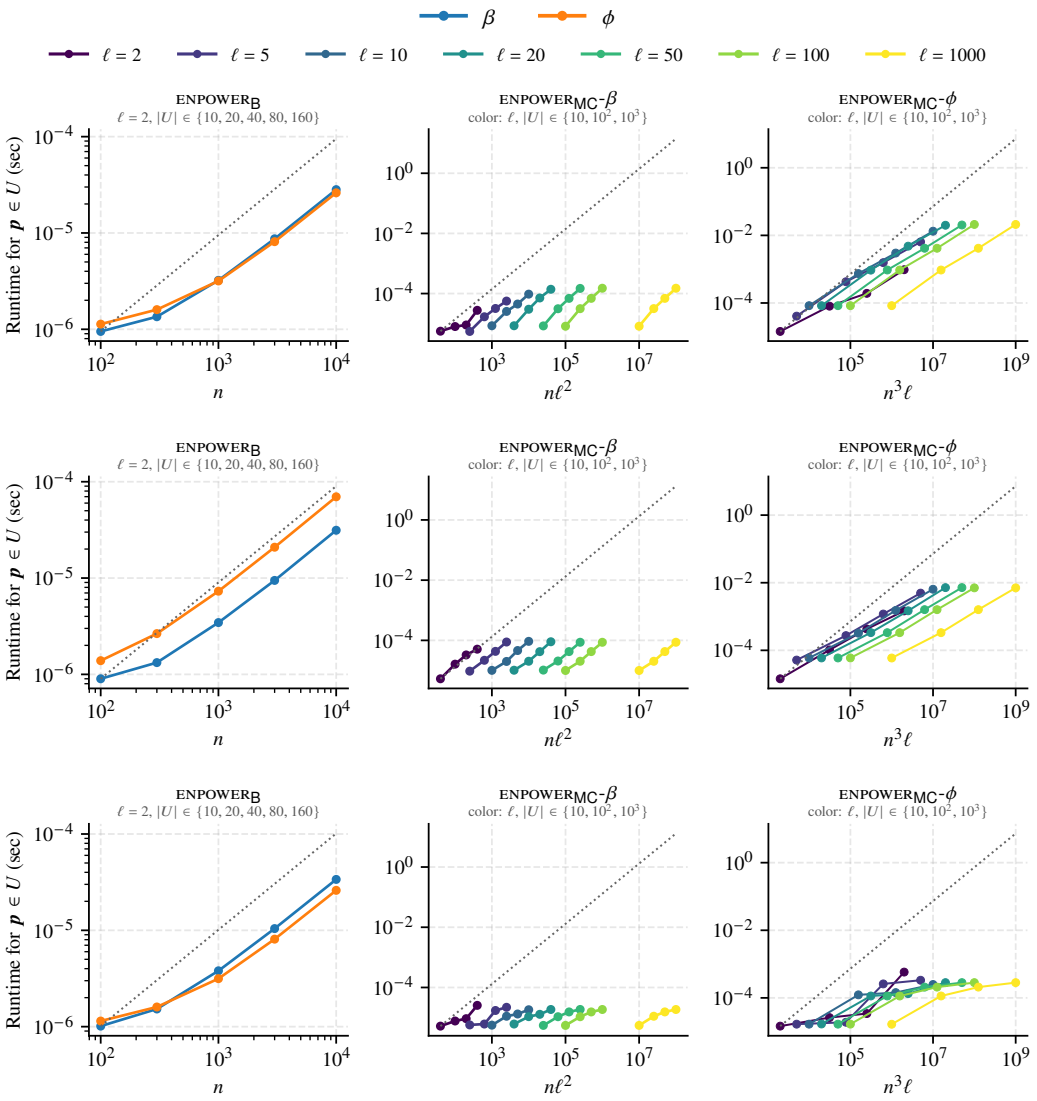


Figure 9: Runtime to process a unique vote profile on synthetic instances, where $p = 0.1$ (top), $p = 0.5$ (middle) and $p = 0.9$ (bottom). The y-axis reports the measured time to compute the power indices under a vote profile $\boldsymbol{p} \in U$ with n players and ℓ labels. See App. G.6 for a detailed description of each panel.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We refer to Sec. 3, Sec. 4 and Sec. 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Sec. 3, Sec. 5 and App. G. We also provide a dedicated "Limitations" section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes],

Justification: see App. C.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We describe our experimental setup and environment in Sec. 5 and App. F, including all details about data processing and model training. We also release our code `https://github.com/iliesarpe/enpower`, which has a container that build the environment used to fully replicate our results. We also provide all the pseudocodes of our methods in App. D.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Our experimental evaluation uses publicly available datasets (see App. F). Our code is released online `https://github.com/iliesarpe/enpower`. In addition our code contains all the scripts (and a related container) to process all datasets according to our exact setup.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: See Sec. 5 and App. F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: concerning runtime: We only report the standard deviation of running the sampling algorithm `MC`. We did not run multiple times baselines `WMG` given their high runtime, and all other algorithms have very stable runtime. Concerning predictive metrics: most of our algorithms are deterministic so they report the same results across different runs. In addition, we do not perform multiple splits of the dataset for prediction tasks given the high runtime of executing all methods.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We describe the experimental setup in Sec. 5 and App. F. In addition all plots report clearly the runtime necessary to perform all experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Upon reviewing the NeurIPS Code of Ethics we confirm that the research presented in this paper complies with its principles and requirements.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: We provide statements supporting the use of our methods for enhancing the interpretability of ensemble models. We do not foresee any negative societal impact due to a misuse of our methods.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: Our work is concerned with providing new algorithms to enhance the interpretability of ensemble models. While our techniques can be *intentionally* misused to reduce ensemble performances, we do not see any direct harm caused by our methods.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We credit all resources that were used in App. F and Sec. 5, and in our code repository `https://github.com/iliesarpe/enpower`.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Our code implementation (`https://github.com/iliesarpe/enpower`) will be publicly released, with a supporting documentation and examples.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our work does not consider crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

Justification: Our work does not consider crowdsourcing or human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: We provide an accompanying LLM usage statement.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.