Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

Supplementary Information

HEMDAG: a family of modular and scalable hierarchical ensemble methods to improve Gene Ontology term prediction

Marco Notaro[1], Marco Frasca[1], Alessandro Petrini[1], Jessica Gliozzo[1], Elena Casiraghi[1], Peter Robinson[2] and Giorgio Valentini[1,3,4]*

[1]AnacletoLab - Dipartimento di Informatica, Università degli Studi di Milano, Via Celoria 18, 20133 Milano, Italy
[2]The Jackson Laboratory for Genomic Medicine: Farmington, CT, US
[3]CINI, National Laboratory in Artificial Intelligence and Intelligent Systems—AIIS, Roma, Italy
[4] Data Science Research Center, Università degli Studi di Milano, 20133 Milano, Italy

*To whom correspondence should be addressed.

Supplementary Table of Contents

Supplementary List of Figures

Supplementary List of Tables

S1 Consistency of the hierarchical predictions

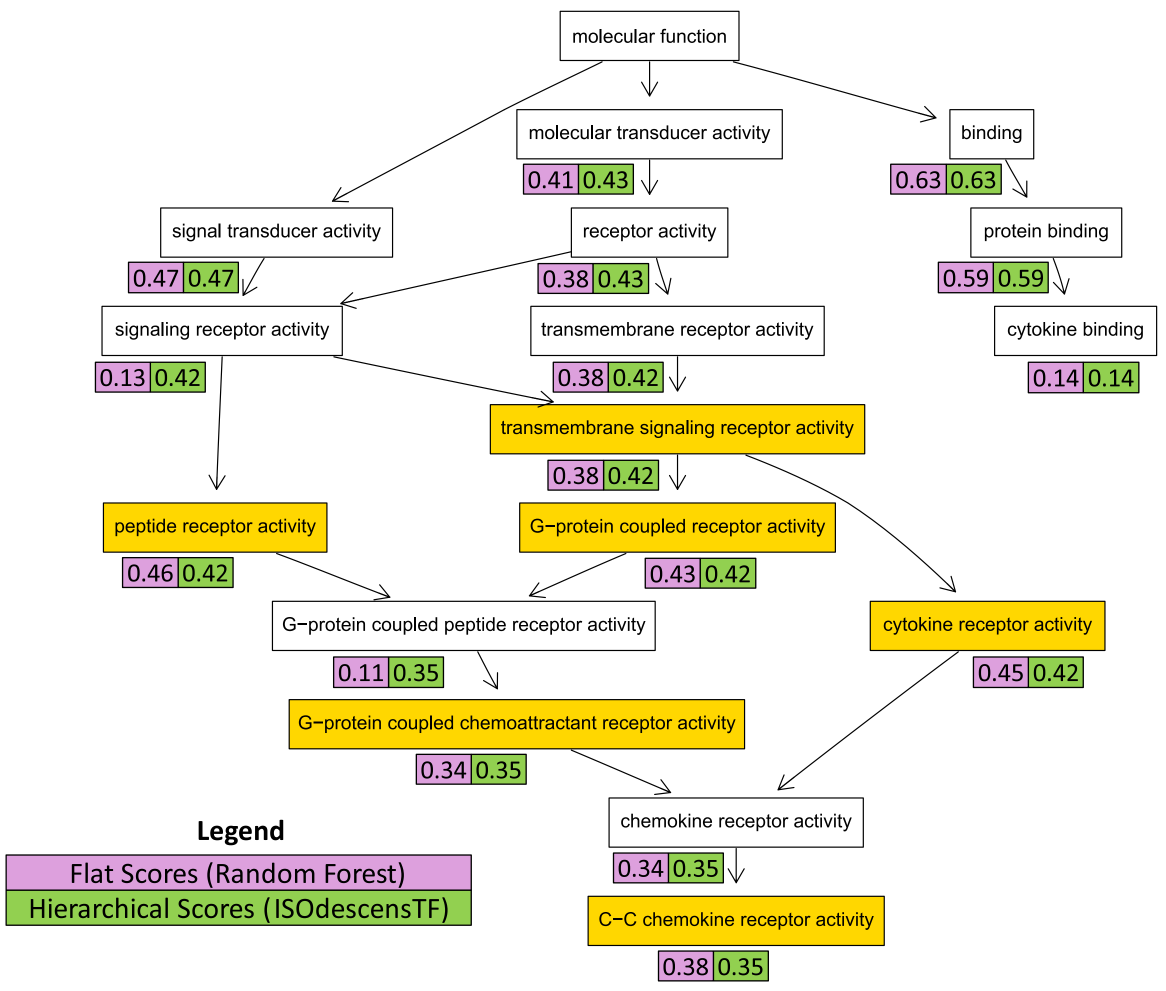

molecular function
molecular transducer activity
0.41 0.43
binding
0.63 0.63
signal transducer activity
0.47 0.47
receptor activity
0.38 0.43
protein binding
0.59 0.59
signaling receptor activity
0.13 0.42
transmembrane receptor activity
0.38 0.42
cytokine binding
0.14 0.14
transmembrane signaling receptor activity
0.38 0.42
peptide receptor activity
0.46 0.42
G−protein coupled receptor activity
0.43 0.42
G−protein coupled peptide receptor activity
0.11 0.35
cytokine receptor activity
0.45 0.42
G−protein coupled chemoattractant receptor activity
0.34 0.35
chemokine receptor activity
0.34 0.35
C−C chemokine receptor activity
0.38 0.35
Legend
Flat Scores (Random Forest)
Hierarchical Scores (ISOdescensTF)

Fig. S1. Flat and hierarchical predictions for the Mouse protein `ENSMUSP00000095029`. The numbers close to each GO term represent flat and hierarchically corrected scores, respectively magenta and green boxes. It is easy to see how the predictions of the hierarchical approach (green boxes) always obey to the True-Path-Rule, (i.e. the scores of the parent nodes are always larger or equal than those of their children nodes), while the flat predictions (magenta boxes) are inconsistent for 6 GO terms (yellow colored boxes.)

S2 Pseudocode and complexity of the ISO-TPR algorithm

Fig. S2 shows the high level pseudo-code of ISO-TPR.

```
Input:
- $G =< V, E >$
- $V = \{1, 2, \ldots, |V|\}$
- $\hat{\boldsymbol{y}} =< \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|V|} >, \quad \hat{y}_i \in [0, 1]$
- $\boldsymbol{w} =< w_1, w_2, \ldots, w_{|V|} >, \quad w_i \in [0, 1]$
begin algorithm
01:  A. $dist := \forall i \in V$ ComputeMaxDist $(G, root(G))$
02:  B. Per-level bottom-up visit of $G$:
03:      for each $d$ from $\max(dist)$ to $0$ do
04:        $N_d := \{i | dist(i) = d\}$
05:        for each $i \in N_d$ do
06:          Select the set $\phi_i$ of "positive" children
07:          $\bar{y}_i := \frac{1}{1+|\phi_i|}(\hat{y}_i + \sum_{j \in \phi_i} \bar{y}_j)$
08:        end for
09:      end for
10:  C. GPAV algorithm
11:  $\hat{\boldsymbol{y}} := \bar{\boldsymbol{y}}$
12:  $V = \{1, 2, \ldots, |V|\}$ topologically ordered;
13:  $H := V$
14:  $\forall i \in V$ set $B_i = \{i\}$; $B_i^- = i^-$; $U_i = \hat{y}_i$; $W_i = w_i$;
15:  for each $k$ from $1$ to $|V|$ do
16:      while exists $i \in B_k^-$ such that $U_i > U_k$ do
17:        find $j \in B_k^-$ such that $U_j := max\{U_i : i \in B_k^-\}$
18:        $H := H \setminus \{j\}$
19:        $B_k^- := B_j^- \cup B_k^- \setminus \{j\}$
20:        $U_k := (W_k U_k + W_j U_k)/(W_k + W_j)$
21:        $B_k := B_k \cup B_j$
22:        $W_k := W_k + W_j$
23:        $\forall i \in B_k$ and $\forall k \in H$ set $\bar{y} := U_k$
24:      end while
25:      $\bar{y} := U_k \, \forall i \in B_k$ and $\forall k \in H$
26:  end for
end algorithm
Output:
- $\bar{\boldsymbol{y}} =< \bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{|V|} >$
```

Fig. S2. ISOtonic True-Path-Rule for DAG (ISO-TPR) pseudo-code.

The computational complexity of block A is $\mathcal{O}(|V| + |E|)$ (topological sort), whereas the complexity of block B is $\mathcal{O}(|V|)$ when graphs are sparse. Finally, block C has a complexity $\mathcal{O}(|V|^2)$, thus resulting in an overall complexity of $\mathcal{O}(|V|^2)$. Nevertheless, if we use the HTD algorithm in lieu of GPAV, the complexity of block C is $\mathcal{O}(|V| + |E|)$, and it follows that the overall computational complexity is nearby linear in the number of nodes, since the GO DAG is sparse.

S3 HEMDAG family

Table S1 summarizes the 20 hierarchical ensemble algorithms of the HEMDAG family. In the following sections each HEMDAG algorithm is described in detail. A step-by-step tutorial available at `https://hemdag.readthedocs.io` shows how to apply the HEMDAG algorithms to real data.

Table S1. HEMDAG family of hierarchical ensemble algorithms. In bold are highlighted the novel HEMs algorithms presented for the first time in this manuscript. The acronyms TF (Threshold-Free), T (Threshold), W (Weight) and WT (Weight-Threshold) at the end of the HEMs names refer to the bottom-up strategy used to select the positive nodes. "None" in the column **Bottom-up step**, points out that HTD and GPAV algorithm are characterized only by the consistency step. All the other algorithms choose in the bottom-up step positive children or descendants nodes according to the family they belong to.

HEMs	**Family**	**Bottom-up Step**	**Consistency Step**	**Description**	**Type**
HTD	HTD	None	HTD	satisfy the hierarchical constraints. No bottom-up step is applied	Parameter-free
GPAV	GPAV		GPAV	satisfy the hierarchical constraints. No bottom-up step is applied	
tprTF	TPR-DAG	Children	HTD	ϕ_i chosen by applying formula (3). HTD is applied to satisfy hierarchical constraints	
ISOtprTF	ISO-TPR		GPAV	ϕ_i chosen by applying formula (3). GPAV is applied to satisfy hierarchical constraints	
descensTF	DESCENS	Descendants	HTD	Δ_i chosen by applying formula (8). HTD is applied to satisfy hierarchical constraints	
ISOdescensTF	ISO-DESCENS		GPAV	Δ_i chosen by applying formula (8). GPAV is applied to satisfy hierarchical constraints	
tprT	TPR-DAG	Children	HTD	ϕ_i chosen by applying formula (4) or (5). HTD is applied to satisfy hierarchical constraints	Parametric
tprW				ϕ_i chosen by applying formula (6). HTD is applied to satisfy hierarchical constraints	
tprWT				combination of tprT and tprW. HTD is applied to satisfy hierarchical constraints	
ISOtprT	ISO-TPR		GPAV	ϕ_i chosen by applying formula (4) or (5). GPAV is applied to satisfy hierarchical constraints	
ISOtprW				ϕ_i chosen by applying formula (6). GPAV is applied to satisfy hierarchical constraints	
ISOtprWT				combination of ISOtprT and ISOtprW. GPAV is applied to satisfy hierarchical constraints	
descensT	DESCENS	Descendants	HTD	Δ_i chosen by applying formula (9) or (10). HTD is applied to satisfy hierarchical constraints	
descensW				Δ_i chosen by applying formula (11). HTD is applied to satisfy hierarchical constraints	
desensWT				combination of descensT and descensW. HTD is applied to satisfy hierarchical constraints	
descensTAU				Δ_i chosen by applying formula (12). HTD is applied to satisfy hierarchical constraints	
ISOdescensT	ISO-DESCENS		GPAV	Δ_i chosen by applying formula (9) or (10). GPAV is applied to satisfy hierarchical constraints	
ISOdescensW				Δ_i chosen by applying formula (11). GPAV is applied to satisfy hierarchical constraints	
ISOdescensWT				combination of ISOdescensT and ISOdescensW. GPAV is applied to satisfy hierarchical constraints	
ISOdescensTAU				Δ_i chosen by applying formula (12). GPAV is applied to satisfy hierarchical constraints	

S3.1 HTD

HTD is the simplest algorithm of the HEMDAG family. The HTD algorithm modifies the flat scores according to the hierarchy of a DAG g through a unique run across the nodes of the graph. For a given example x, the flat predictions $f(x) = \hat{y}$ are hierarchically corrected to $\bar{y}$, by per-level visiting the nodes of the DAG from top to bottom according to the following simple rule:

$$\bar{y}_i := \begin{cases} \hat{y}_i & \text{if} \quad i \in root(G) \\ \min_{j \in par(i)} \bar{y}_j & \text{if} \quad \min_{j \in par(i)} \bar{y}_j < \hat{y}_i \\ \hat{y}_i & \text{otherwise} \end{cases} \tag{1}$$

The pseudocode of the HTD algorithm is presented and discussed in Notaro *et al.* (2017).

S3.2 GPAV

The GPAV algorithm is described in the section 2.3.2 of the main manuscript and the pseudo-code is presented in the step C on the Fig. S2.

S3.3 ISO-TPR family

ISO-TPR is a family of methods that integrate isotonic regression (Barlow, 1972) and TPR algorithms (Valentini, 2011; Notaro *et al.*, 2017). The *vanilla* ISO-TPR algorithm adopt a per-level bottom-up visit of the DAG to modify the flat predictions $\hat{y}_i$ according to the following equation:

$$\bar{y}_i := \frac{1}{1 + |\phi_i|}(\hat{y}_i + \sum_{j \in \phi_i} \bar{y}_j) \tag{2}$$

where ϕ_i are the positive children of i. Different ISO-TPR variants can be designed according to the strategy used to select the positive children ϕ_i:

1. *threshold-free strategy* (TF): the positive nodes are those children that can increment the score of the node i, that is those nodes that achieve a score higher than that of their parents:
$$\phi_i := \{j \in child(i) | \bar{y}_j > \hat{y}_i\} \tag{3}$$
2. *threshold strategy* (T):
   a. a unique threshold $\bar{t}$ is a priori selected for all nodes to determine the set of positives
   $$\phi_i := \{j \in child(i) | \bar{y}_j > \bar{t}\}, \forall i \in V \tag{4}$$
   For instance, if the flat predictions are probabilities, it could be meaningful to a priori select $\bar{t} = 0.5$
   b. a threshold is selected to maximize some imbalance-aware performance metric $\mathcal{M}$ estimated on the training data, e.g. the F-measure or the Area Under the Precision-Recall curve (AUPRC). Namely, the threshold is selected to maximize the measure $\mathcal{M}(j, t)$ on the training data for the term j with respect to the threshold t. The corresponding set of positives is:
   $$\phi_i := \{j \in child(i) | \bar{y}_j > t^*_j, t^*_j = \arg\max_t \mathcal{M}(j, t)\}, \forall i \in V \tag{5}$$
   Internal cross-validation has been be used to select t^*_j within a set of possible thresholds $t \in (0, 1)$;

By applying eq. 3 we design the algorithm ISOtprTF. Instead, by adopting eq. 4 or 5 we design the algorithm ISOtprT. We also design the weighted ISO-TPR version (ISOtprW) by adding a weight $w \in [0, 1]$ to balance the contribution of the parent node i and its positive children:

$$\bar{y}_i := w\hat{y}_i + \frac{(1-w)}{|\phi_i|} \sum_{j \in \phi_i} \bar{y}_j \tag{6}$$

If $w = 1$, no weight is attributed to the children and the ISO-TPR reduces to the GPAV algorithm. If $w = 0$, only the predictors associated to the positive children nodes "vote" to predict node i. In the intermediate cases we attribute more importance to the predictor for the node i or to its children depending on the values of w. The weighted-threshold variant ISOtprWT, characterized by a double parametrization, is designed by combining the variants ISOtprT and ISOtprW. The pseudo-code of the ISO-TPR algorithms is presented in Fig. S2.

S3.4 ISO-DESCENS family

As shown in Valentini (2011) for tree-based hierarchies, the contribution of the descendants of a given node decays exponentially with their distance from the node itself and it is straightforward to see that this property also holds for DAG structured taxonomies. To overcome this limitation and in order to enhance the contribution of the most specific nodes to the overall decision of the ensemble we designed ISO-DESCENS. The novelty of ISO-DESCENS consists in strongly considering the contribution of all the descendants of each node instead of only that of its children. Therefore ISO-DESCENS predictions are more influenced by the information embedded in the leaf nodes, that are the classes containing the most informative and meaningful information from a biological standpoint. ISO-DESCENS variants can be designed on the choice of the positive descendants Δ_i. The same strategies adopted for the choice of ϕ_i described in section S3.3 can be also adopted for the choice of the positive descendants Δ_i, simply by replacing ϕ_i with Δ_i and $child(i)$ with $desc(i)$ in the equations shown in S3.3. More precisely, the *vanilla* ISO-DESCENS algorithm adopts a per-level bottom-up visit of the DAG to modify the flat predictions $\hat{y}_i$ according to the following equation:

$$\bar{y}_i := \frac{1}{1 + |\Delta_i|}(\hat{y}_i + \sum_{j \in \Delta_i} \bar{y}_j) \tag{7}$$

The ISOdescensTF variant selects the positive descendants nodes according to a threshold-free strategy:

$$\Delta_i := \{j \in desc(i) | \bar{y}_j > \hat{y}_i\} \tag{8}$$

Instead, the ISOdescensT variant selects the positive nodes according to a threshold strategy. The threshold can be unique for all the nodes:

$$\Delta_i := \{j \in desc(i) | \bar{y}_j > \bar{t}\}, \forall i \in V \tag{9}$$

or it can be selected by maximizing an imbalance-aware performance metric, such as AUPRC or F_{max}, on the training data:

$$\Delta_i := \{j \in desc(i) | \bar{y}_j > t_j^*, t_j^* = \arg\max_t \mathcal{M}(j, t)\}, \forall i \in V \tag{10}$$

The weighted ISO-TPR version (ISOtprW) adds a weight $w \in [0, 1]$ to balance the contribution of the parent node i and its positive descendants:

$$\bar{y}_i := w\hat{y}_i + \frac{(1-w)}{|\Delta_i|} \sum_{j \in \Delta_i} \bar{y}_j \tag{11}$$

The weighted-threshold variant ISOdescensWT, characterized by two hyper-parameters, is designed by combining the variants ISOdescensT and ISOdescensW. Finally, we also designed a variant specific only for ISO-DESCENS, named descensTAU, by adding a weight $\tau \in [0, 1]$ to simultaneously leverage the contribution of positive children and descendants (excluding the children) of a term:

$$\bar{y}_i := \frac{\tau}{1 + |\phi_i|}(\hat{y}_i + \sum_{j \in \phi_i} \bar{y}_j) + \frac{1-\tau}{1 + |\delta_i|}(\hat{y}_i + \sum_{j \in \delta_i} \bar{y}_j) \tag{12}$$

where $\delta_i = \Delta_i \setminus \phi_i$ are the descendants of i without its children. If $\tau = 1$ we consider only the contribution of the positive children of i, if $\tau = 0$ only the descendants that are not children contribute to the score, while for intermediate values of τ we can balance the contribution of ϕ_i and δ_i positive nodes. All the DESCENS variants adopt the GPAV algorithm to assure the consistency of the predictions.

S3.5 TPR-DAG family

The algorithms belonging to the TPR-DAG family choose ϕ_i by using one of the selection strategies described in section S3.3 and by adopting HTD algorithm instead of the GPAV approach in the consistency step.

S3.6 DESCENS family

The algorithms belonging to the DESCENS family choose Δ_i by using one of the selection strategies described in section S3.4 and by adopting HTD algorithm instead of the GPAV approach in the consistency step.

S4 Data preparation pipeline

In Fig. S3 is depicted the pipeline adopted to prepare the dataset for the experiments discussed in Section 3 of the main manuscript. The main goal of this pipeline is to limit as much as possible the number of the unmapped identifiers for the couple UniProt-AC towards STRING-ID in the UniProtKB mapping provided by the GOA database. Our data preparation pipeline assures the uniqueness of couples of identifiers UniProt - STRING. Indeed we just isolated from the GOA gaf file all UniProt proteins annotated with GO terms supported by an experimental evidence code. These UniProt identifiers are guaranteed to be unique by GOA itself. Then we used these identifiers to extrapolate respectively from SwissProt and TrEMBL database the relative protein sequences. Next, we blasted the unmapped UniProt identifiers against the STRING database in order to recover the couple of identifiers UniProt-STRING. This step retrieves unmapped couples of identifiers UniProt-STRING only if they have an identity of 100% and if the retrieved couple of identifiers do not exist in the UniProtKB mapping file provided by GOA database. The Perl code of the part of the pipeline that assures that no duplicates are present in out dataset is available at the following link: `https://github.com/marconotaro/godata-pipe/blob/main/build_data/check_blastp_result.pl`. The most time-consuming step of this pipeline, is the part in which we isolate the FASTA sequence of the unmapped UniProt-AC identifiers from the TrEMBL database (22Gb zipped). This step requires about a couple of hours (by using a machine having an Intel Xeon CPU E5-2630 2.60GHz with 128GB of RAM, but it must be repeated just once (separately for each organism).

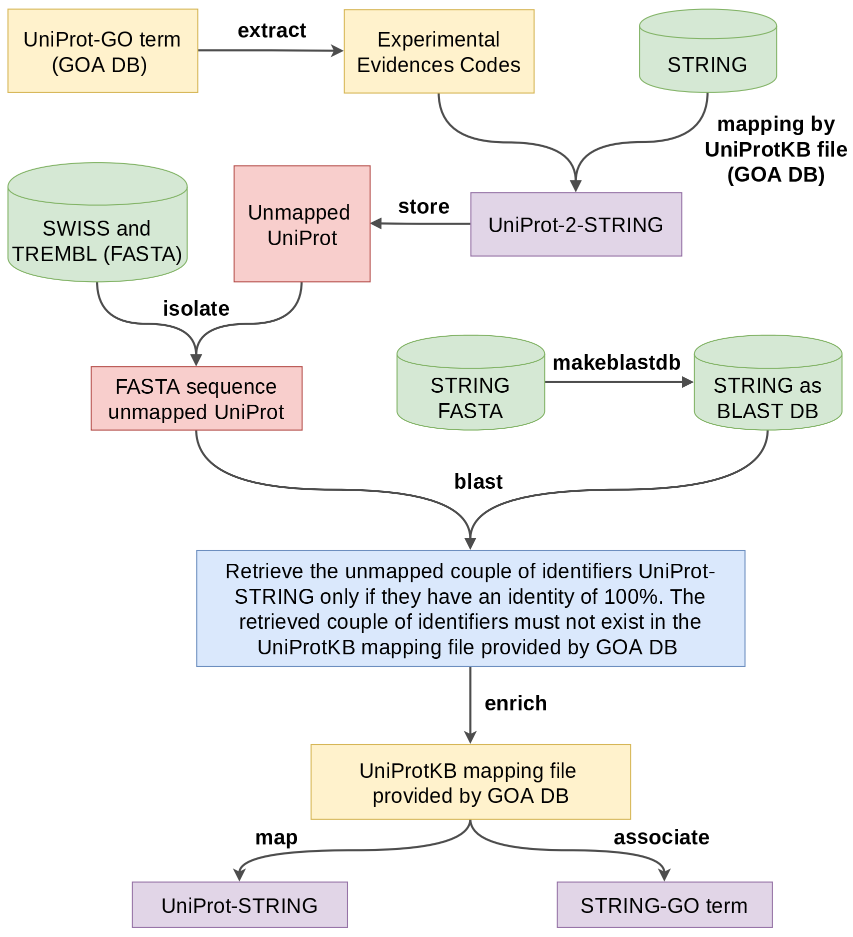

UniProt-GO term (GOA DB)
extract
Experimental Evidences Codes
STRING
mapping by UniProtKB file (GOA DB)
SWISS and TREMBL (FASTA)
Unmapped UniProt
store
UniProt-2-STRING
isolate
FASTA sequence unmapped UniProt
STRING FASTA
makeblastdb
STRING as BLAST DB
blast
Retrieve the unmapped couple of identifiers UniProt-STRING only if they have an identity of 100%. The retrieved couple of identifiers must not exist in the UniProtKB mapping file provided by GOA DB
enrich
UniProtKB mapping file provided by GOA DB
map
associate
UniProt-STRING
STRING-GO term

Fig. S3. Pipeline adopted to prepare the datasets.

The first entry of the column **Statistics** of both Table S2 and Table S3 refers the total unique number of UniProt-AC associated with a GO terms, whereas the fourth entry indicates the total number of unique GO terms associated to a UniProt-AC. These two entries are identical before and after applying the pipeline illustrated in Fig. S3, because the GOA database provides annotations by using UniProt-AC as primary identifier. The second and the third entry show respectively the number of mapped and unmapped STRING-ID versus UniProt-AC before and after enriching the UniProtKB mapping file. Finally, the fifth and the sixth entry refers respectively to the total number of GO terms associated with a STRING-ID and to the total number of GO terms not associated with a STRING-ID.

Table S2. Statistics before and after applying the data preparation pipeline illustrated in Fig. S3 by using the GOA release of December 2017.

Organism	Statistics	GOA db	After Enrichment
CAEEL	UniProt-AC associated with GO term	3449	3449
	STRING-ID mapped with UniProt-AC	3338	3405
	STRING-ID unmapped with UniProt-AC	111	44
	GO term associated with UniProt-AC	3083	3083
	GO term associated with STRING-ID	3052	3061
	GO term not associated with STRING-ID	31	22
CHICK	UniProt-AC associated with GO term	740	740
	STRING-ID mapped with UniProt-AC	521	544
	STRING-ID unmapped with UniProt-AC	219	196
	GO term associated with UniProt-AC	1204	1204
	GO term associated with STRING-ID	1018	1048
	GO term not associated with STRING-ID	186	156
DANRE	UniProt-AC associated with GO term	4227	4227
	STRING-ID mapped with UniProt-AC	3566	3736
	STRING-ID unmapped with UniProt-AC	661	491
	GO term associated with UniProt-AC	3051	3051
	GO term associated with STRING-ID	2840	2880
	GO term not associated with STRING-ID	211	171
DROME	UniProt-AC associated with GO term	6179	6179
	STRING-ID mapped with UniProt-AC	5675	6025
	STRING-ID unmapped with UniProt-AC	504	154
	GO term associated with UniProt-AC	5510	5510
	GO term associated with STRING-ID	5393	5444
	GO term not associated with STRING-ID	117	66
HUMAN	UniProt-AC associated with GO term	13603	13603
	STRING-ID mapped with UniProt-AC	12795	13160
	STRING-ID unmapped with UniProt-AC	808	443
	GO term associated with UniProt-AC	11134	11134
	GO term associated with STRING-ID	11006	11042
	GO term not associated with STRING-ID	128	92
MOUSE	UniProt-AC associated with GO term	11265	11265
	STRING-ID mapped with UniProt-AC	10592	10948
	STRING-ID unmapped with UniProt-AC	673	317
	GO term associated with UniProt-AC	11101	11101
	GO term associated with STRING-ID	10959	11029
	GO term not associated with STRING-ID	142	72

Table S3. Statistics before and after applying the data preparation pipeline illustrated in Fig. S3 by using the GOA release of June 2020.

Organism	Statistics	GOA db	After Enrichment
CAEEL	UniProt-AC associated with GO term	3856	3856
	STRING-ID mapped with UniProt-AC	3738	3827
	STRING-ID unmapped with UniProt-AC	118	29
	GO term associated with UniProt-AC	3156	3156
	GO term associated with STRING-ID	3128	3140
	GO term not associated with STRING-ID	28	16
CHICK	UniProt-AC associated with GO term	698	698
	STRING-ID mapped with UniProt-AC	583	590
	STRING-ID unmapped with UniProt-AC	115	108
	GO term associated with UniProt-AC	1206	1206
	GO term associated with STRING-ID	1145	1147
	GO term not associated with STRING-ID	61	59
DANRE	UniProt-AC associated with GO term	4722	4722
	STRING-ID mapped with UniProt-AC	3813	4121
	STRING-ID unmapped with UniProt-AC	909	601
	GO term associated with UniProt-AC	3341	3341
	GO term associated with STRING-ID	3078	3167
	GO term not associated with STRING-ID	263	174
DROME	UniProt-AC associated with GO term	6028	6028
	STRING-ID mapped with UniProt-AC	5686	5887
	STRING-ID unmapped with UniProt-AC	342	141
	GO term associated with UniProt-AC	5801	5801
	GO term associated with STRING-ID	5734	5755
	GO term not associated with STRING-ID	67	46
HUMAN	UniProt-AC associated with GO term	15699	15699
	STRING-ID mapped with UniProt-AC	15265	15356
	STRING-ID unmapped with UniProt-AC	434	343
	GO term associated with UniProt-AC	11935	11935
	GO term associated with STRING-ID	11875	11895
	GO term not associated with STRING-ID	60	40
MOUSE	UniProt-AC associated with GO term	11837	11837
	STRING-ID mapped with UniProt-AC	11562	11622
	STRING-ID unmapped with UniProt-AC	275	215
	GO term associated with UniProt-AC	11875	11875
	GO term associated with STRING-ID	11824	11830
	GO term not associated with STRING-ID	51	45

Table S4 and Table S5 show the percentage of STRING-ID mapped with an UniProt-AC and the GO terms associated with a STRING-ID before and after applying the data preparation pipeline shown in Fig. S3.

Table S4. Percentage of recovered identifiers and GO terms reported in Table S2 before and after applying the data preparation pipeline shown in Fig. S3.

	Percentage of identifiers and GO terms rescued in Table S2			
		GOA db	After Enrichment	Tot. Rescued
CAEEL	UniProt-AC-STRING-ID	96.80%	98.70%	1.90%
	STRING-ID-GO terms	99.00%	99.30%	0.30%
CHICK	UniProt-AC-STRING-ID	70.40%	73.50%	3.10%
	STRING-ID-GO terms	84.60%	87.00%	2.40%
DANRE	UniProt-AC-STRING-ID	84.40%	88.40%	4.00%
	STRING-ID-GO terms	93.10%	94.40%	1.30%
DROME	UniProt-AC-STRING-ID	91.80%	97.50%	5.70%
	STRING-ID-GO terms	97.90%	98.80%	0.90%
HUMAN	UniProt-AC-STRING-ID	94.10%	96.70%	2.60%
	STRING-ID-GO terms	98.90%	99.20%	0.30%
MOUSE	UniProt-AC-STRING-ID	94.00%	97.20%	3.20%
	STRING-ID-GO terms	98.70%	99.40%	0.70%
Average across organisms	UniProt-AC-STRING-ID	88.60%	92.00%	3.40%
	STRING-ID – GO terms	95.40%	96.40%	1.00%

Table S5. Percentage of recovered identifiers and GO terms reported in Table S3 before and after applying the data preparation pipeline shown in Fig. S3.

	Percentage of identifiers and GO terms rescued in Table S3			
		GOA db	After Enrichment	Tot. Rescued
CAEEL	UniProt-AC-STRING-ID	96.90%	99.20%	2.30%
	STRING-ID-GO terms	99.10%	99.50%	0.40%
CHICK	UniProt-AC-STRING-ID	83.50%	84.50%	1.00%
	STRING-ID-GO terms	94.90%	95.10%	0.20%
DANRE	UniProt-AC-STRING-ID	80.70%	87.30%	6.60%
	STRING-ID-GO terms	92.10%	94.80%	2.70%
DROME	UniProt-AC-STRING-ID	94.30%	97.70%	3.40%
	STRING-ID-GO terms	98.80%	99.20%	0.40%
HUMAN	UniProt-AC-STRING-ID	97.20%	97.80%	0.60%
	STRING-ID-GO terms	99.50%	99.70%	0.20%
MOUSE	UniProt-AC-STRING-ID	97.70%	98.20%	0.50%
	STRING-ID-GO terms	99.60%	99.60%	0.00%
Average across organisms	UniProt-AC-STRING-ID	91.70%	94.10%	2.40%
	STRING-ID – GO terms	97.30%	98.00%	0.70%

S5 Performance metrics

We evaluated the generalization performance of the methods by considering two different scenarios, i.e. (a) *term-centric* and (b) *protein-centric* measure (Jiang *et al.*, 2016). These two types of evaluations address the following related questions: (a) what are the gene products associated with a specific functional term and (b) what are the functions of a given protein. Therefore, the term-centric evaluation is more appropriate to prioritize genes products respect to a specific function (i.e. an ontology term), whereas the protein-centric evaluation is more appropriate to predict the functions of a given protein.

1. *Term-centric metric*. For each GO term we computed the classic Aurea Under the ROC Curve (AUROC) and the Area Under the Precision Recall Curve (AUPRC) to take into account the imbalance of annotated versus unannotated GO terms. Indeed the AUPRC is more informative on unbalanced dataset than the classic AUROC (Saito and Rehmsmeier, 2015). The ROC curve is computed by plotting the *recall* (or *sensitivity*) against the *false positive rate* (or $1 - specificity$) at different thresholds, whereas the precision-recall curve shows the trade-off between precision and recall at different cutoffs. For a particular functional term i, the recall (Rc), the specificity (Sp) and the precision (Pc) at a given score threshold $\tau \in [0, 1]$ are defined as follow:

$$Rc(i, \tau) = \frac{\sum_j \mathbb{1}(y_i^j > \tau \wedge j \in A_i)}{|A_i|} \tag{13}$$

$$Sp(i, \tau) = \frac{\sum_j \mathbb{1}(y_i^j \leq \tau \wedge j \in \overline{A}_i)}{|\overline{A}_i|} \tag{14}$$

$$Pc(i, \tau) = \frac{\sum_j \mathbb{1}(y_i^j > \tau \wedge j \in A_i)}{\sum_i \mathbb{1}(i \in P_i(\tau))} \tag{15}$$

   where $y_i^j \in [0, 1]$ is the predicted score (likelihood) that a gene product j is associated with the GO term i, $P_i(\tau)$ denotes the set of gene products that have a predicted score greater than or equal to τ for a given GO term i, $\tau \in [0, 1]$ is the given threshold, A_i is the set of gene products annotated with GO term i and $\overline{A}_i$ is the set complement with regard to the whole corpus and $\mathbb{1}$ is the standard indicator function.

2. *Protein-centric metric*. Precision (Pr), Recall (Rc) and the maximum achievable F-score (F_{max}) using thresholds $\tau \in [0, 1]$ are calculated as follows:

$$Pr(i, \tau) = \frac{1}{N} \sum_{j=1}^{N} \frac{\sum_i \mathbb{1}(i \in P_j(\tau) \wedge i \in T_j)}{\sum_i \mathbb{1}(i \in P_j(\tau))} \tag{16}$$

$$Rc(i, \tau) = \frac{1}{N} \sum_{j=1}^{N} \frac{\sum_i \mathbb{1}(i \in P_j(\tau) \wedge i \in T_j)}{\sum_i \mathbb{1}(i \in T_j))} \tag{17}$$

$$F_{max} = max_\tau \left\{ \frac{2 \cdot Pr(\tau) \cdot Rc(\tau)}{Pr(\tau) + Rc(\tau)} \right\} \tag{18}$$

   where i denotes a protein function (GO term) in the gene ontology (excluding the root node), $P_j(\tau)$ denotes the set of terms that have a predicted scores greater than or equal to τ for a given protein j, T_j denotes the set of experimentally determined terms for a given gene product j, N the number of examples having at least one annotation with a GO term and $\mathbb{1}$ stands for the a standard indicator function. In other words the F_{max} measure is the maximum hierarchical F-score achievable by "a posteriori" setting the optimal decision threshold (Jiang *et al.*, 2016). We warn the reader that the hierarchical F-score as defined in eq. (18) provides an over-optimistic assessment of the hierarchical score. Indeed the threshold τ in eq. (18) is by definition "a posteriori" selected, by choosing the optimal τ^* that optimizes the $F - score$ having a set of pre-computed scores. An unbiased evaluation should embed the selection of the optimal τ within the learning process. Nevertheless, we use this measure since it is the reference gene-centric metric within the computational biology community, as witnessed by its usage in the CAFA international challenges (Radivojac *et al.*, 2013; Jiang *et al.*, 2016; Zhou *et al.*, 2019). For both measures, by averaging across proteins or terms, we can obtain an overall picture of the prediction performance of the methods.

S6 HEMDAG versus flat classifiers

S6.1 Flat classifiers

As flat methods used as base learners in our hierarchical ensembles we applied the supervised machine learning methods shown in Table S6. To implement them we used the `caret` R package (short for Classification And REgression Training) (Kuhn, 2008), since it provides an uniform interface to execute more than 250 predictive models.

Flat classifier	**Reference paper**	**Learning parameter**	**Default value**
C5.0 Decision Tree (C50)	Quinlan (1993)	mtry	2
Lasso and Elastic-Net Regularized Generalized Linear Models (glmnet)	Friedman *et al.* (2010)	alpha	1
		lambda	100
Linear Discriminant Analysis (lda)	Venables and Ripley (2002)	none	none
Logit Boost (logit)	Dettling and Bühlmann (2003)	boosting iterations	10
Multi Layer Perceptron (mlp)	Rumelhart *et al.* (1986)	hidden neurons	5
		hidden layers	1
Naive Bayes (nb)	Zhang (2004)	Laplace smoothing	0
Random Forest (rf)	Breiman (2001)	mtry	10
		trees numbers	500
		splitting rule	gini
		maximum tree depth	1
Support Vector Machine (svm)	Cortes and Vapnik (1995)	C	0.001
Support Vector Machine active-learning (svm-al)	Frasca and Cesa-Bianchi (2018)	active learning selection size	150
		budget of negatives	600
Bagged Ensemble of Decision Tree (treebag)	Breiman (1996)	bootstrap replications	25
		mtry	2
		maximum tree depth	30
Extreme Gradient Boosting (xgboost)	Chen and Guestrin (2016)	boosting iterations	15
		learning rate	0.3
		L2 Regularization	0
		L1 Regularization	0

Table S6. Default parameter setting of the flat classifiers (short form in brackets) used as base learners in our hierarchical ensemble methods.

S6.2 STRING network

Table S7. STRING network (v10.5) main characteristics.

Organism	**Proteins**	**Interactions**
CAEEL (*C. elegans*)	15,752	2,889,134
CHICK (*G. Gallus*)	14,093	3,172,417
DANRE (*D. rerio*)	23,449	13,187,568
DROME (*D. melanogaster*)	13,702	3,981,727
HUMAN (*H. sapiens*)	19,576	5,676,528
MOUSE (*M. musculus*)	21,151	6,307,021

S6.3 Proteins and GO terms

Table S8. Number of GO terms, edges and proteins, for each GO domain, that we considered in the experiments (December 2017 release).

Organism	Domain	Terms	Edges	Proteins
CAEEL	BP	1,335	2,458	2,597
	MF	186	231	1,806
	CC	221	392	1,924
CHICK	BP	246	411	330
	MF	43	53	292
	CC	58	88	318
DANRE	BP	1,182	2,088	3,351
	MF	118	153	1,000
	CC	89	145	550
DROME	BP	2,244	4,197	5,238
	MF	327	419	2,928
	CC	348	614	3,748
HUMAN	BP	3,460	6,492	8,169
	MF	760	988	11,203
	CC	541	1,001	9,412
MOUSE	BP	3,899	7,372	8,139
	MF	511	662	7,248
	CC	445	818	7,496

S6.4 Experimental set-up and data

S6.4.1 Data

Protein-Protein interactions were retrieved from the STRING – version 10.5 (Szklarczyk *et al.*, 2015). The main characteristics of the STRING networks are reported in Table S7. Having considered feature-based flat classifiers, we represented each protein as a vector of features, that we directly obtained from the rows of the STRING weighted adjacency matrix. Functional annotations cover the three GO domains BP, MF and CC. Both is_a and $part\_of$ relationships were considered to define the GO DAGs, to assure that the annotations can be safely propagated through the ontology. From the Gene Ontology Annotation (GOA) database (Barrell *et al.*, 2009) we downloaded the protein-GO term associations (December 2017 release) by extracting just the experimentally supported annotations (`EXP`, `IDA`, `IPI`, `IMP`, `IGI`, `IEP`). We used UniProtKB identifier mapping file to map the UniProtKB versus the STRING protein identifiers. The pipeline adopted to build the dataset is schematically illustrated in Fig. S3 and its source code is available at `https://github.com/AnacletoLAB/godata-pipe`. Finally, to avoid the prediction of those GO terms having too few annotations for a reliable assessment, we pruned all the GO terms having less than 10 annotations. The considered task covers more than 75K annotated proteins and about 7K unique GO terms. Table S8 shows the number of GO terms and proteins considered in the experiments.

S6.4.2 Experimental set-up

Due to the high dimensionality of the feature space, carrying out experiments using all the available protein features shown in Table S7 is impractical from a computational standpoint. Consequently, during the training phase we adopted an univariate feature selection based on the Pearson's correlation coefficient between the features and the GO term labeling vector to choose the 100 top-ranked features. This operation, in most cases, also led to an improvement in the accuracy (data not shown). The feature selection was cross-validated in an unbiased way, i.e the top-ranked features were selected during the training phase (Ambroise and McLachlan, 2002). As performance metrics we adopted *term-centric* and *protein-centric* measures. As *term-centric* measure we used both the classical Area Under the ROC Curve (AUROC) and the Area Under the Precision Recall Curve (AUPRC), which takes into account the imbalance of annotated versus unannotated GO terms (Saito and Rehmsmeier, 2015). As *protein-centric* measure we used the F_{max} measure, i.e. the maximum hierarchical F value achievable by "a posteriori" setting the optimal decision threshold (Jiang *et al.*, 2016) – see Section S5 for further details on the evaluation metrics.

S6.5 Heatmaps

Heatmaps shown in Fig. S4 and Fig. S5 are produced simply by using the equation shown in the caption of Fig 2 of the main manuscript respectively with AUROC and F_{max}.

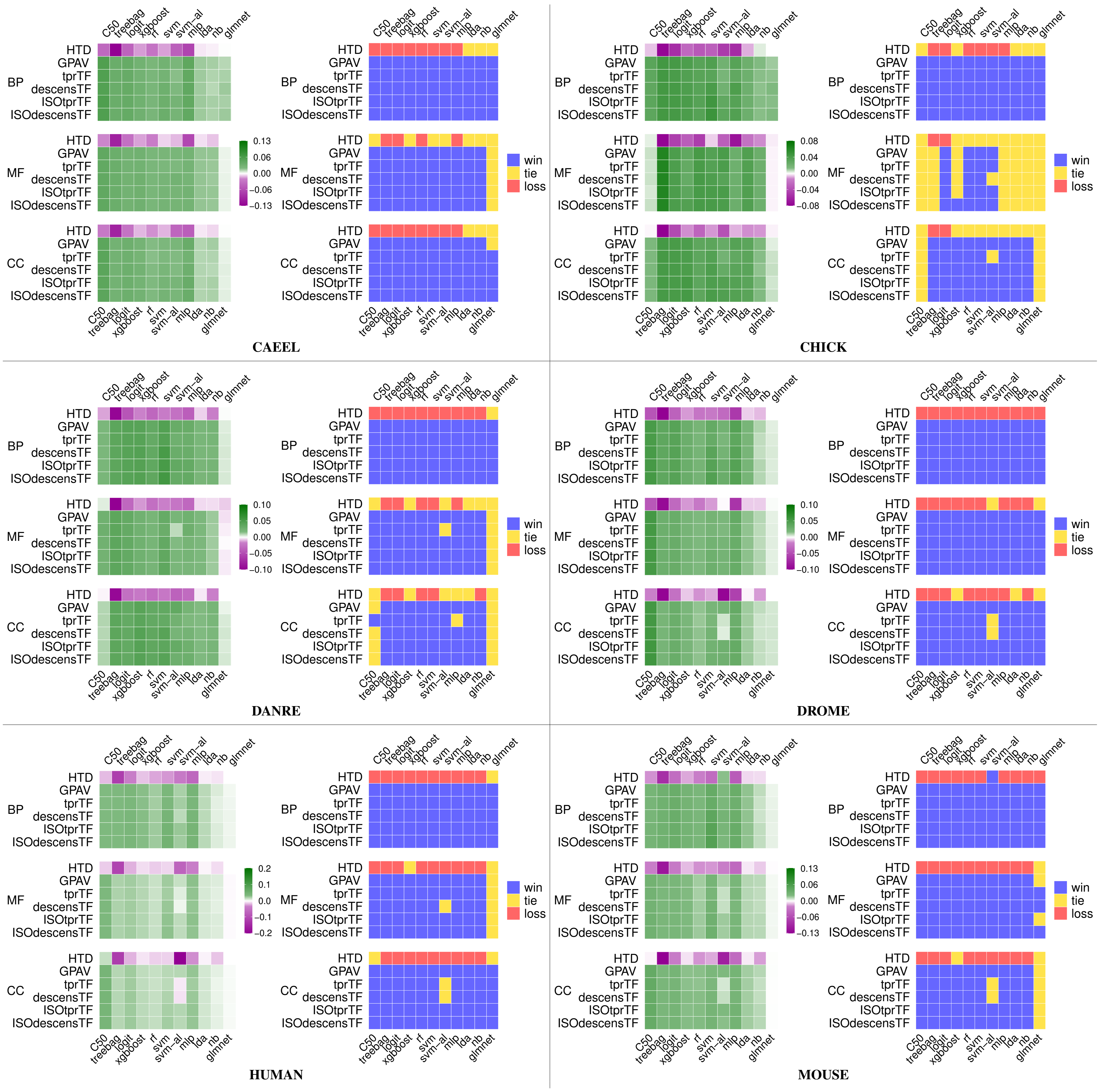


Fig. S4. Synoptic comparison of hierarchical ensemble methods against flat approaches across GO ontologies and organisms for the metric AUROC. In each panel the left heatmap refers to direct comparison of AUROC results: green color indicates better HEMDAG results; the right panel refers to the paired Wilcoxon rank sum test. Blue cells show significantly better results for HEMDAG.

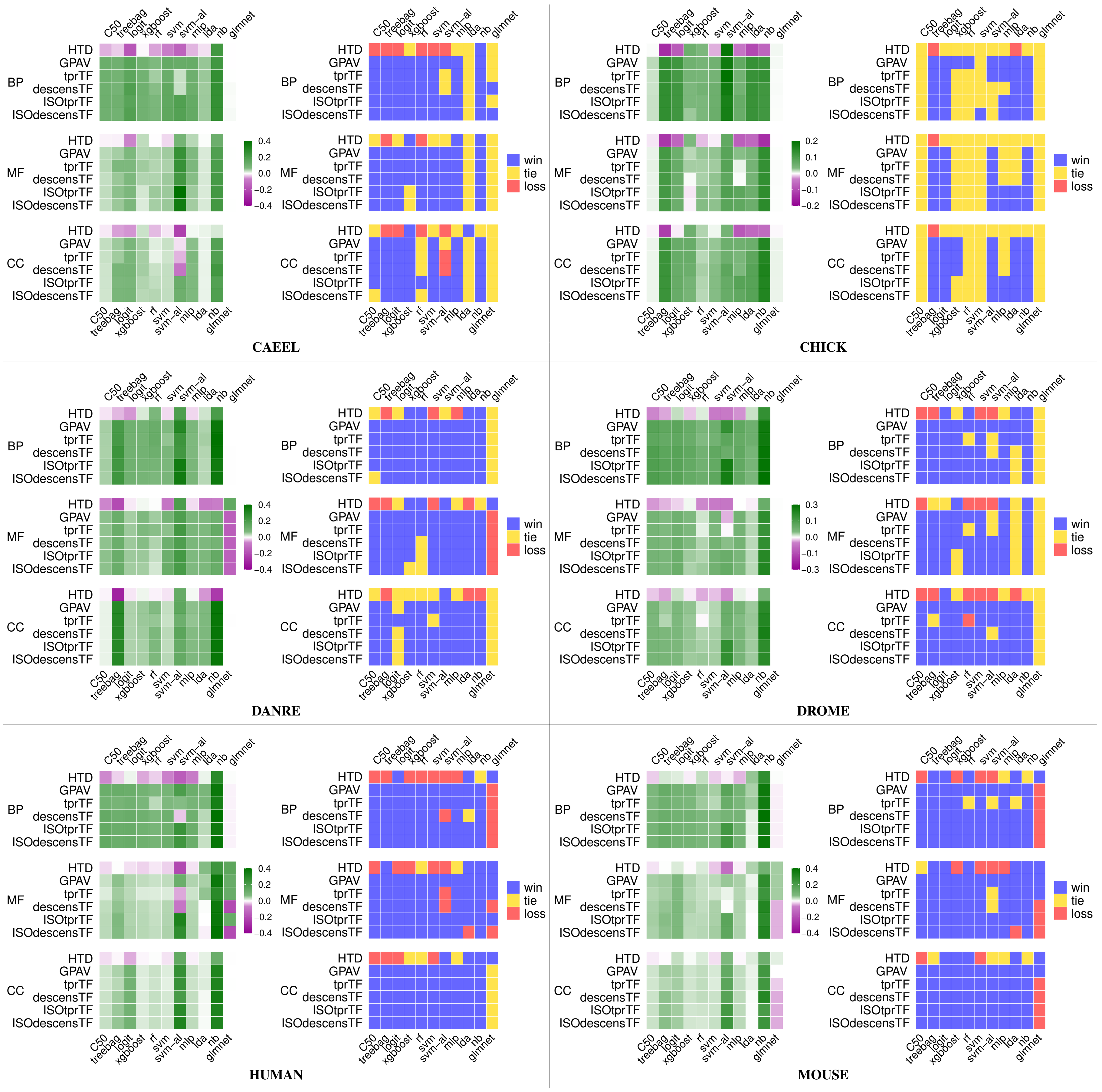


Fig. S5. Synoptic comparison of hierarchical ensemble methods against flat approaches across GO ontologies and organisms for the metric F_{max}. In each panel the left heatmap refers to direct comparison of F_{max} results: green color indicates better HEMDAG results; the right panel refers to the paired Wilcoxon rank sum test. Blue cells show significantly better results for HEMDAG.

S6.6 Win-tie-loss

Table S9 shows for each performance metric how often a hierarchical ensemble algorithm wins, ties or loses against a flat classifier across organisms and GO domains, according to the two-sided paired Wilcoxon rank sum test ($\alpha = 10^{-4}$).

Table S9. Counting of wins, ties and loses of HEMs vs flat classifiers. For each performance metric, we have in total 1188 occurrences since we considered 6 HEMs, 11 flat classifiers, 3 GO domains and 6 organisms.

	AUROC	**AUPRC**	$\mathbf{F_{max}}$	**Tot.**
win	904	858	824	2586
tie	149	226	266	641
loss	135	104	98	337
Tot.	1188	1188	1188	3564

Heatmaps depicted in Fig. S6 compare a hierarchical ensemble algorithm (row) against a flat classifier (column), by putting together the results coming from all organisms and GO domains. More precisely, the heatmaps show when the comparison of a pair is statistically significant according to a two-sided paired Wilcoxon rank sum test. In particular, we say that a hierarchical ensemble algorithm significantly "beats" (green cells) a flat classifier if the Wilcoxon test rejects the null hypothesis ($\alpha < 0.05$); conversely, we say that a hierarchical ensemble significantly "loses"(blue cells) against a flat classifier and then we say that a hierarchical ensemble method "ties" (white cells) against a flat classifier when no statistically significant difference is observed. Since we took into account six different organisms and three GO domains, a heatmap cell can range from a maximum of +18 (i.e., the hierarchical algorithm always "wins" against a flat classifier) to a minimum of -18 (i.e., the hierarchical algorithm always "loses" against a flat classifier).

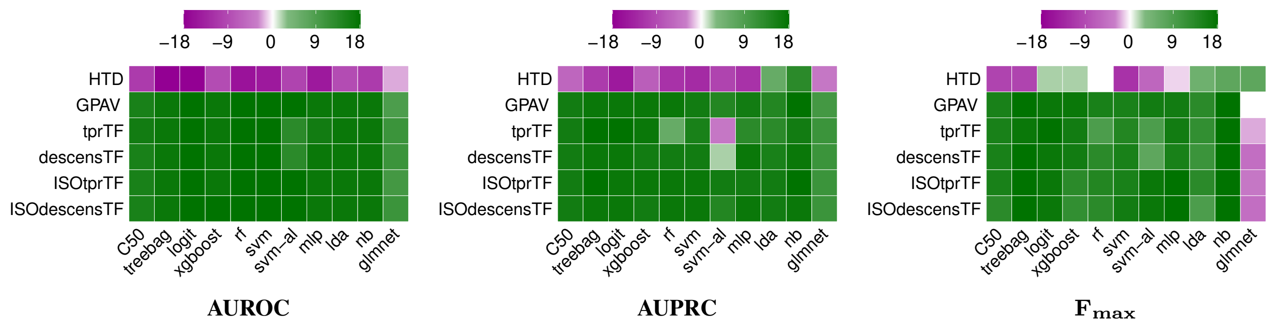


Fig. S6. Win-tie-loss heatmaps grouping the results coming from all organisms and GO domains and counting how many times an ensemble algorithm wins, ties or loses against a flat classifier, according to the two-sided paired Wilcoxon rank sum test with $\alpha < 10^{-4}$. Since we took into account six different organisms and three GO domains, a heatmap cell can range from a maximum of +18 to a minimum of -18.

S6.7 Boxplots

Box-and-whiskers plots reported below show separately for each organism taken into account in our experiments, the distributions of the AUPRC values across GO terms (having 10 or more annotations) between flat classifiers and true-path-rule ensemble methods. The star symbol ($\star$) denotes the significance according to the two-sided paired Wilcoxon sum rank test:

- p-value $< 10^{-6} \rightarrow \star\star\star$;
- p-value $< 10^{-4} \rightarrow \star\star$;
- p-value $< 10^{-2} \rightarrow \star$;
- p-value $\geq 10^{-2} \rightarrow ns$;

Observing these box-and-whiskers plots across all the organisms it easy to note that hierarchical ensemble algorithms achieve statistically significant better results than flat machine learning classifiers. These results also show that the performance of hierarchical ensembles mostly depends on that of the underlying flat base learners. This is not surprising since the improvement introduced by hierarchical methods is determined also by the capability of the base learners to provide correct and at least partially consistent predictions. Indeed when the flat classifier provides random or very noisy predictions, it is really unlikely that the hierarchical ensemble approaches are able to improve the flat predictions (see for instance the predictions returned by glmnet method). On average non linear classifiers, as random forests and gradient boosting machines work better than linear ones, and from this standpoint the box-and-whiskers plots show that these kind of classifiers lead to better results, in both the "flat" and the improved hierarchical ensemble predictions. It is likely that the performances of the flat classifiers C5.0, glmnet and naive bayes are very low, because we did not tune the learning parameter, but we used the default setting to avoid the high computational complexity due to the model selection. Moreover, it is important to mention that the improvement of hierarchical ensemble methods upon flat machine learning classifier is not due to the tuning of any hyper-parameters, since all the hierarchical algorithms used in these experiments are parameter-free. For support vector machines (svm), there are different kernels which can be employed for this classifier, such as polynomial, gaussian and radial kernel. Probably these non-linear classifiers may lead to improved results. However, we run the linear kernel since the other kernels had a too high computational complexity for the task of predicting thousands of different GO terms in different organisms.

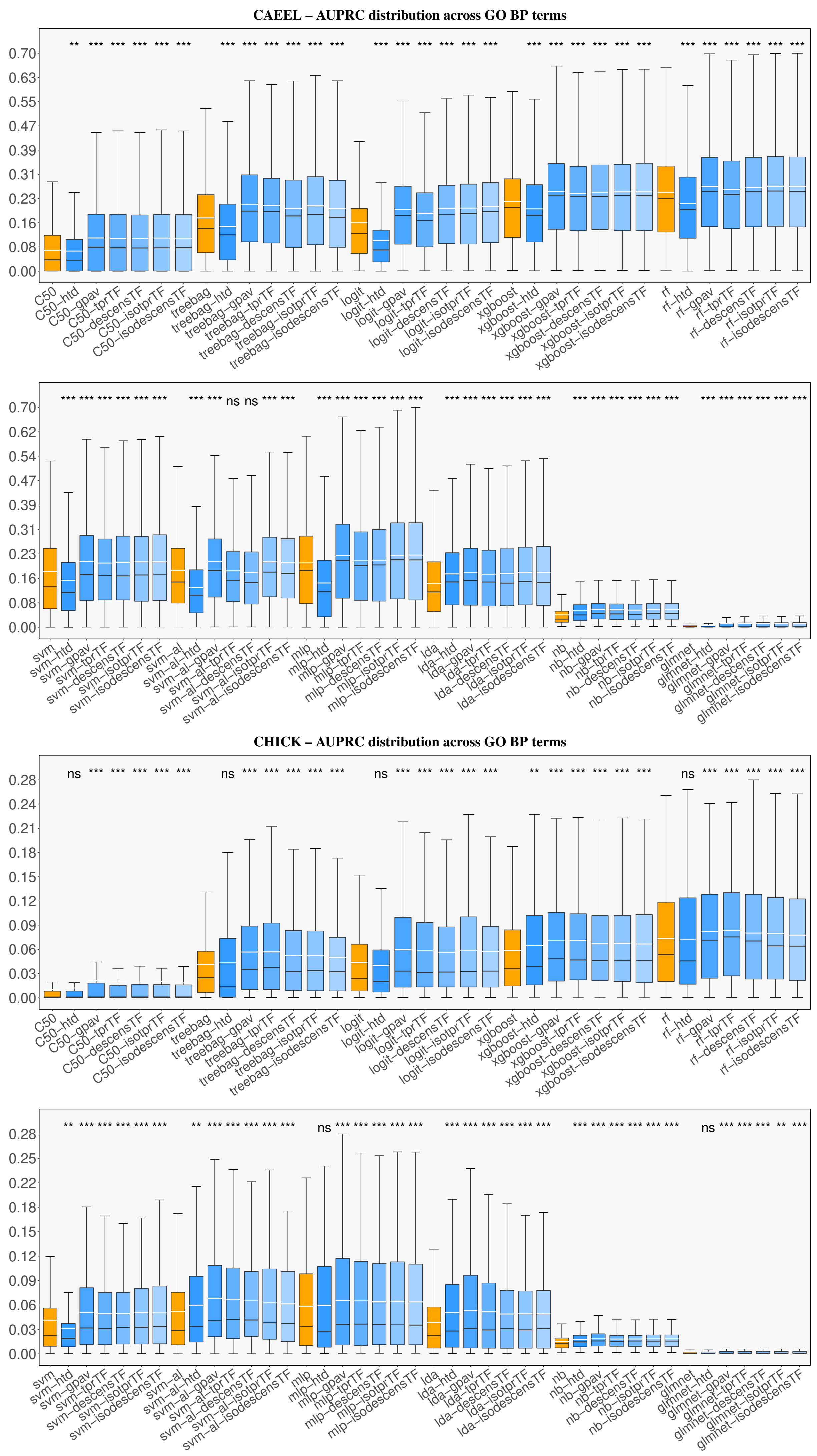


Fig. S7. Distribution of AUPRC values across the GO terms (BP ontology) having 10 or more annotation for the organisms C. elegans (CAEEL) and G. Gallus (CHICK). The black and white line in the boxplots refer respectively to the median and the mean of an method computed across GO terms.

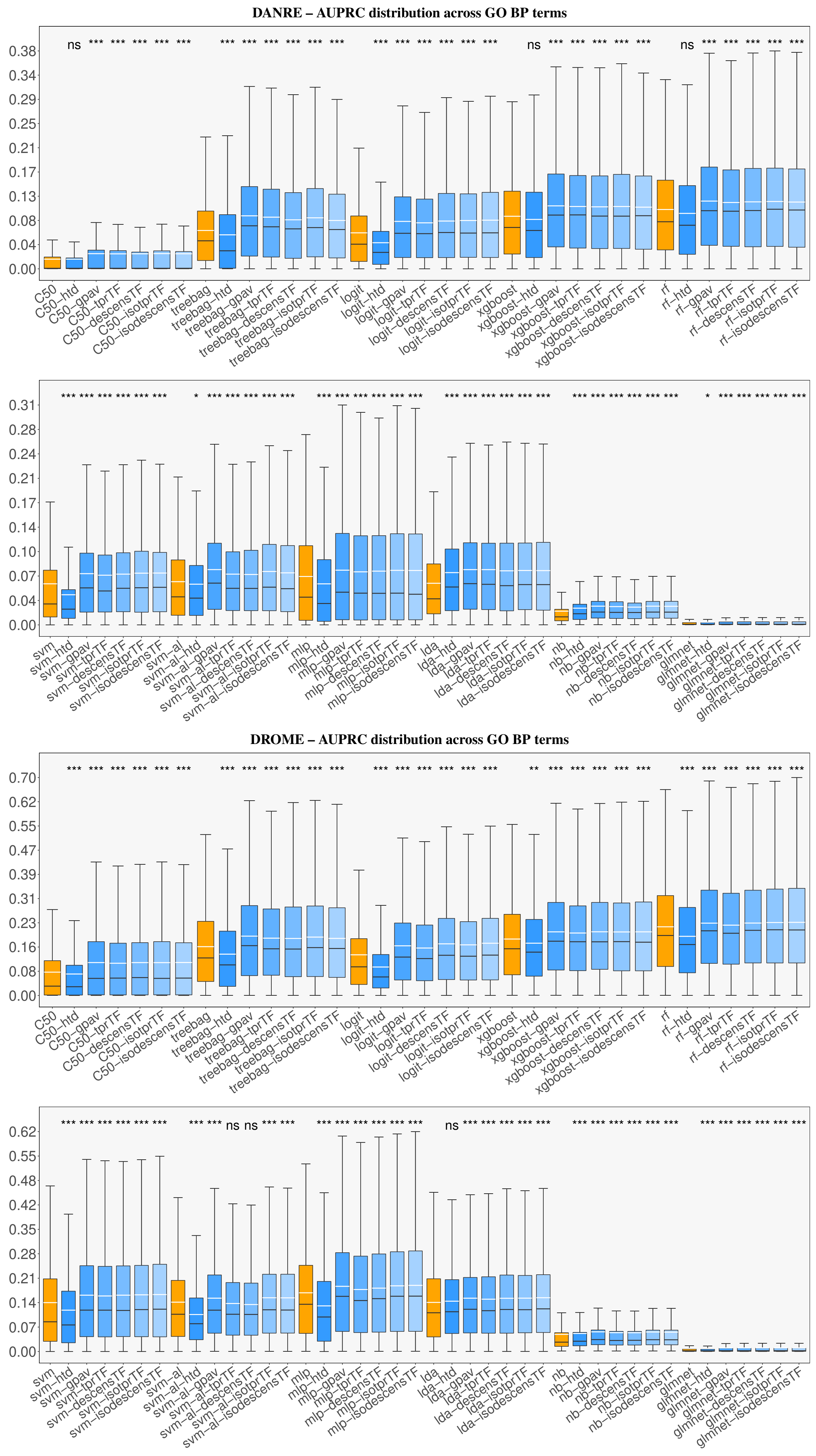


Fig. S8. Distribution of AUPRC values across the GO terms (BP ontology) having 10 or more annotation for the organisms D. rerio (DANRE) and D. melanogaster (DROME). The black and white line in the boxplots refer respectively to the median and the mean of an method computed across GO terms.

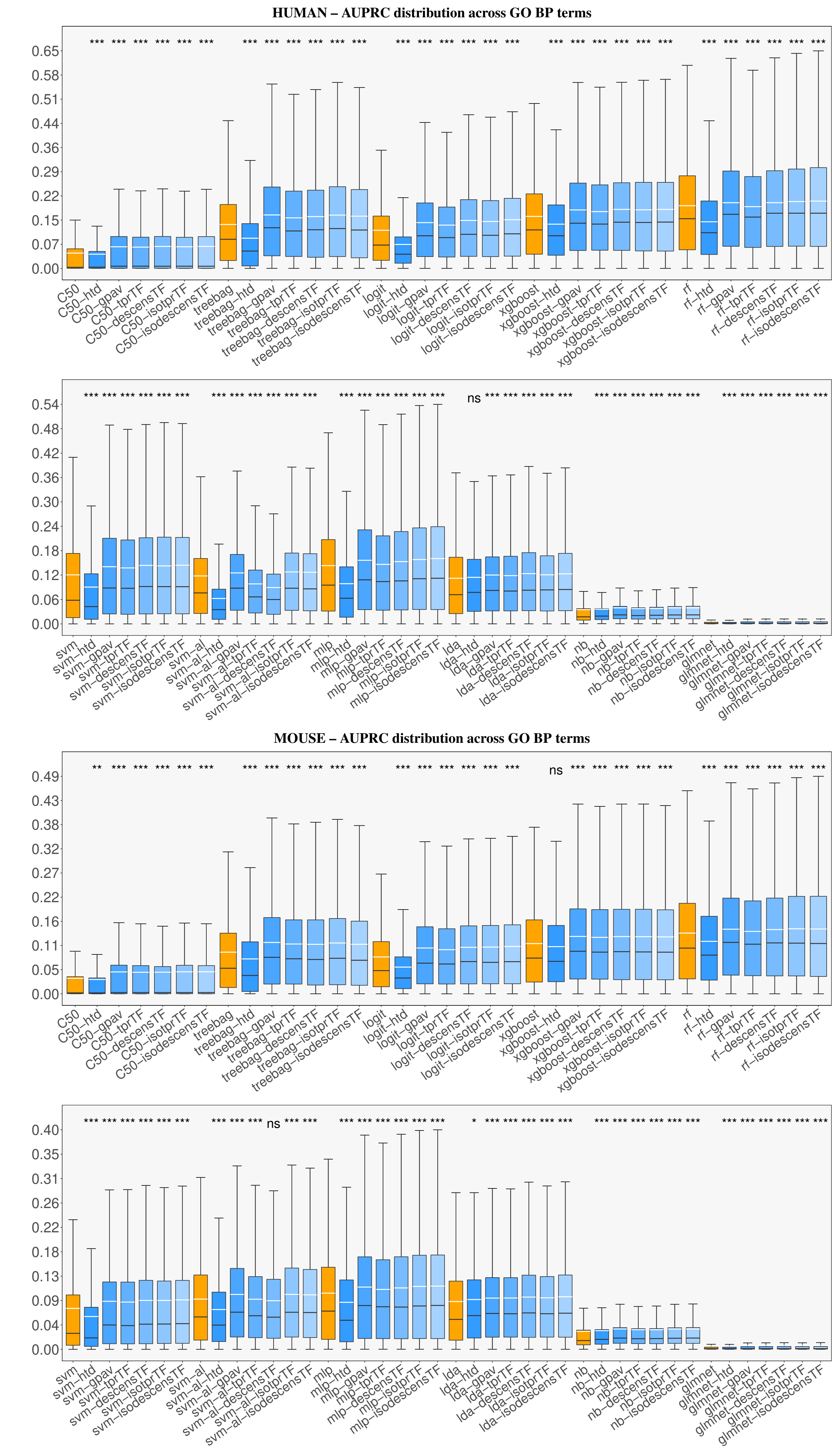
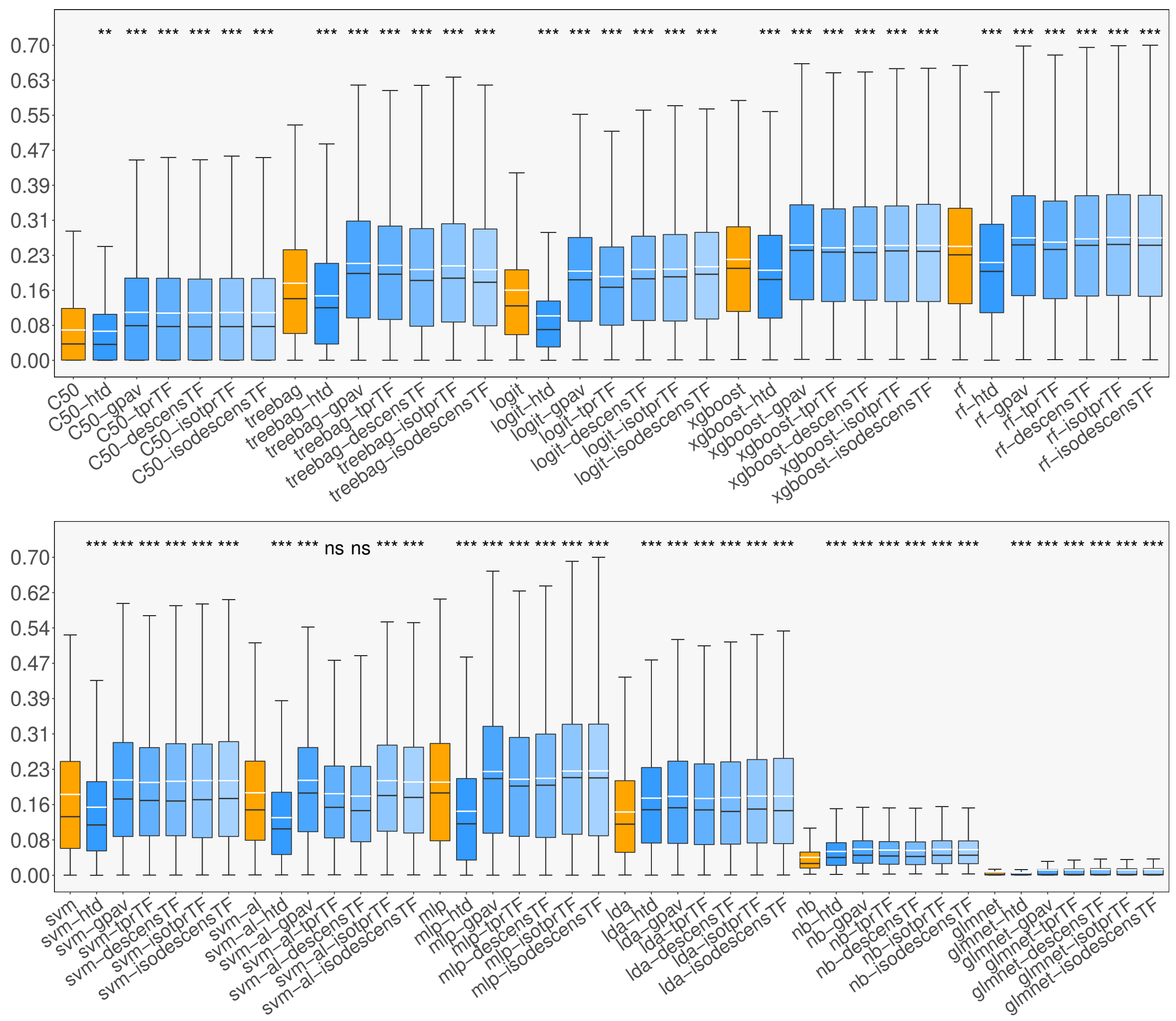


Fig. S9. Distribution of AUPRC values across the GO terms (BP ontology) having 10 or more annotation for the organisms H. sapiens (HUMAN) and M. musculus (MOUSE). The black and white line in the boxplots refer respectively to the median and the mean of an method computed across GO terms.

CAEEL – AUPRC distribution across GO MF terms



CHICK – AUPRC distribution across GO MF terms

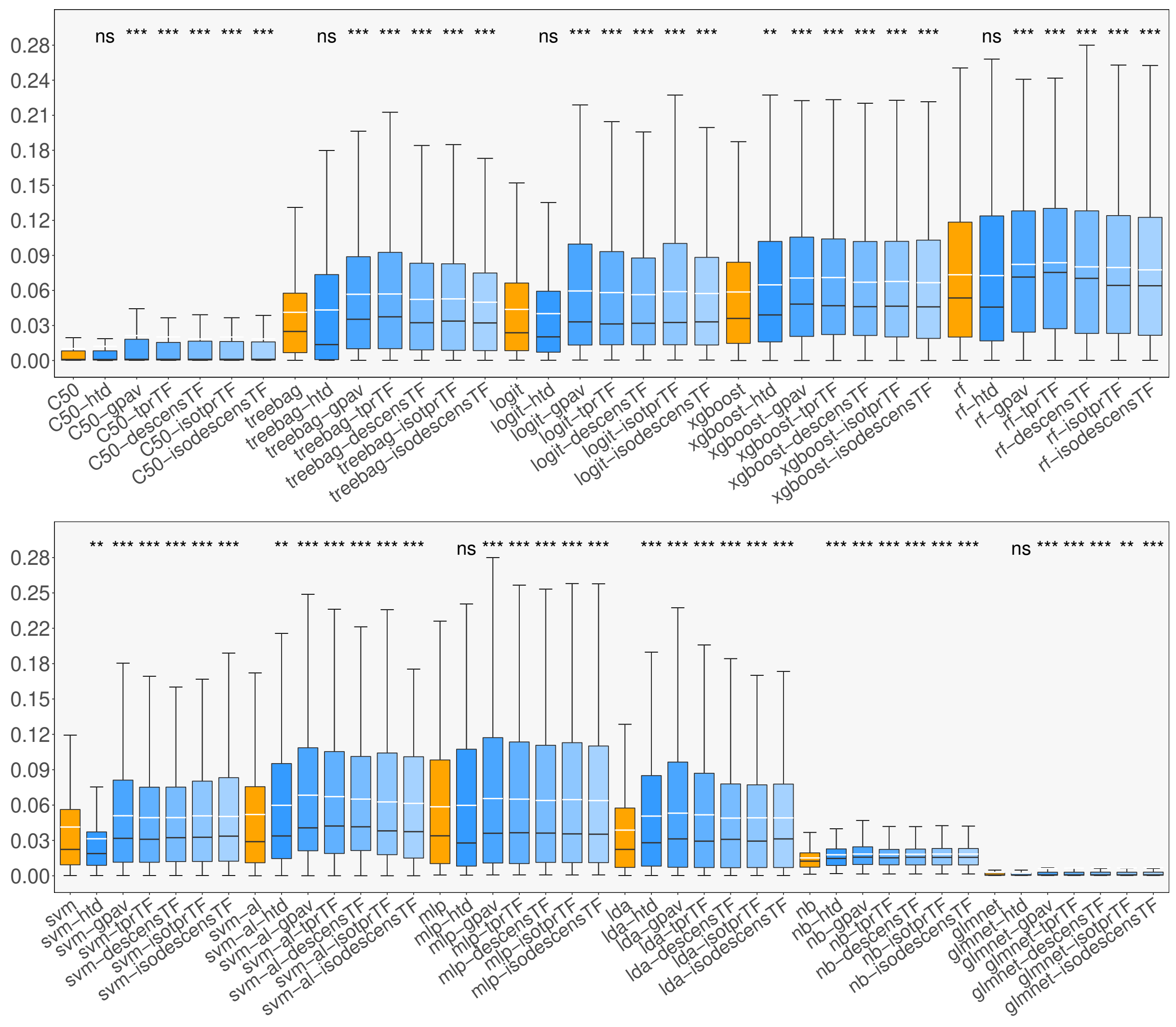


Fig. S10. Distribution of AUPRC values across the GO terms (MF ontology) having 10 or more annotation for the organisms C. elegans (CAEEL) and G. Gallus (CHICK). The black and white line in the boxplots refer respectively to the median and the mean of an method computed across GO terms.

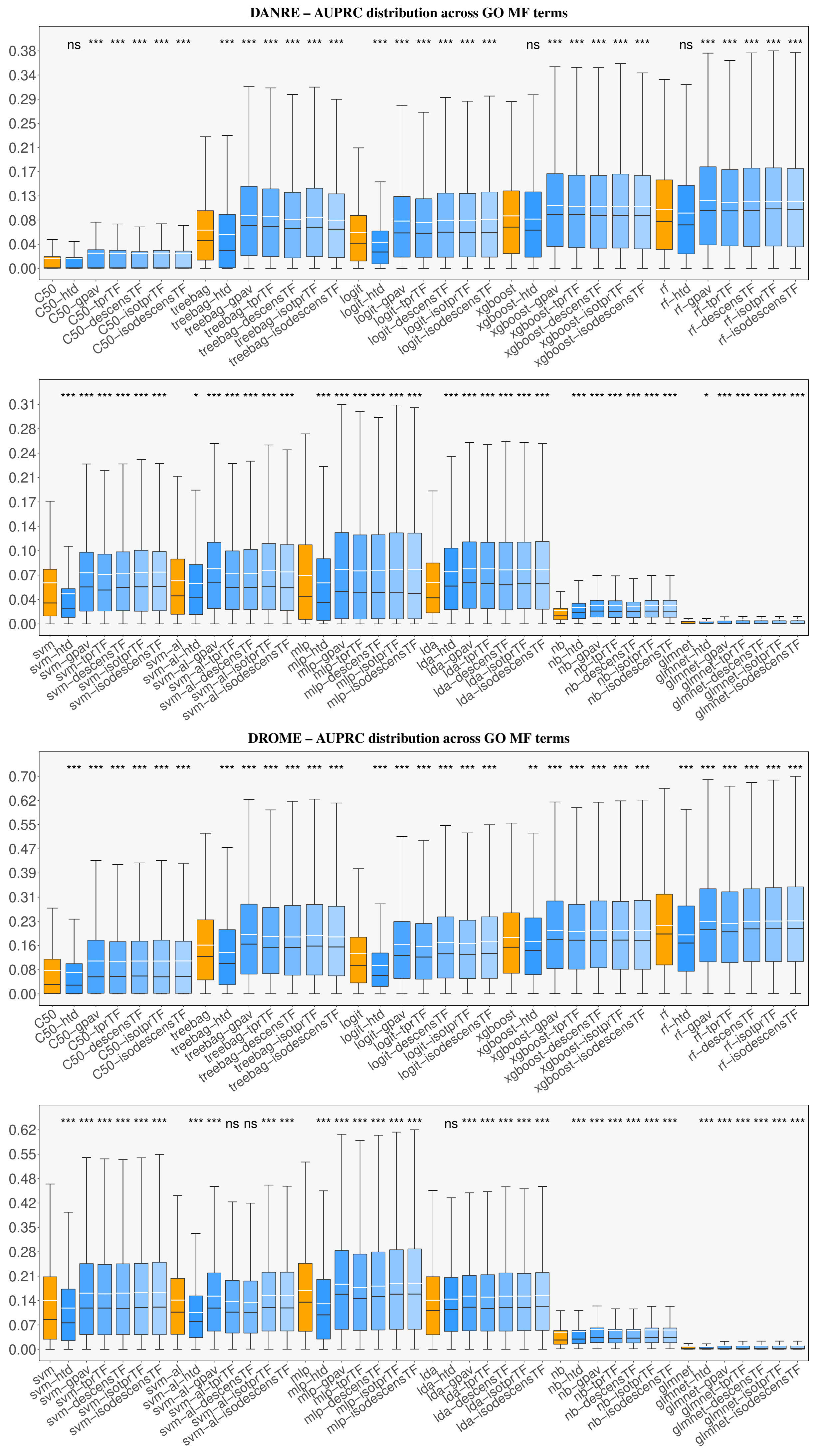


Fig. S11. Distribution of AUPRC values across the GO terms (MF ontology) having 10 or more annotation for the organisms D. rerio (DANRE) and D. melanogaster (DROME). The black and white line in the boxplots refer respectively to the median and the mean of an method computed across GO terms.

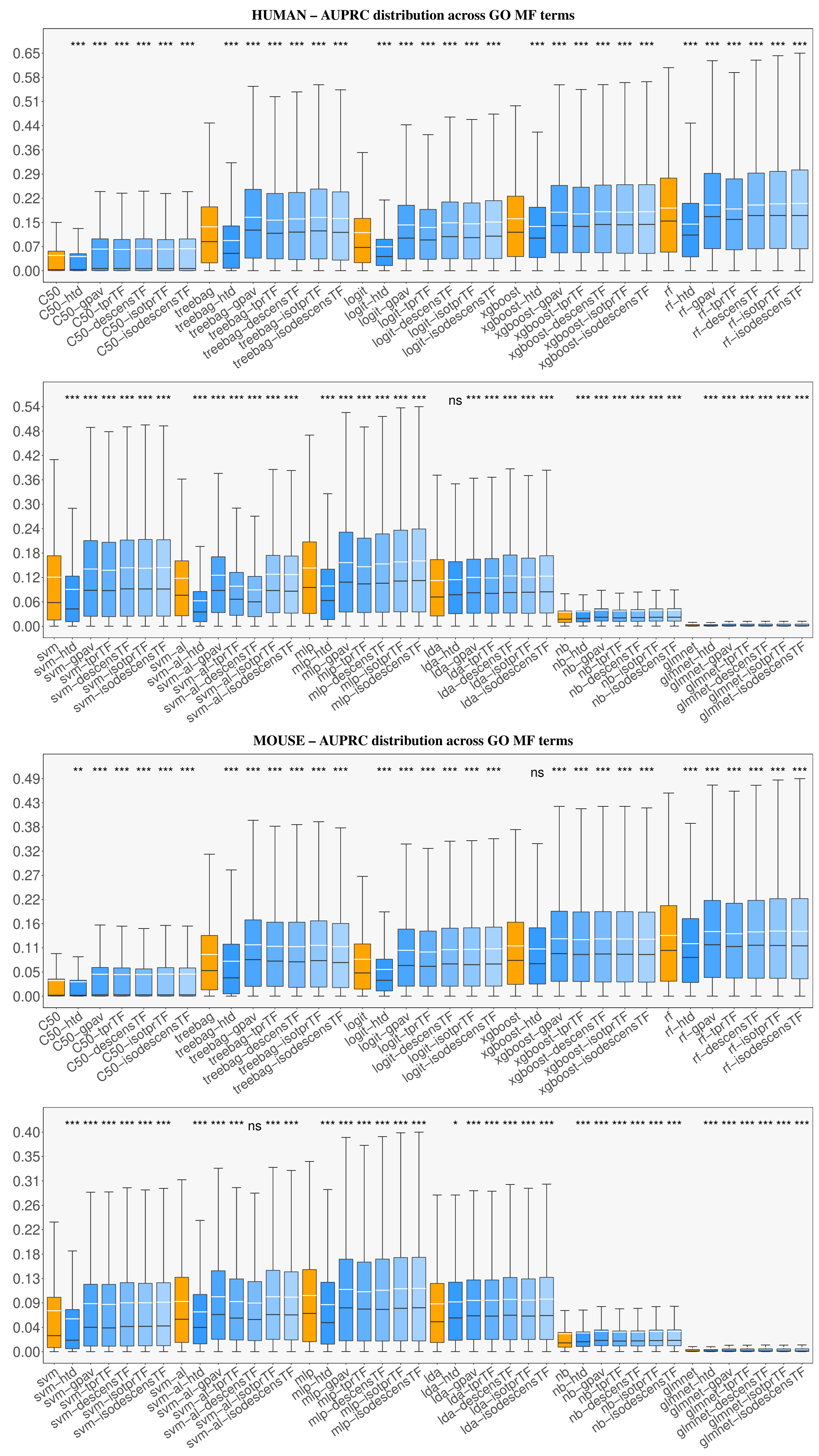
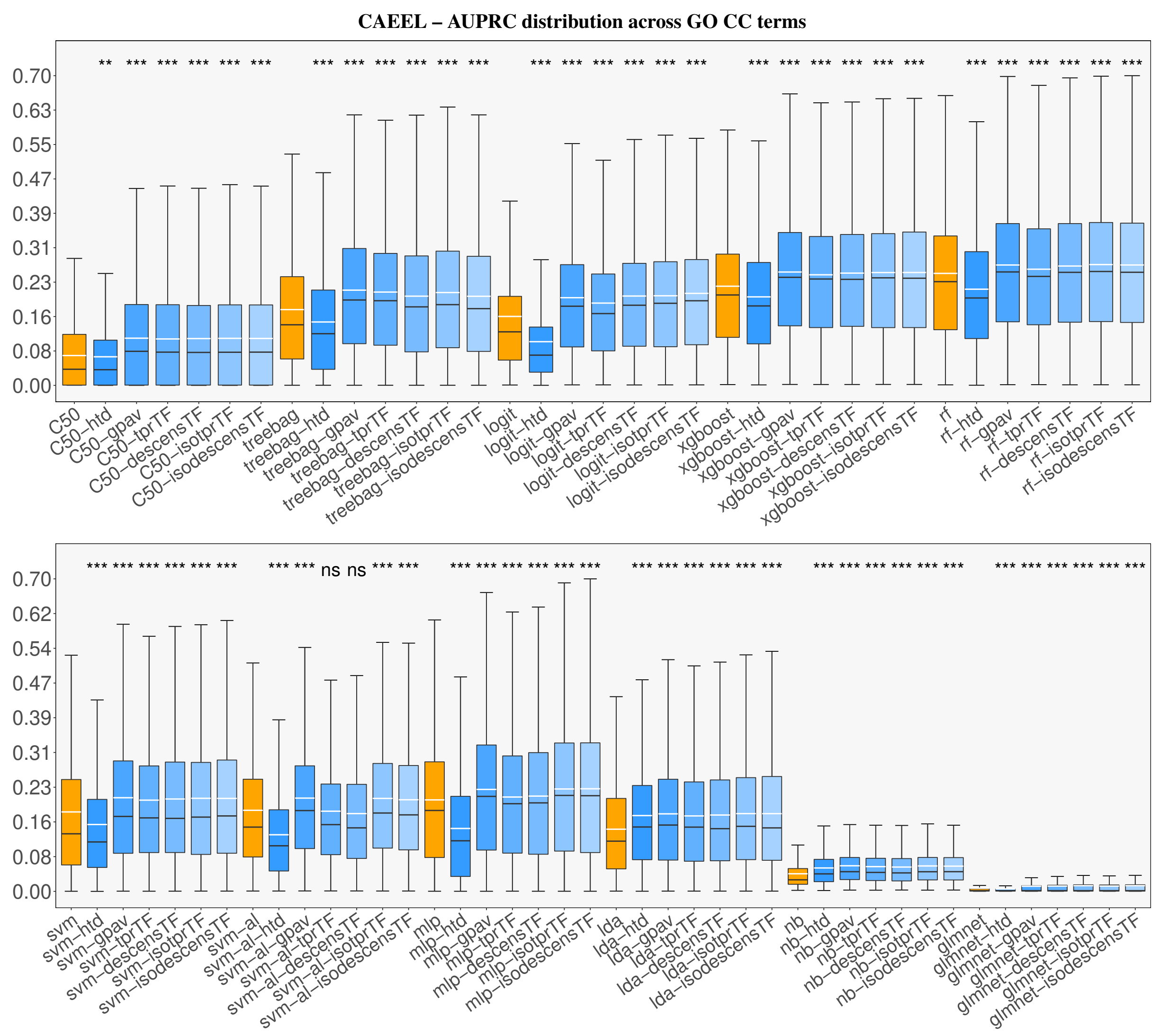


Fig. S12. Distribution of AUPRC values across the GO terms (MF ontology) having 10 or more annotation for the organisms H. sapiens (HUMAN) and M. musculus (MOUSE). The black and white line in the boxplots refer respectively to the median and the mean of an method computed across GO terms.



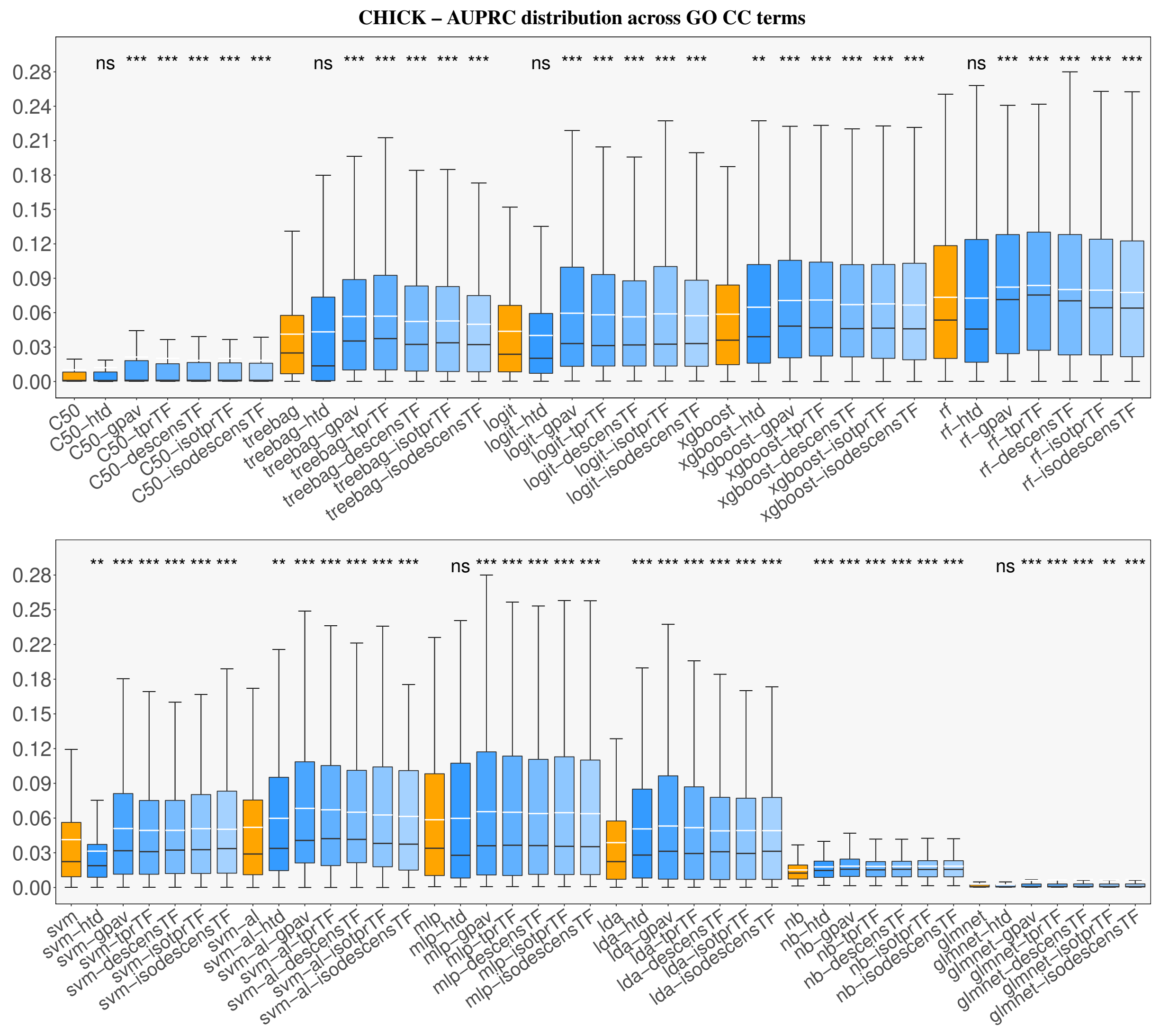


Fig. S13. Distribution of AUPRC values across the GO terms (CC ontology) having 10 or more annotation for the organisms C. elegans (CAEEL) and G. Gallus (CHICK). The black and white line in the boxplots refer respectively to the median and the mean of an method computed across GO terms.

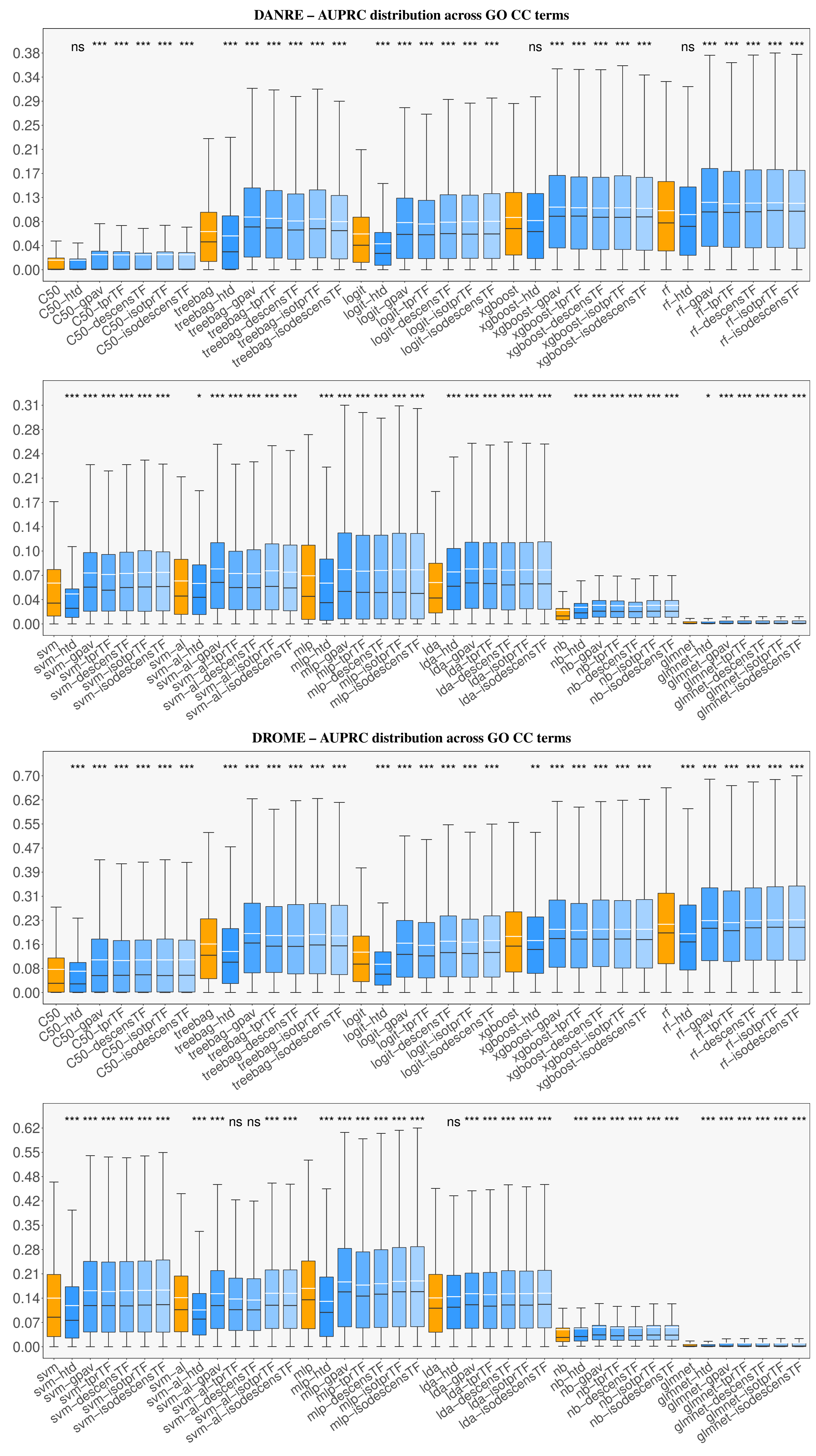


Fig. S14. Distribution of AUPRC values across the GO terms (CC ontology) having 10 or more annotation for the organisms D. rerio (DANRE) and D. melanogaster (DROME). The black and white line in the boxplots refer respectively to the median and the mean of an method computed across GO terms.

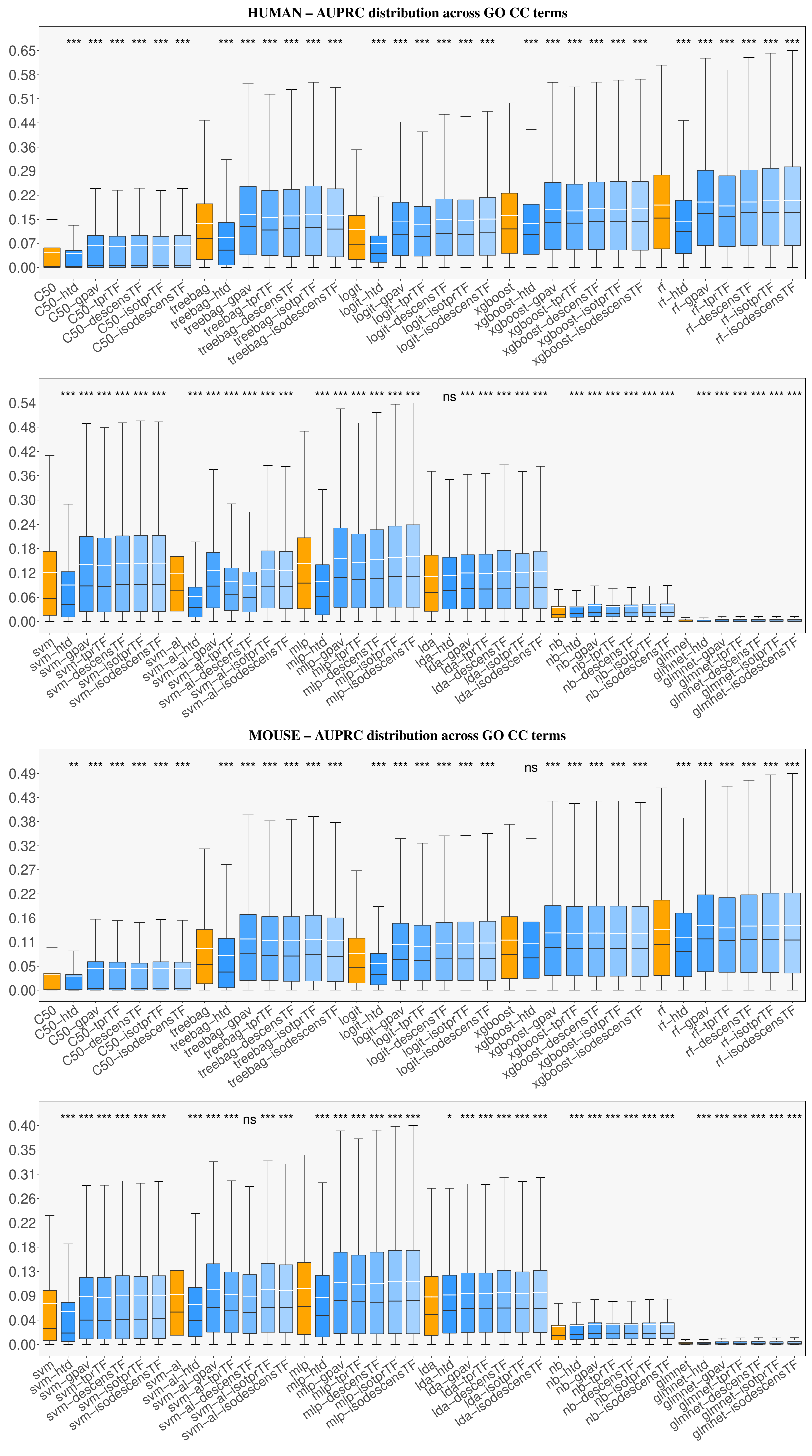


Fig. S15. Distribution of AUPRC values across the GO terms (CC ontology) having 10 or more annotation for the organisms H. sapiens (HUMAN) and M. musculus (MOUSE). The black and white line in the boxplots refer respectively to the median and the mean of an method computed across GO terms.

S6.8 Analysis of the impact of hyperparameter selection on the performance of flat classifiers and HEMDAG

Fig. S16 shows the experimental comparison between HEMDAG and the flat classifiers when hyperparameter selection is performed (by using a random search). In Table S10 we show the grid of the hyperparameters adopted for the model selection. We limited the comparison to two model organisms (*C. elegans* and *D. rerio*) using the two best flat classifiers (SVM and Random Forest, according to the results shown in the previous sections) to avoid the computational burden due to the application of random search in a double cross-validation (external + internal) experimental setting for model selection. Results shows that HEMDAG also in this setting significantly outperforms flat classifiers. Both flat classifiers and HEMDAG performance are improved by model selection: on the average with respect to the values obtained before tuning, AUROC, AUPRC and F_{max} are improved respectively of 0.9%, 5.5%, 4.9% with flat classifiers, and of 0.8%, 6.6% and 4.6% with HEMDAG. The improvements are averaged across model organisms, ontologies, flat classifiers and hierarchical ensembles.

Flat classifier	**Reference paper**	**Learning parameter**	**Parameter values**
Random Forest (rf)	Breiman (2001)	mtry	$5, 10, 20, 50$
		number of trees	$10, 100, 250, 500, 750, 1000$
		splitting rule	gini
		maximum tree depth	$1, 5, 10, 50$
Support Vector Machine (svm)	Cortes and Vapnik (1995)	C	$0.0001, 0.001, 0.01, 0.1, 0.5, 1, 10$

Table S10. Grid of the hyperparameters of the flat classifiers (short form in brackets) used as base learners in our hierarchical ensemble methods.

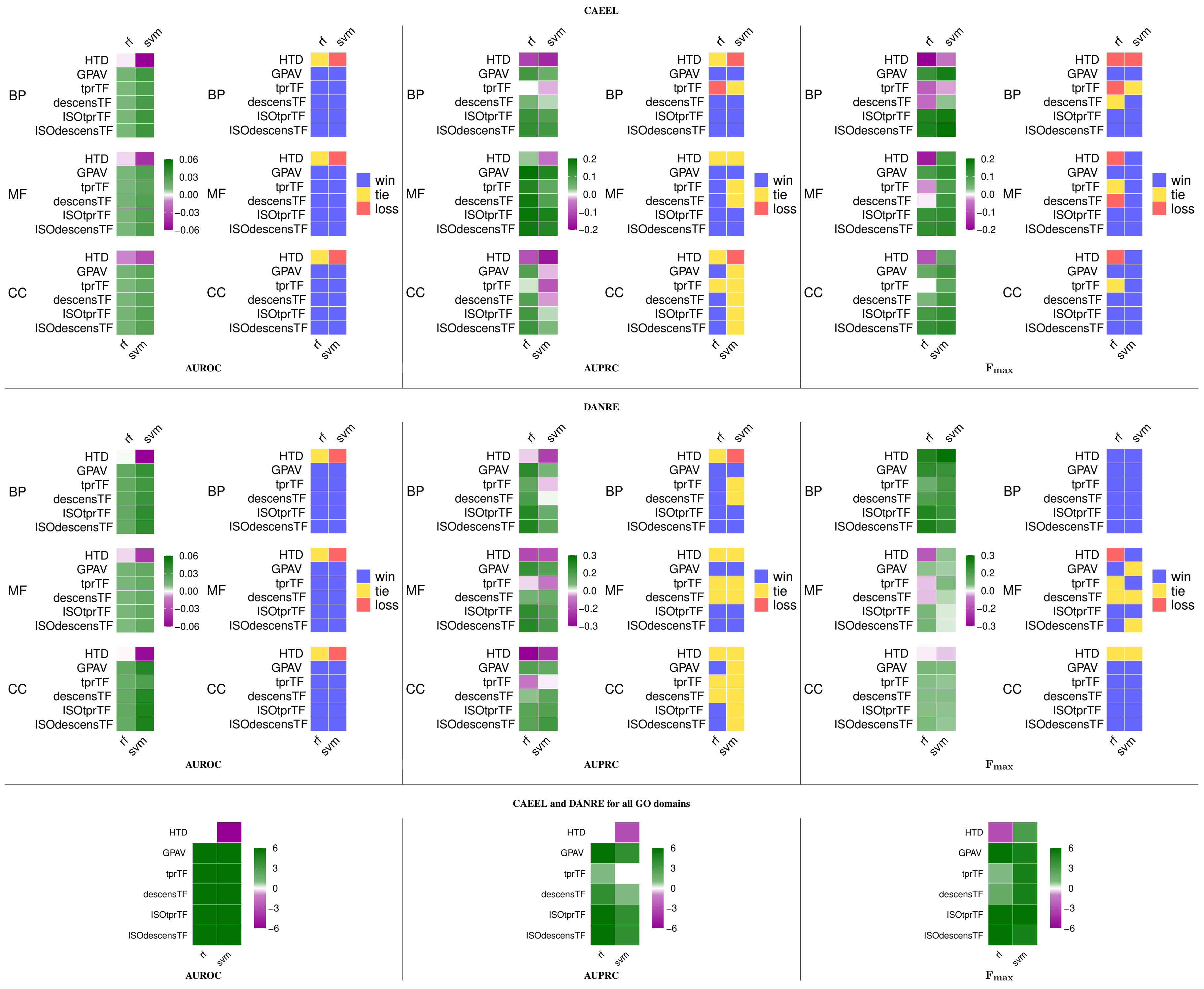


Fig. S16. Top and central rows: synoptic comparison for the metric AUROC, AUPRC and F_{max}, obtained by hierarchical ensemble methods against flat approaches across GO ontologies and C. elegans and D. rerio organisms when a random search of model parameters is performed. In each panel, the left heatmap refers to the relative difference between HEMDAG and flat classifier results: green color indicates better HEMDAG results; the right heatmap refers to win-tie-loss resuls according to the paired Wilcoxon rank sum test: blue cells show significantly better results for HEMDAG. Bottom row: win-tie-loss heatmaps grouping the results coming from the two organisms and GO domains and counting how many times an ensemble algorithm wins, ties or loses against a flat classifier, according to the two-sided paired Wilcoxon rank sum test with $\alpha < 10^{-4}$. Since we took into account two organisms and three GO domains, a heatmap cell can range from a maximum of +6 to a minimum of -6. Green color stand for better HEMDAG results.

S6.9 Experiments without IPI annotations

We performed tests with two model organisms (*C. elegans* and *D. rerio*) and the two best performing flat classifiers (SVM and Random Forest, according to the results shown in the previous section) by removing `IPI` annotations and functional interactions. The rationale behind this experiment is that GO annotations including `IPI` evidence codes could lead to increased estimates of the performance improvement achieved by HEMDAG ensembles since they are trained with the STRING PPI interaction network. In Fig. S17 we report the obtained heatmaps for AUROC, F_{max}, and AUPRC, which show that the performance improvement due to application of HEMDAG models is maintained, while the overall performance are only slightly decreased with respect to the experimental setting including IPI annotations.

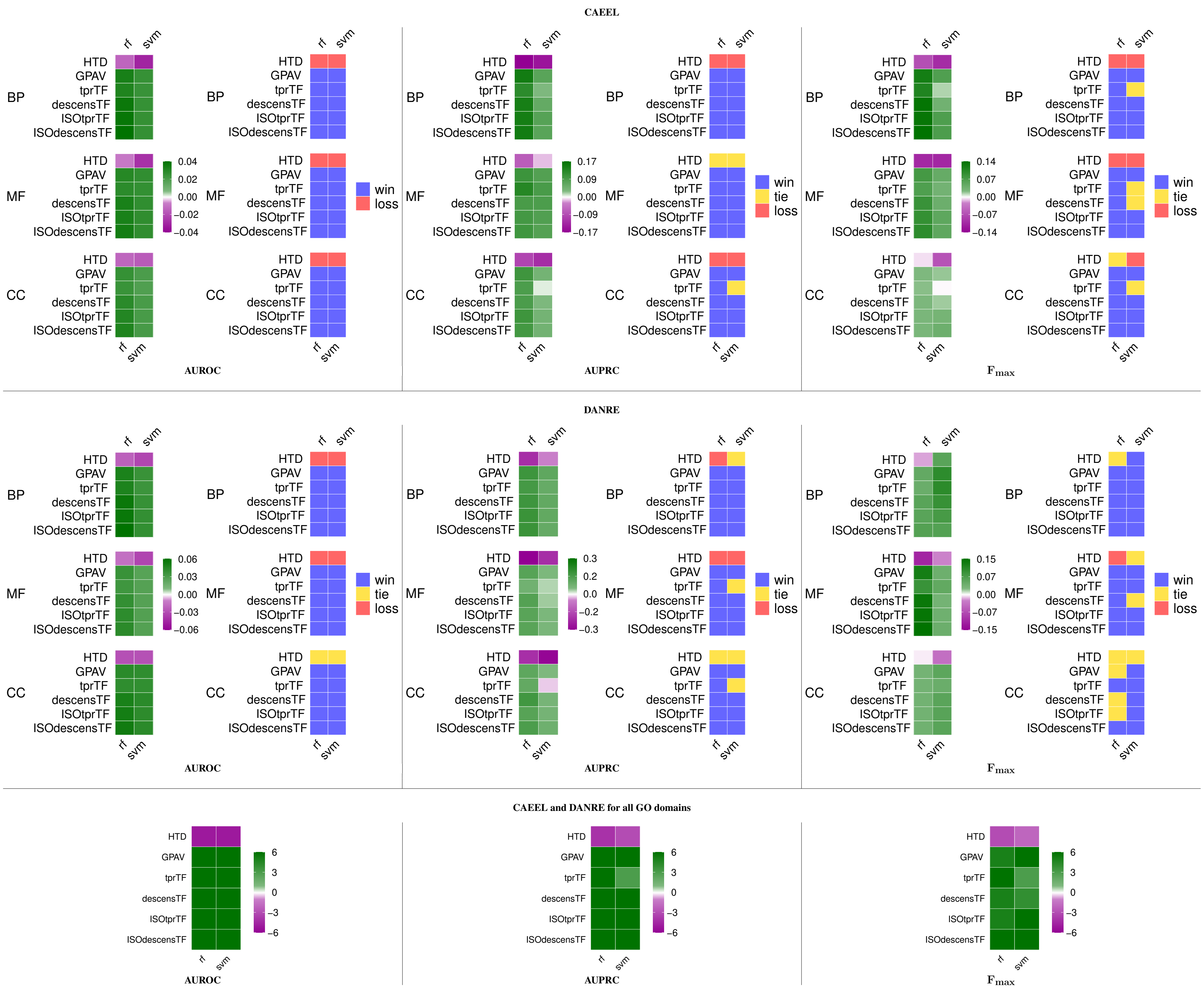


Fig. S17. Top and central rows: synoptic comparison for the metric AUROC, AUPRC and F_{max}, obtained by hierarchical ensemble methods against flat approaches across GO ontologies and C. elegans and D. rerio organisms when interactions due to `IPI` codes and STRING are removed. In each panel, the left heatmap refers to the relative difference between HEMDAG and flat classifier results: green color indicates better HEMDAG results; the right heatmap refers to win-tie-loss resuls according to the paired Wilcoxon rank sum test: blue cells show significantly better results for HEMDAG. Bottom row: win-tie-loss heatmaps grouping the results coming from the two organisms and GO domains and counting how many times an ensemble algorithm wins, ties or loses against a flat classifier, according to the two-sided paired Wilcoxon rank sum test with $\alpha < 10^{-4}$. Since we took into account two organisms and three GO domains, a heatmap cell can range from a maximum of +6 to a minimum of -6. Green color stand for better HEMDAG results.

S7 HEMDAG validation on synthetic data

S7.1 Experimental set-up.

In order not to depend on a specific organism, we randomly generated the $n \times m$ label matrix L, for $n = 500$ proteins (the number of proteins could not be increased more due to the massive set of experiments to be run). The DAG schema of the GO MF ontology (release 2020-04-23) has been adopted, for a total of $m = 4182$ terms. In order to simulate a realistic distribution of annotations, we first generated a labeling for most specific terms according to a Binomial distribution $B(n, 10^{-3})$, with 10^{-3} success probability estimated as an average from real label matrices. Then, we operated the transitive closure of annotations to respect the true-path-rule, obtaining a final label matrix having a minimum of 1 to a maximum of 500 positive proteins for a given GO term. 500 positives are only for the root of ontology, which means all proteins are positive for this term; for this reason, we excluded it from the computation of the average performance. Given a $n \times m$ (flat) prediction matrix $M \in [0, 1]^{n \times m}$, let AUPRC$(M, L)$ be its AUPRC averaged across GO terms (see Section S5) with regard to labels L, and $\overline{M}$ be the matrix obtained from M by applying a HEMDAG algorithm. The aim is to estimate the probability $p = P(\text{AUPRC}(\overline{M}, L) > \text{AUPRC}(M, L))$. To this end, T random matrices $M^{(1)}, \ldots, M^{(T)}$ were randomly generated and the random variable $Y = \sum_{i=1}^{T} H(\text{AUPRC}(\overline{M}^{(i)}, L) - \text{AUPRC}(M, L))$ evaluated, where H is the Heaviside function. The variable $\frac{Y}{T}$ is an estimator of p.

S7.2 Estimating confidence interval for p

To determine a reliable confidence interval of a binomial proportion it is required both $Tp, T(1-p) \geq 5$ (or 10) (Brown *et al.*, 2001). In order to define the confidence interval of p at $1 - \delta = 0.95$ confidence level, three cases are distinguished:

1. $Y = 0$. The confidence interval is $[0, 1 - \delta^{\frac{1}{T}}]$ (Clopper and Pearson, 1934; Louis, 1981);
2. $1 \leq Y \leq 5$. Y is approximately distributed according to the Poisson distribution with expected value $\lambda = Y$. The confidence interval is thereby $\left[\frac{1}{2T}\chi^2_{2Y,1-\frac{\delta}{2}}, \frac{1}{2T}\chi^2_{2(Y+1),\frac{\delta}{2}}\right]$, where χ^2_k is a chi squared random variable with k degrees of freedom;
3. $5 < Y < T - 5$. Y is approximately distributed according to a normal distribution with expected value Y and variance $Y(1 - \frac{Y}{T})$. The Agresti-Coull interval estimator is adopted (Agresti and Coull, 1998), which is more stable for values of Y closer to the outliers (Brown *et al.*, 2001). Thus, the confidence interval is $\frac{Y+2}{T+4} \pm \frac{1}{T+4}\sqrt{(Y+2)(T-Y-2)}z_{1-\frac{\delta}{2}}$, where $z_{1-\alpha}$ is the $1 - \alpha$ percentile of the standard normal distribution.

In the cases symmetric to case 1 ($Y = T$) and to case 2 ($T - 5 \leq Y < T$), we operate analogously on the random variable $T - Y$.

The obtained confidence intervals for p are shown in Table S11. For the methods having hyperparameters ($\bar{t}$ for tprT, ISOtprT, descensT, ISOdescensT, w for tprW, ISOtprW, descensW, ISOdescensW, $\bar{t}$ and w for tprWT, ISOtprWT, desensWT, ISOdescensWT, τ for descensTAU and ISOdescensTAU), the hyperparameters have been selected through an internal cross validation on a grid of 5 evenly spaced values in the interval $0.1, \ldots, 0.9$. The computational burden of the whole evaluation did not allow to use more refined hyperparameter grids.

Table S11. Confidence interval for the probability p that the ensemble algorithms of the HEMDAG family improve flat predictions. Best results in boldface.

Method	**Unif**	**Leaves**	**Root**	**Mixed**
HTD	[0.489, 0.551]	[0.085, 0.123]	[0.000, 0.003]	[0.282, 0.340]
GPAV	**[0.469, 0.531]**	**[0.997, 1.000]**	**[0.997, 1.000]**	**[0.679, 0.736]**
tprTF	[0.470, 0.532]	[0.876, 0.913]	[0.997, 1.000]	[0.481, 0.543]
ISOtprTF	**[0.462, 0.524]**	**[0.997, 1.000]**	**[0.997, 1.000]**	**[0.681, 0.737]**
descensTF	[0.474, 0.536]	[0.673, 0.730]	[0.002, 0.012]	[0.481, 0.543]
ISOdescensTF	**[0.465, 0.527]**	**[0.997, 1.000]**	**[0.997, 1.000]**	**[0.681, 0.737]**
tprT	[0.466, 0.528]	[0.000, 0.003]	[0.000, 0.003]	[0.178, 0.228]
tprW	[0.471, 0.533]	[0.802, 0.849]	[0.997, 1.000]	[0.481, 0.543]
tprWT	[0.507, 0.569]	[0.840, 0.883]	[0.913, 0.944]	[0.368, 0.429]
ISOtprT	[0.477, 0.539]	[0.993, 1.000]	[0.997, 1.000]	[0.569, 0.630]
ISOtprW	**[0.488, 0.550]**	**[0.997, 1.000]**	**[0.997, 1.000]**	**[0.681, 0.737]**
ISOtprWT	[0.459, 0.521]	[0.994, 1.000]	[0.943, 0.968]	[0.571, 0.632]
descensT	[0.472, 0.534]	[0.000, 0.003]	[0.000, 0.003]	[0.176, 0.226]
descensW	[0.466, 0.528]	[0.599, 0.660]	[0.001, 0.009]	[0.481, 0.543]
desensWT	[0.506, 0.568]	[0.831, 0.875]	[0.000, 0.003]	[0.370, 0.431]
descensTAU	[0.462, 0.524]	[0.651, 0.709]	[0.988, 0.996]	[0.479, 0.541]
ISOdescensT	[0.478, 0.540]	[0.919, 0.949]	[0.997, 1.000]	[0.569, 0.630]
ISOdescensW	**[0.481, 0.543]**	**[0.997, 1.000]**	**[0.997, 1.000]**	**[0.681, 0.737]**
ISOdescensWT	[0.472, 0.534]	[0.997, 1.000]	[0.997, 1.000]	[0.569, 0.630]
ISOdescensTAU	**[0.476, 0.538]**	**[0.997, 1.000]**	**[0.997, 1.000]**	**[0.683, 0.739]**

S8 HEMDAG versus structured output SOTA methods

S8.1 STRING network

Table S12. STRING network (v11) main characteristics.

Organism	Proteins	Interactions
CAEEL (*C. elegans*)	18,181	3,709,383
CHICK (*G. Gallus*)	14,683	3,063,715
DANRE (*D. rerio*)	24,681	7,058,228
DROME (*D. melanogaster*)	13,046	1,981,482
HUMAN (*H. sapiens*)	19,354	5,879,727
MOUSE (*M. musculus*)	21,291	5,972,403

S8.2 Proteins and GO terms

Table S13. Number of GO terms, edges and proteins for each GO domain that we considered in the time-lapse experiments (2017-2020).

Organism	Domain	Terms	Edges	Proteins	Training Proteins	Test Proteins
CAEEL	BP	1,332	2,458	2,828	2,600	228
	MF	179	216	1,862	1,800	62
	CC	192	315	2,195	1,919	276
CHICK	BP	285	474	401	320	81
	MF	44	49	345	274	71
	CC	53	75	328	264	64
DANRE	BP	1,277	2,260	4,011	3,348	663
	MF	128	155	1,138	997	141
	CC	77	117	630	551	79
DROME	BP	2,265	4,250	5,342	5,192	150
	MF	314	395	3,002	2,917	85
	CC	317	508	3,879	3,721	158
HUMAN	BP	3,522	6,584	8,698	8,119	579
	MF	762	967	13,081	11,126	1,955
	CC	513	866	10,378	9,351	1,027
MOUSE	BP	3,967	7,491	8,655	8,117	538
	MF	497	630	7,561	7,219	342
	CC	423	708	7,953	7,472	481

S8.3 Data and experimental set-up

S8.3.1 Data

As protein-protein interaction network, we used the most recent release of STRING – v11 (Szklarczyk *et al.*, 2018). The topological characteristics of the STRING networks (v11.0) are reported in Table S12. We downloaded the protein-GO term associations (June 2020 release) from GOA database, by extracting only the annotations supported by experimental evidence codes (`EXP`, `IDA`, `IPI`, `IMP`, `IGI`, `IEP`, `HTP HDA HMP HGI HEP`) and by pruning all the GO terms having less than 10 annotations. In Table S13 we report the statistics of the data used in the time lapse experiments. We used the annotations of an old GO release (December 2017) to predict the protein functions of a more recent GO release (June 2020). Since among different GO releases some functional terms can change, become obsolete or also removed, we mapped the old GO terms to the new ones by parsing the annotation file of December 2017 GO release by using as key the alt-ID taken from the obo file of June 2020 GO release. To map ontology terms between releases we used the in-house developed Perl5 module `obogaf::parser`, which is publicly available both on `CPAN` and `Bioconda` repository (`https://anaconda.org/bioconda/perl-obogaf-parser`) alongside a comprehensive tutorial (`https://obogaf-parser.readthedocs.io`). The source code used to build the datasets is available at `https://github.com/AnacletoLAB/godata-pipe`.

S8.3.2 Experimental set-up

Let us denote by T the set of proteins having at least 1 annotation with a GO term of the "old" GO release (2017), and by S the set of newly annotated proteins, i.e. gene products having at least one new annotation in the "new" GO release (2020), but unannotated in the December 2017 GO release. Hence we have that $S \cap T = \emptyset$. We used the set T as training set and the set S as hold-out test set to assess the capability of predicting newly annotated proteins using only the annotations of the previous GO release. We used as positives all the proteins in T annotated with a GO term and as negatives all the remaining proteins in T. To evaluate the performance of the SOTA methods, we executed the DeepGOPlus Python code as published in Kulmanov and Hoehndorf (2020) and we adapted the GOstruct C++ source code to a time-lapse hold-out procedure. As evaluation metrics we used the Area Under the ROC Curve (AUROC), Area Under the Precision Recall Curve (AUPRC) and the F_{max} – see Section S5 for more details.

S8.4 Heatmaps

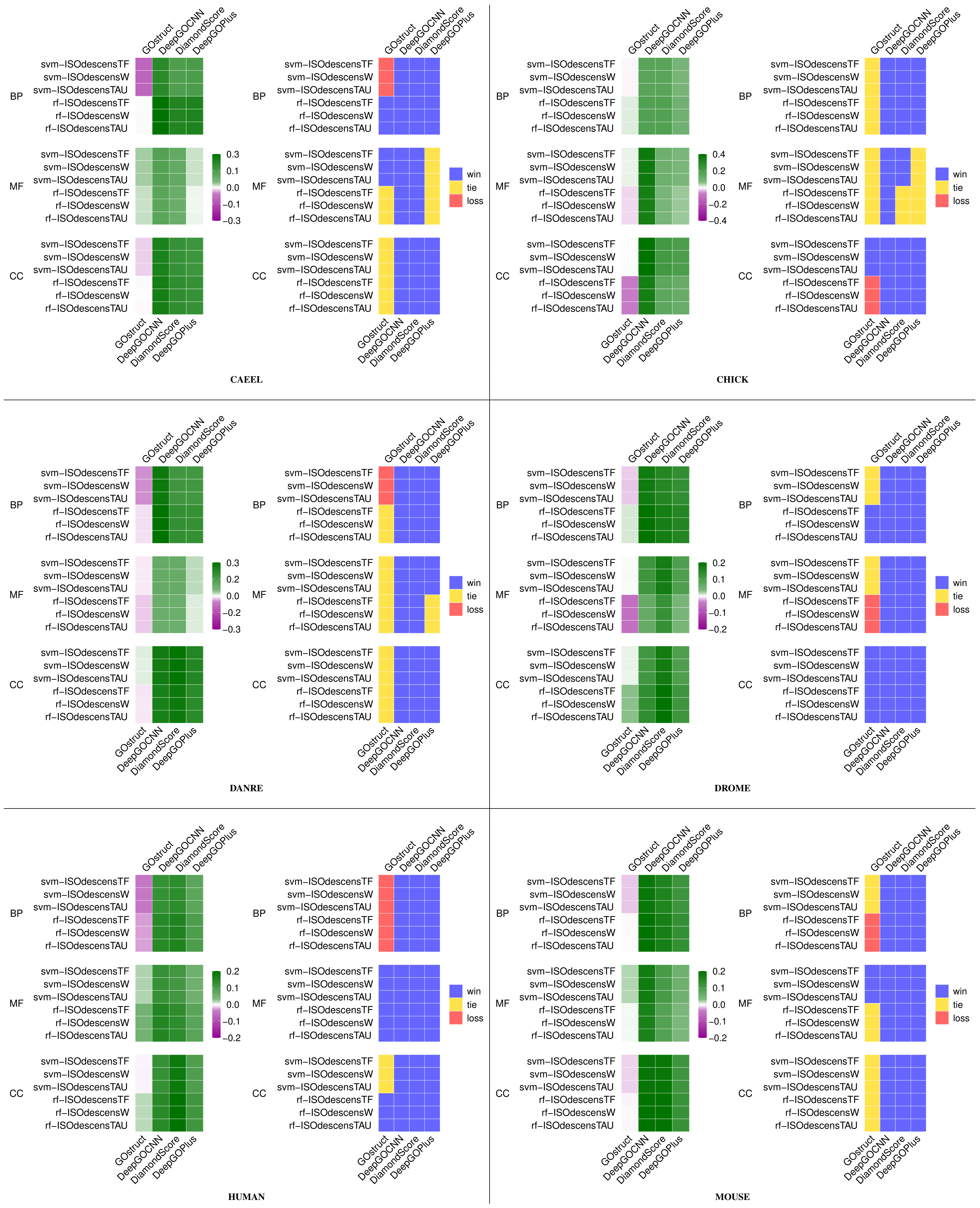


Fig. S18. Synoptic comparison of hierarchical ensemble methods vs structured output SOTA methods across GO ontologies and organisms for the metric AUROC. In each panel the left heatmap refers to direct comparison of AUROC results: green color indicates better HEMDAG results; the right panel refers to the paired Wilcoxon rank sum test. Blue cells show significantly better results for HEMDAG.

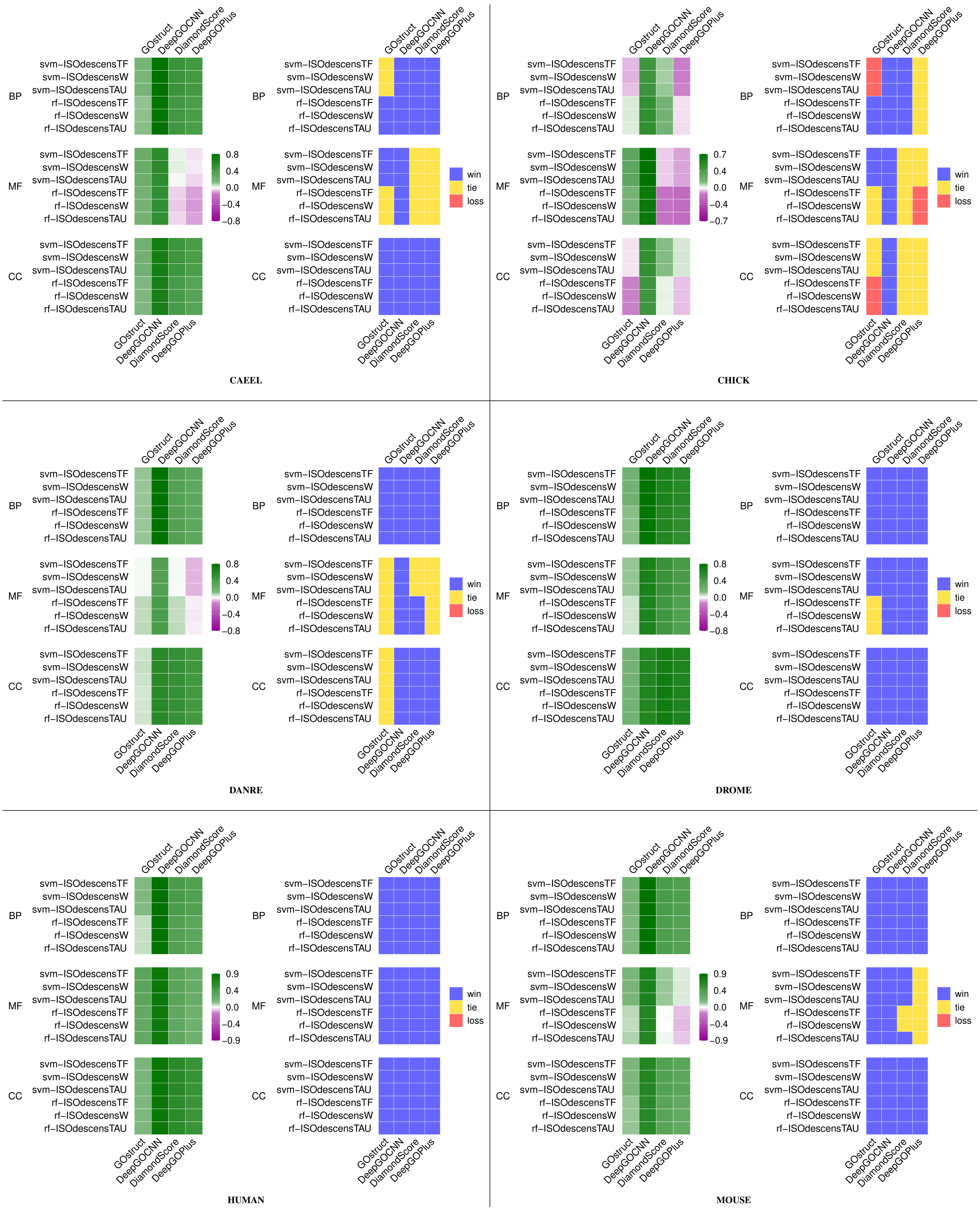


Fig. S19. Synoptic comparison of hierarchical ensemble methods vs structured output SOTA methods across GO ontologies and organisms for the metric AUPRC. In each panel the left heatmap refers to direct comparison of AUPRC results: green color indicates better HEMDAG results; the right panel refers to the paired Wilcoxon rank sum test. Blue cells show significantly better results for HEMDAG.

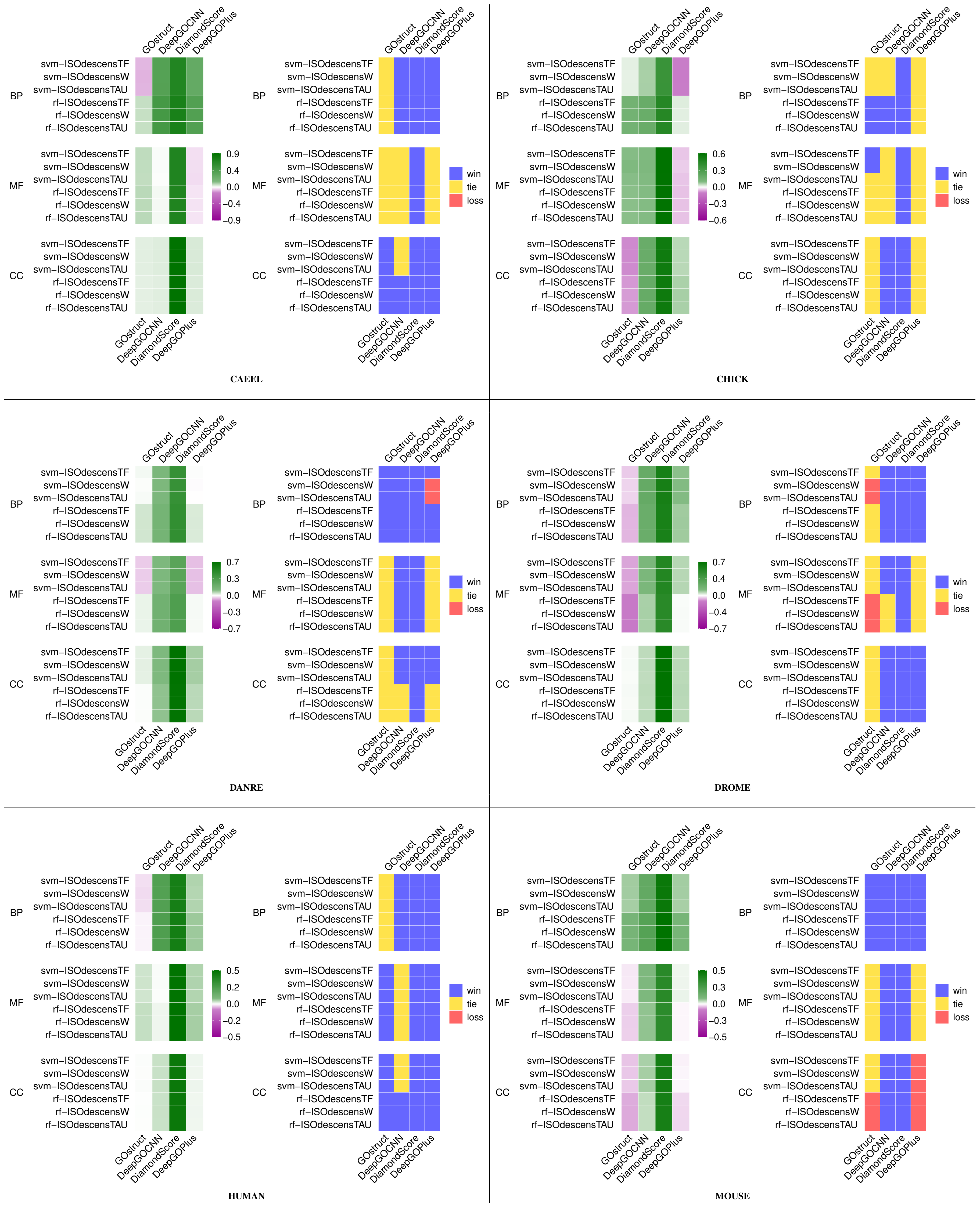


Fig. S20. Synoptic comparison of hierarchical ensemble methods vs structured output SOTA methods across GO ontologies and organisms for the metric F_{max}. In each panel the left heatmap refers to direct comparison of F_{max} results: green color indicates better HEMDAG results; the right panel refers to the paired Wilcoxon rank sum test. Blue cells show significantly better results for HEMDAG.

S8.5 Win-tie-loss

Table S14 shows for each performance metric how often HEMDAG wins, ties or loses against a SOTA approaches across organisms and GO domains, according to the two-sided paired Wilcoxon rank sum test ($\alpha = 0.05$).

Table S14. Counting of wins, ties and losses of ISO-DESCENS vs SOTA methods. For each performance metric, we have in total 432 occurrences since we considered 6 HEMs, 4 SOTA, 3 GO domains and 6 organisms.

	AUROC	**AUPRC**	$\mathbf{F_{max}}$	**Tot.**
win	336	340	268	944
tie	75	80	155	310
loss	21	12	9	42
Tot.	432	432	432	1296

Heatmap depicted in Fig. S21 compares ISO-DESCENS algorithms (rows) against competitor structured output methods (columns), by putting together the results coming from all organisms and GO domains. More precisely, the heatmap shows when the comparison of a pair is statistically significant according to a two-sided paired Wilcoxon rank sum test for the metric AUROC. In particular, we say that a hierarchical ensemble algorithm significantly "beats" (green cells) a competitor method if the Wilcoxon test rejects the null hypothesis ($\alpha < 0.05$); conversely, we say that a hierarchical ensemble significantly "loses"(blue cells) against the competitor and then we say that a hierarchical ensemble method "ties" (white cells) against a competitor method when no statistically significant difference is observed. Since we took into account six different organisms and three GO domains, a heatmap cell can range from a maximum of +18 (i.e., the hierarchical algorithm always "wins" against a SOTA method) to a minimum of -18 (i.e., the hierarchical algorithm always "loses" against a SOTA method).

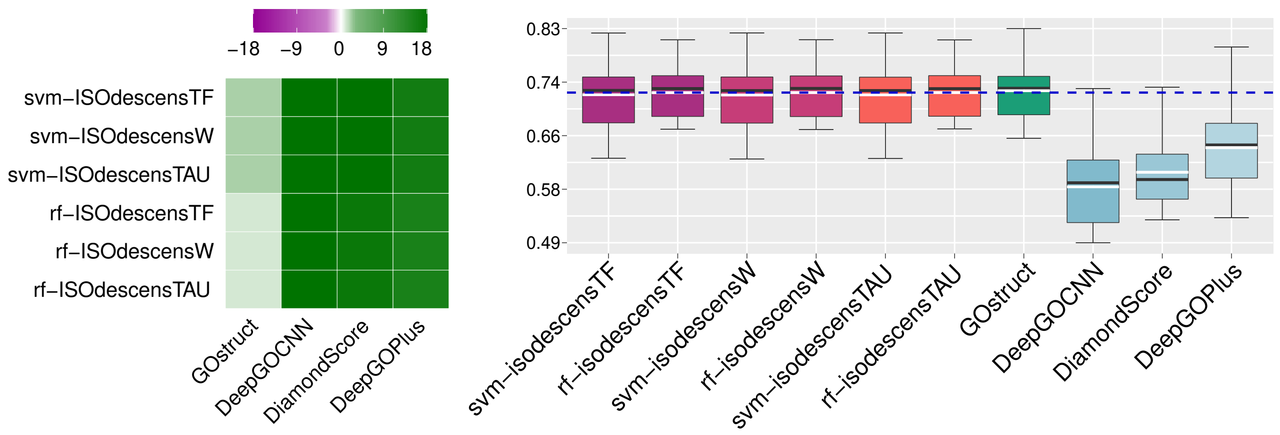


Fig. S21. Win-tie-loss heatmap and boxplot for the AUROC metric grouping the results coming from all organisms and GO domains and counting how many times an ensemble algorithm wins, ties or loses against a structured output SOTA method according to the two-sided paired Wilcoxon rank sum test with $\alpha < 0.05$. Since we took into account six different organisms and three GO domains, a heatmap cell can range from a maximum of +18 to a minimum of -18. The black and white line in the boxplots refer respectively to the median and the mean of an method across organisms and ontologies.

S8.6 Boxplots

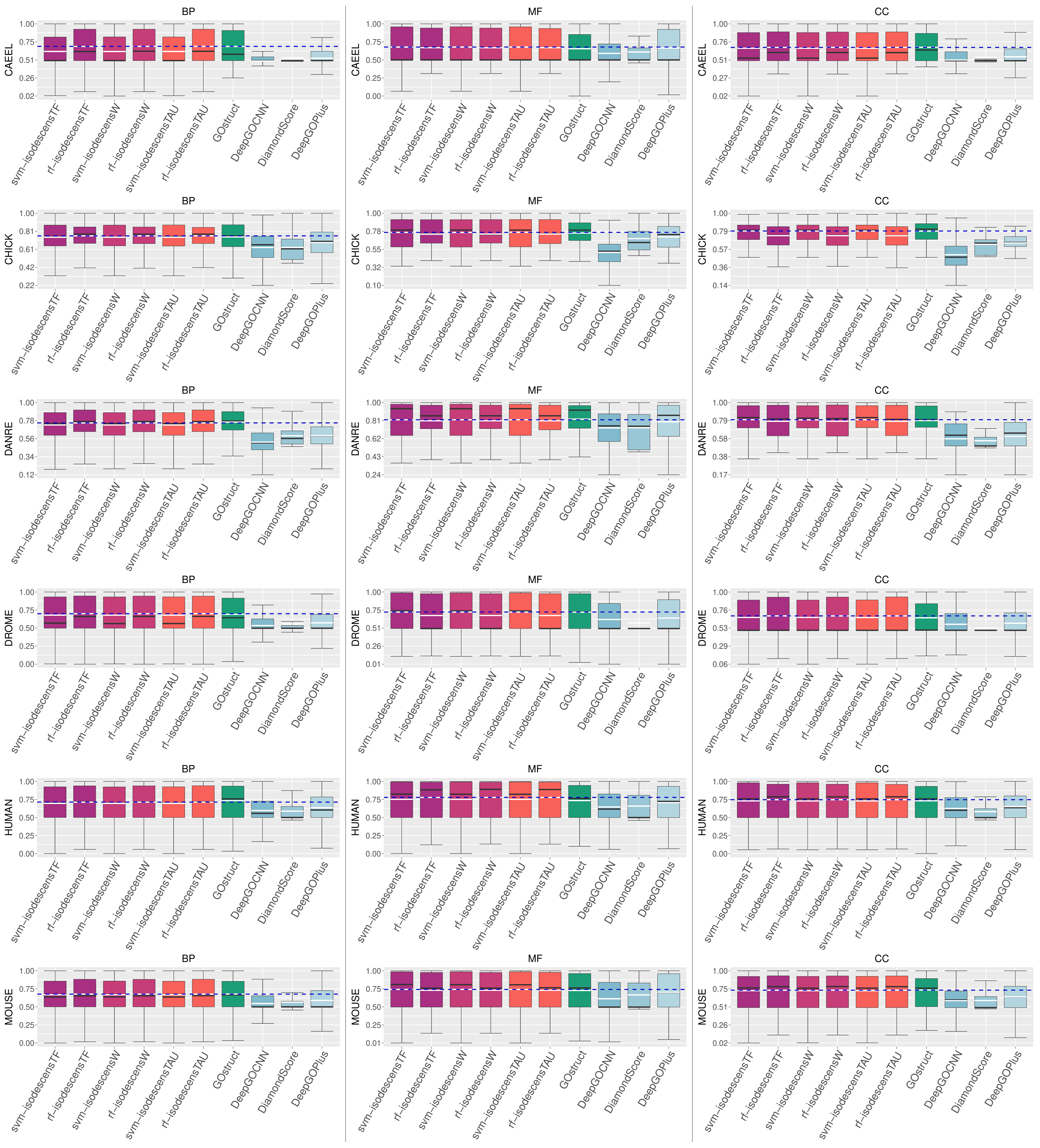


Fig. S22. Distribution of AUROC values across GO terms for each GO domain and organism between HEMDAG and structured output SOTA methods. The black and white line in the boxplots refer respectively to the median and the mean of an method computed across GO terms.

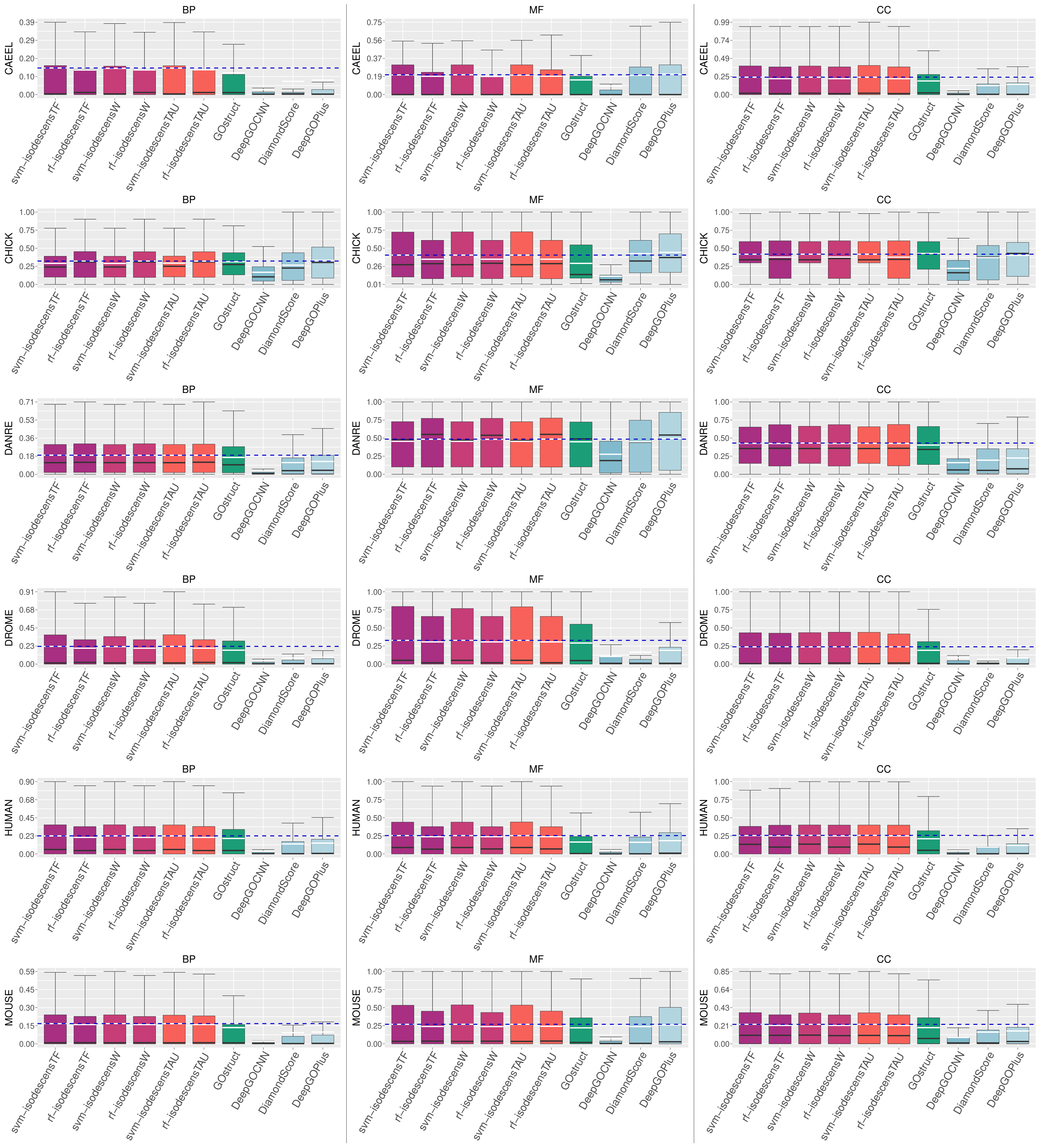


Fig. S23. Distribution of AUPRC values across GO terms for each GO domain and organism between HEMDAG and structured output SOTA methods. The black and white line in the boxplots refer respectively to the median and the mean of an method computed across GO terms.

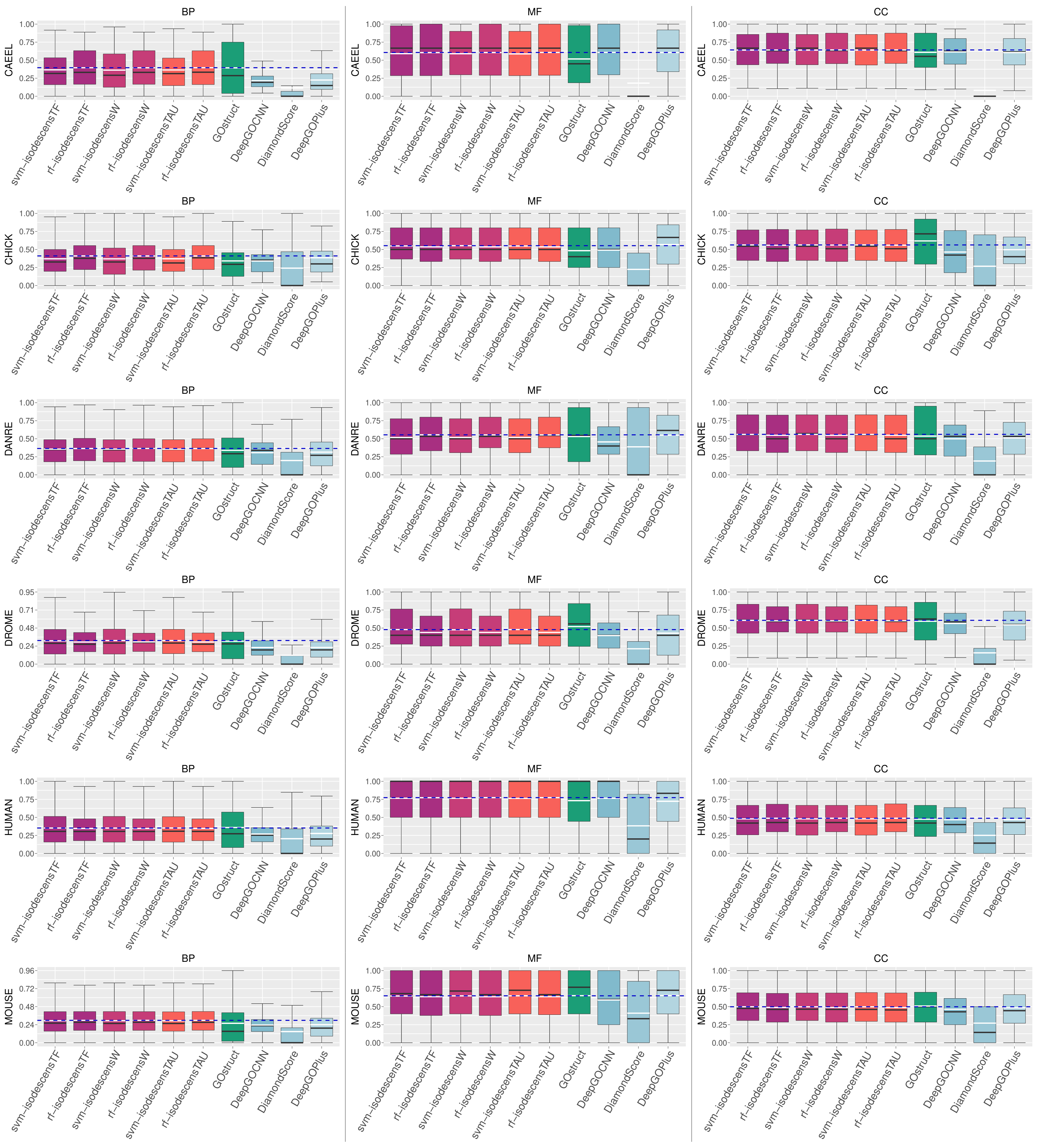


Fig. S24. Distribution of F_{max} values across GO terms for each GO domain and organism between HEMDAG and structured output SOTA methods. The black and white line in the boxplots refer respectively to the median and the mean of an method computed across GO terms.

S8.7 Precision-Recall curves

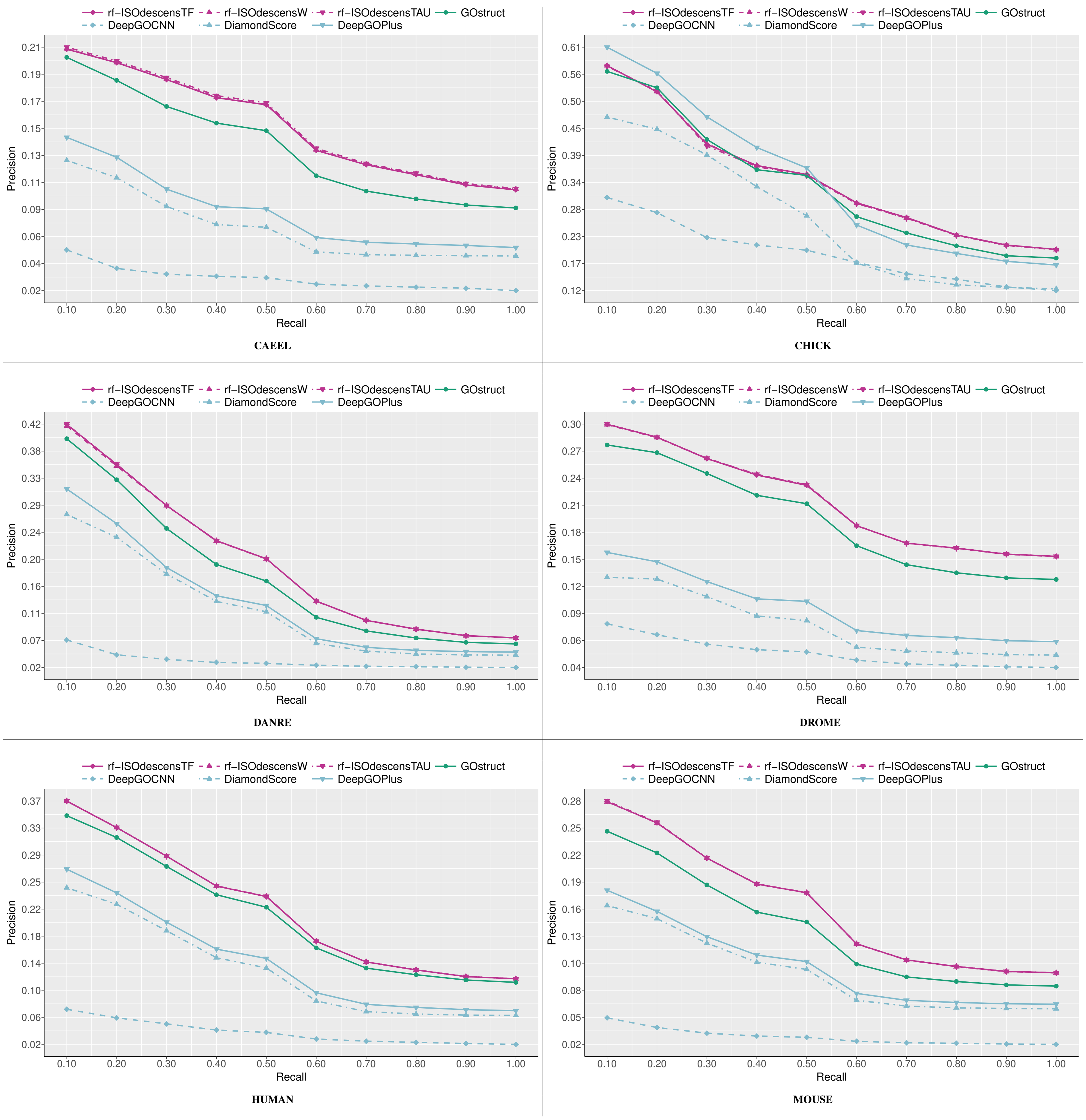


Fig. S25. Comparison of precision at different recall levels averaged across BP terms between ISO-DESCENS (by using rf as base learner) and SOTA structured output methods.

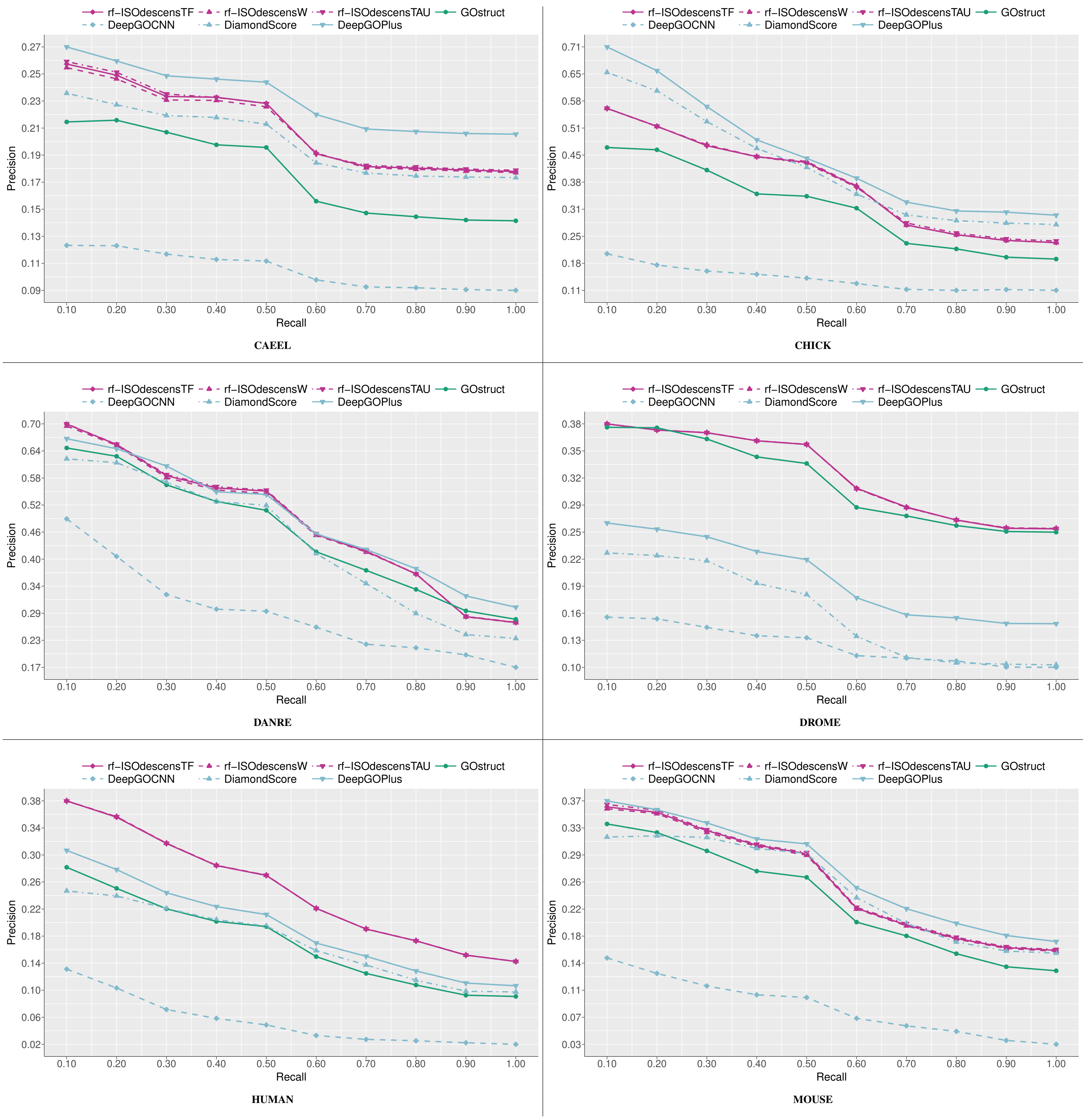


Fig. S26. Comparison of precision at different recall levels averaged across MF terms between ISO-DESCENS (by using rf as base learner) and SOTA structured output methods.

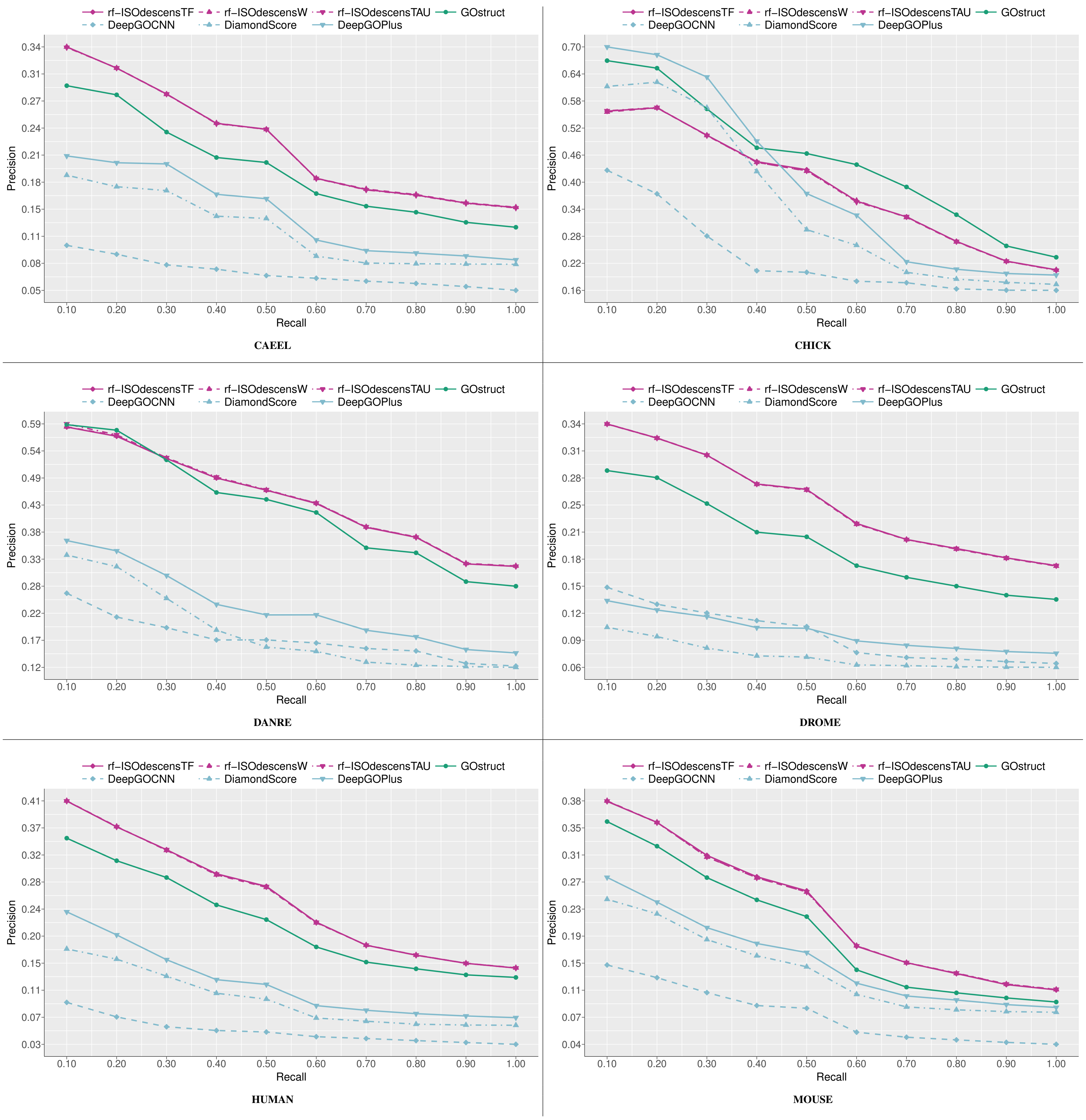


Fig. S27. Comparison of precision at different recall levels averaged across CC terms between ISO-DESCENS (by using rf as base learner) and SOTA structured output methods.

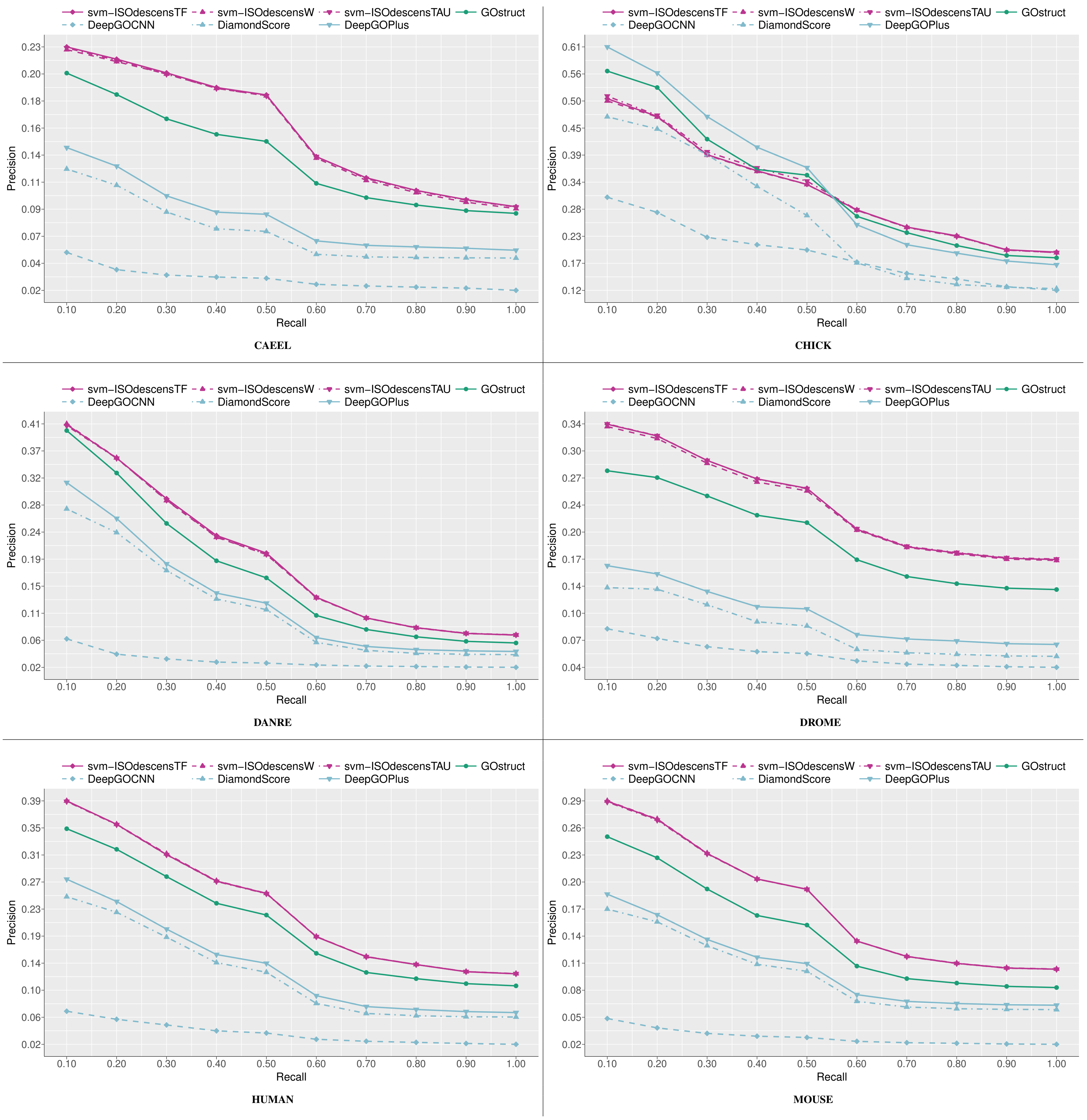


Fig. S28. Comparison of precision at different recall levels averaged across BP terms between ISO-DESCENS (by using SVMs as base learners) and SOTA structured output methods.

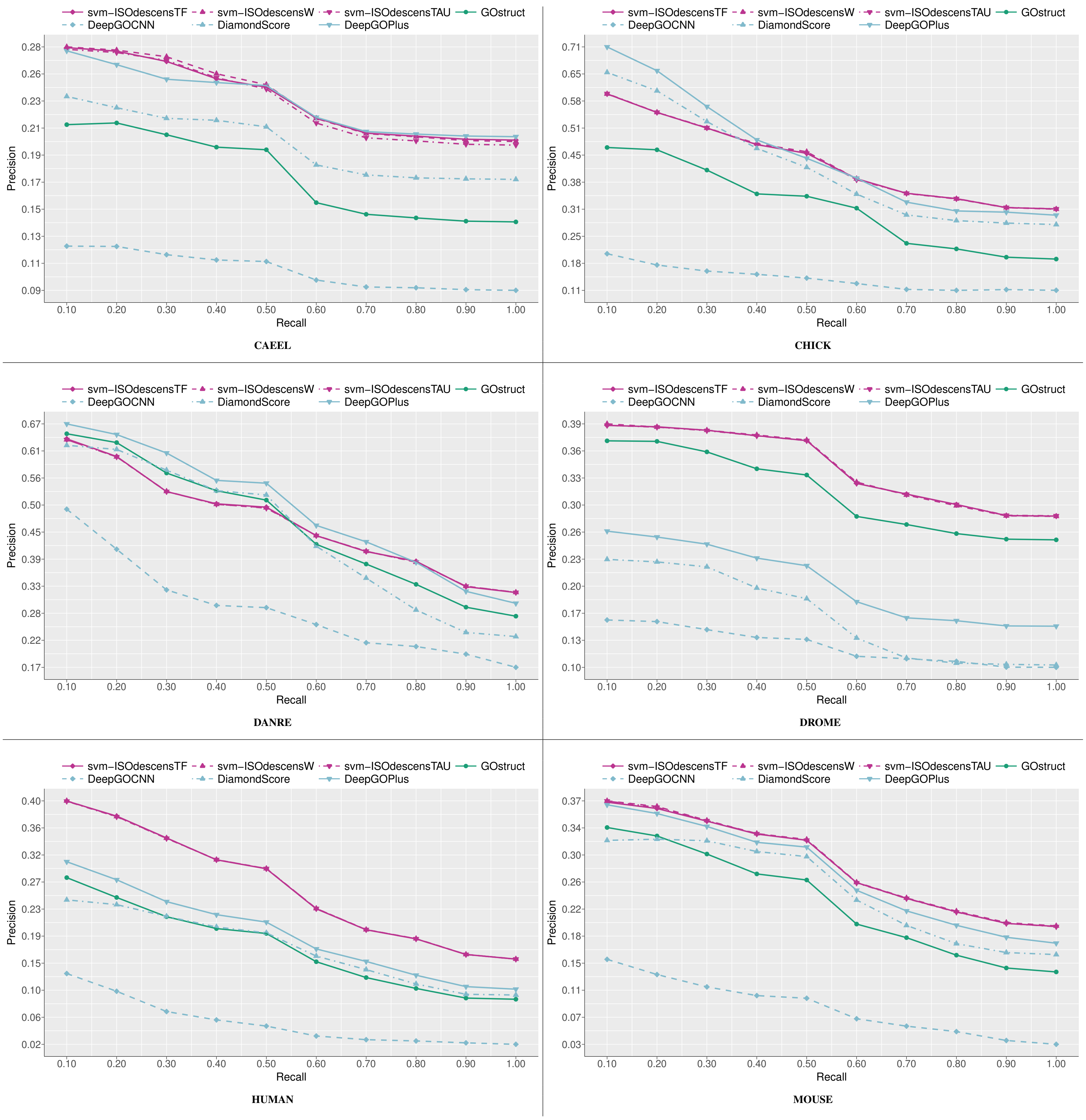
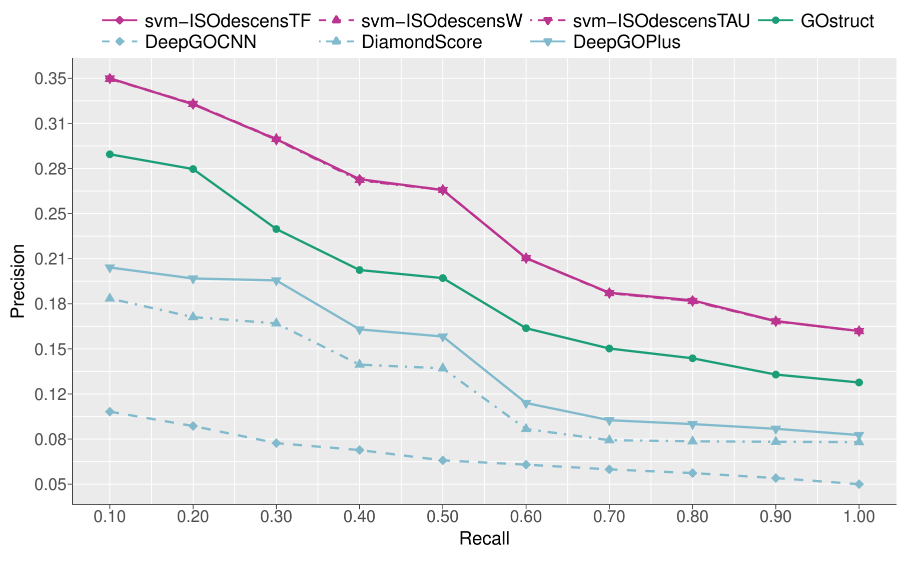
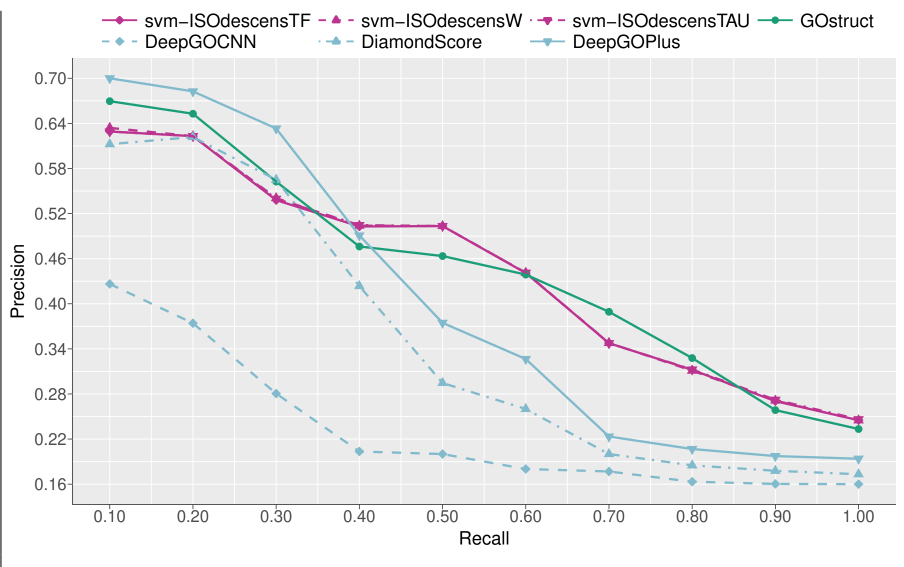


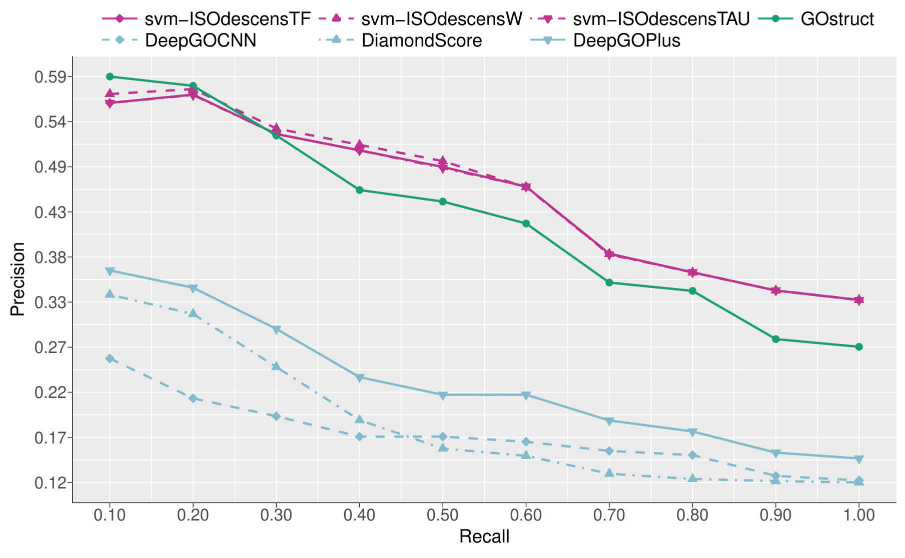
Fig. S29. Comparison of precision at different recall levels averaged across BP terms between ISO-DESCENS (by using SVMs as base learners) and SOTA structured output methods.



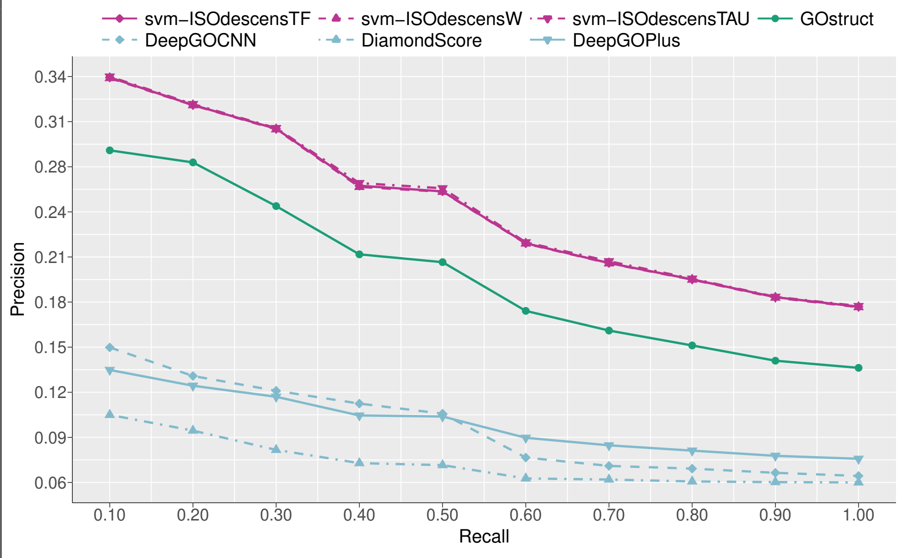
CAEEL



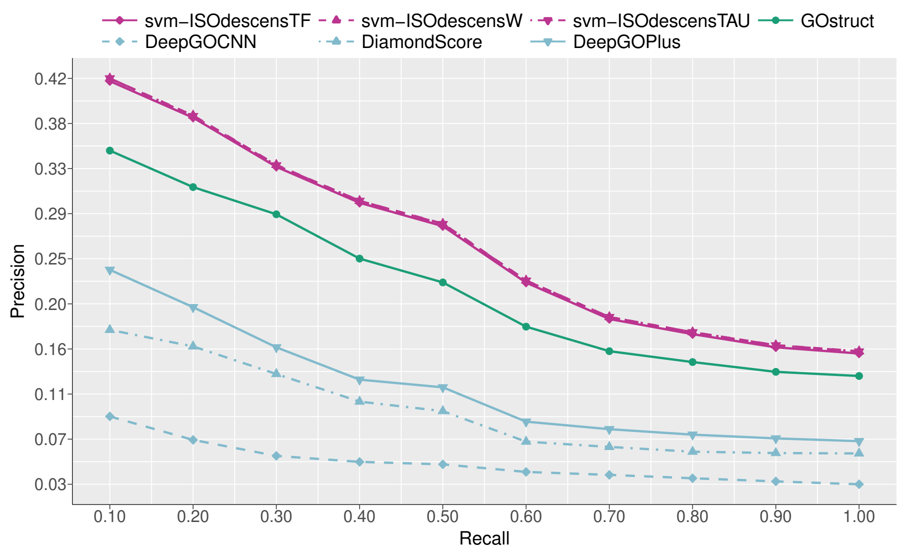
CHICK



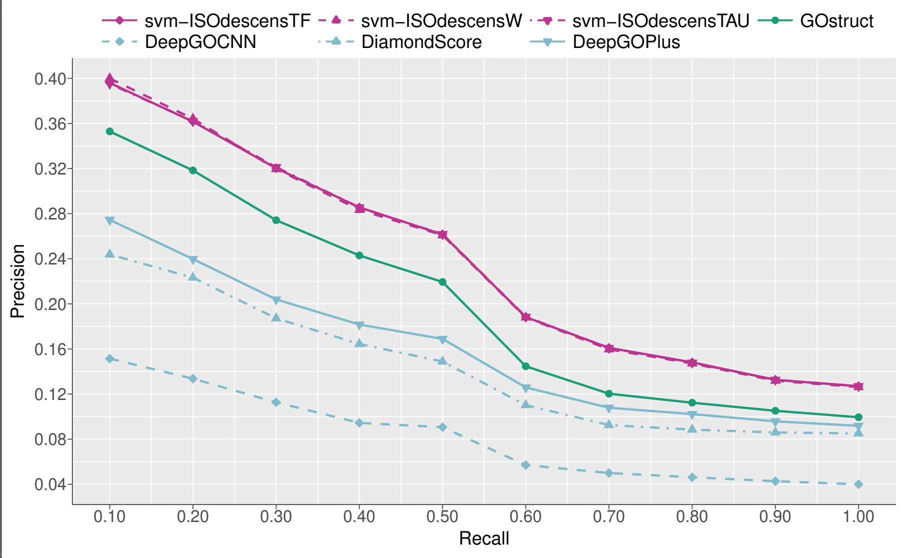
DANRE



DROME



HUMAN



MOUSE

Fig. S30. Comparison of precision at different recall levels averaged across BP terms between ISO-DESCENS (by using SVMs as base learners) and SOTA structured output methods.

S8.8 Empirical computational time

Table S15 compares empirical computational time in minutes (training + test phase) between HEMDAG algorithms and structured output SOTA methods. To calculate the running time of HEMDAG algorithms, we firstly averaged the computational time of the hierarchical algorithms and then we summed (separately for each GO domain) the elapsed time of the flat classifier to the average time of the hierarchical algorithms. The computational time of *svm* is low because we implemented a multi-thread version of LiblineaR. Instead, to calculate the elapsed time of DeepGOCNN, DiamondScore, DeepGOPlus we just summed (for each GO domain) the elapsed time of the deep convolutional neural network with that coming from respectively the sequence similarity (diamond) and the hierarchical correction step. For GOstruct, we simply report the elapsed time for each ontology. From Table S15 we can note that HEMDAG methods are in average significantly faster than GOstruct, DiamondScore, DeepGOCNN and DeepGOPlus across all the considered organisms and GO domains, even if two of our HEMs variants (i.e. ISOdescensW, ISOdescensTAU) have one hyper-parameter to be tuned. All the experiments were executed by using a machine having an Intel Xeon CPU E5-2630 2.60GHz with 128GB of RAM.

Table S15. Empirical computational time (in minutes): HEMDAG vs structured-output SOTA methods. rf stands for random forest and svm for support vector machine. The elapsed time (in minutes) includes both training and test phase. Numbers in brackets represent the elapsed time of the training and the test phase.

Organism	Domain	HEMDAG (rf+hem)	HEMDAG (svm+hem)	GOstruct	DeepGOCNN	DiamondScore	DeepGOPlus
CAEEL	BP	65 (43 + 22)	22 (2 + 20)	705	309 (309+0)	309 (309+0)	310 (309+1)
	MF	4 (3 + 1)	1 (0 + 1)	72	223 (223+0)	223 (223+0)	223 (223+0)
	CC	5 (4 + 1)	1 (0 + 1)	108	174 (174+0)	174 (174+0)	174 (174+0)
CHICK	BP	4 (2 + 2)	2 (0 + 2)	0	55 (55+0)	55 (55+0)	55 (55+0)
	MF	0 (0 + 0)	0 (0 + 0)	0	11 (11+0)	11 (11+0)	11 (11+0)
	CC	0 (0 + 0)	0 (0 + 0)	0	12 (12+0)	12 (12+0)	12 (12+0)
DANRE	BP	92 (67 + 25)	25 (3 + 22)	1565	318 (318+0)	318 (318+0)	320 (318+2)
	MF	3 (2 + 1)	1 (0 + 1)	4	153 (153+0)	153 (153+0)	153 (153+0)
	CC	1 (1 + 0)	0 (0 + 0)	0	95 (95+0)	95 (95+0)	95 (95+0)
DROME	BP	291 (206 + 85)	86 (10 + 76)	6324	487 (487+0)	487 (487+0)	491 (487+4)
	MF	13 (11 + 2)	2 (0 + 2)	347	330 (330+0)	330 (330+0)	330 (330+0)
	CC	20 (17 + 3)	4 (1 + 3)	715	332 (332+0)	332 (332+0)	332 (332+0)
HUMAN	BP	994 (707 + 287)	335 (67 + 268)	23971	841 (840+1)	840 (840+0)	850 (840+10)
	MF	340 (307 + 33)	92 (59 + 33)	19414	946 (946+0)	946 (946+0)	948 (946+2)
	CC	156 (143 + 13)	42 (28 + 14)	13021	616 (616+0)	616 (616+0)	618 (616+2)
MOUSE	BP	1129 (786 + 343)	385 (72 + 313)	26758	728 (727+1)	727 (727+0)	738 (727+11)
	MF	85 (77 + 8)	14 (6 + 8)	4127	581 (581+0)	581 (581+0)	582 (581+1)
	CC	79 (72 + 7)	13 (6 + 7)	5263	561 (561+0)	561 (561+0)	562 (561+1)

References

Agresti, A. and Coull, B. A. (1998). Approximate is better than exact for interval estimation of binomial proportions. *Statistical Science*, **52**(2), 119–126.

Ambroise, C. and McLachlan, G. J. (2002). Selection bias in gene extraction on the basis of microarray gene-expression data. *PNAS*, **99**(10), 6562–6566.

Barlow, R. (1972). *Statistical Inference Under Order Restrictions: The Theory and Application of Isotonic Regression*. J. Wiley.

Barrell, D., *et al.* (2009). The GOA database in 2009 - an integrated gene ontology annotation resource. *NAR*, **37**(Database-Issue), 396–403.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, **24**(2), 123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, **45**(1), 5–32.

Brown, L. D., *et al.* (2001). Interval estimation for a binomial proportion. *Statistical Science*, **16**, 101–133.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 785–794.

Clopper, C. J. and Pearson, E. S. (1934). The use of confidence or fiducial limits illustrated in the case of the binomial. *biometrika*, **26**(4), 404–413.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, **20**(3), 273–297.

Dettling, M. and Bühlmann, P. (2003). Boosting for tumor classification with gene expression data. *Bioinformatics*, **19**(9), 1061–1069.

Frasca, M. and Cesa-Bianchi, N. (2018). Combining cost-sensitive classification with negative selection for protein function prediction.

Friedman, J., *et al.* (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, **33**(1), 1—22.

Jiang, Y., *et al.* (2016). An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome Biol.*, **17**(1), 184.

Kuhn, M. (2008). Building predictive models in r using the caret package. *Journal of Statistical Software, Articles*, **28**(5), 1–26.

Kulmanov, M. and Hoehndorf, R. (2020). DeepGOPlus: improved protein function prediction from sequence. *Bioinformatics*, **36**(2), 422–429.

Louis, T. (1981). Confidence intervals for a binomial parameter after observing no successes. *The American Statistician*, **35**, 154.

Notaro, M., *et al.* (2017). Prediction of human phenotype ontology terms by means of hierarchical ensemble methods. *BMC Bioinform.*, **18**(1), 449:1–449:18.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Radivojac, P., *et al.* (2013). A large-scale evaluation of computational protein function prediction. *Nat. methods*, **10**(3), 221–227.

Rumelhart, D., *et al.* (1986). Learning representations by back-propagating errors. *Nature*, **323**(6088), 533–536.

Saito, T. and Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, **10**(3), 1–21.

Szklarczyk, D., *et al.* (2015). STRING v10: protein-protein interaction networks, integrated over the tree of life. *NAR*, **43**, 447–452.

Szklarczyk, D., *et al.* (2018). STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *NAR*, **47**(D1), D607–D613.

Valentini, G. (2011). True path rule hierarchical ensembles for genome-wide gene function prediction. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, **8**(3), 832–847.

Venables, W. N. and Ripley, B. D. (2002). *Modern applied statistics with S, 4th Edition*. Statistics and computing. Springer.

Zhang, H. (2004). The optimality of naive bayes. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, Miami Beach, Florida, USA*, pages 562–567.

Zhou, N., *et al.* (2019). The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. *Genome Biol.*, **20**(1), 244.