

划分、专业化和路由：一种新的高效集成学习方法

Jakub Piwko, Jdrzej Ruciski, Dawid Pudowski, Antoni Zajko, Patryzja ak,
Mateusz Zacharecki, Anna Kozak, Katarzyna Wonica

Warsaw University of Technology
jakub.piwko2.stud@pw.edu.pl, jedrzej.rucinski.stud@pw.edu.pl,
dawid.pludowski.stud@pw.edu.pl

摘要 集成学习已被证明能够有效提升预测性能，但诸如装袋法、提升法和动态集合选择（DES）等传统方法存在计算成本高且适应异构数据分布能力有限的问题。为解决这些问题，我们提出了 Hellsemble，这是一种新颖且可解释的二元分类集成框架，在训练和推理过程中利用了数据集复杂性。Hellsemble 通过迭代地将误分类实例从较简单的模型传递给后续模型，逐步将数据集划分为“难度圈”，形成一组专业基础学习器。每个模型在越来越具有挑战性的子集上进行训练，而一个单独的路由模型则根据推断出的难度来分配新实例到最合适的基线模型。Hellsemble 实现了强大的分类准确率，同时保持计算效率和可解释性。在 OpenML-CC18 和 Tabzilla 基准测试中的实验结果表明，Hellsemble 通常优于经典集成方法。我们的研究发现表明，采用实例级别的难度为构建高效且稳健的集成系统提供了有希望的方向。

Keywords: 集成学习 · 动态集成选择 · 二分类问题

1 介绍

最近，经典机器学习越来越倾向于集成方法——结合多种学习算法以提高预测性能的技术。这些方法侧重于在相同数据集上训练多个模型，然后以各种方式聚合它们的输出以进行最终预测。流行的示例包括装袋法，例如随机森林 [3]、提升法，例如 XGBoost [7]、投票集成 [10] 以及更复杂的 AutoML 系统中使用的策略，如 AutoGluon [12]、AutoSklearn [14] 或 TPOT [20]。集成方法通常有助于减少过拟合，特别是在集成包含捕获数据中不同模式且能够弥补委员会中其他模型弱点的多样化模型时。

然而，集成方法通常存在一些缺点。它们通常计算成本较高，因为需要训练多个模型，并且往往是在整个数据集上进行训练。此外，在推理过程

中，为了做出单一预测，通常需要所有组件模型参与，这导致了较高的计算成本。

传统全局集成的一个局限性是它们对数据集进行统一处理。现实世界的数据集通常很复杂，表现出异构结构、复杂的依赖关系以及特征空间中的不平衡分布。应用单一模型甚至全局集成可能导致注意到主导模式，而忽略一些细微但重要的关系。在这种情况下，模型可能在数据集中代表性不足或更难处理的区域泛化能力较差。

为了解决这个问题，提出了动态集成选择 (DES) 方法。这些方法在一个模型池上训练数据集，并在推理时动态选择最合适的模型来预测每个新实例。选择通常基于每个模型在查询实例局部邻域上的性能，通常是通过特征空间中的最近邻来识别的。

虽然 DES 方法通过利用专精于特征空间不同区域的模型更好地处理异构数据，但它们仍然会在推理时间产生显著的开销。这包括寻找新实例的邻域以及维护所有模型的详细性能记录。此外，它们的性能在很大程度上取决于定义邻域时所选择的度量标准和模型组合策略。而且，DES 中的所有模型仍然是在整个数据集上进行训练的，这可能会限制它们的专业化，并错过利用数据结构分割的机会。

在本文中，我们提出了**地狱聚类**—一种用于二分类任务的新颖集成学习框架，在训练和推理过程中明确利用了数据集的复杂性，同时保持较低的计算成本。与传统的集成方法在完整数据集上拟合多个模型不同，Hellsemble 在训练过程中逐步将数据集分割为难度递增的子空间。这是通过检查每个模型按顺序正确或错误分类哪些实例来实现的。被一个简单模型误分类的实例会被传递给链中的下一个简单模型，后者试图更有效地学习其结构。

这一过程产生了一个由专业模型组成的委员会，每个模型专注于数据集中的不同子集。这不仅减少了训练开销，还提供了关于数据集中实例难度分布的有价值元信息。

本文的主要贡献是：

1. 我们引入了**地狱组合**，这是一个新颖、简单且可解释的框架，用于创建二分类任务中的集成模型，结合了数据集分割和动态模型选择技术。
2. 我们证明了 Hellsemble 在 OpenML-CC18 和 Tabzilla 的基准数据集上，与经典机器学习模型相比具有竞争力的表现，并且通常在分类准确性方面优于它们。

2 相关工作

集成学习因其通过结合多个模型来提高预测准确性的能力，已成为机器学习中广泛采用的一种技术。经典策略包括装袋法，它通过对自助样本 [9] 独立训练模型构建集成，以及提升法 [15]，该方法按顺序专注于纠正先前模型的错误。另一种流行的方法是投票，其中多个基础模型进行预测，最终输出由多数或加权共识决定。堆叠在此基础上更进一步，通过训练元模型来整合基础学习器 [21] 的预测结果。近年来的 AutoML 框架，如在自动化机器学习系统中发现的那些，则采用贪婪搜索或优化策略来高效地选择和组合模型 [4, 5]。像 AutoSklearn [14] 和 AutoGluon [13] 这样的系统不仅实现了模型选择的自动化，还实现了集成构建的自动化，通常在极少的人工干预下就能获得稳健且有竞争力的表现。

动态模型选择旨在通过将预测过程定制到每个特定实例来提高性能。这种方法不是对所有输入应用固定的集成，而是根据输入空间局部区域的估计能力动态地选择一个或多个模型 [17, 6]。通常，这涉及使用基于距离的方法（如 k 近邻）分析查询实例的邻域，以评估每个模型在类似过去的案例中表现如何 [16, 22]。然后为该实例选择最合适的模型，可能提高异构数据集中的准确性，在这些数据集中，特征空间的不同区域需要不同的归纳偏置。一些技术包括对每个模型的能力进行特殊估计，这有助于选择正确的预测器集合 [8]。这种适应性使系统能够比静态集成更有效地处理复杂的数据分布。

另一个有望提升集成性能的方向涉及路由机制，其中输入实例根据学习规则或分类器输出被定向到特定模型或模型子集。这些基于路由器的系统通常通过训练一个独立的路由函数来操作，该函数学会将数据点分配给最适合处理它们的模型 [1]。这种方法在深度学习中被使用，在图像分类中创建通往适合任务的模型路径 [19]，以及在大语言模型中根据查询难度将其移动到合适的模型 [11]。通过在训练过程中学会划分输入空间，这些方法可以减少集成中的冗余并避免过拟合，同时提供并行推理和扩展性的机会。

3 地狱组合

Hellsemble 框架建立在三个关键概念之上：动态模型选择、迭代（从预定义集合中顺序或贪婪）模型添加，以及基于路由器的预测路由。具体来说，Hellsemble 为每个新实例仅从其委员会中选择一个模型来进行预测，使其成为一种动态选择方法。在训练过程中，它采用贪心策略，在每次迭代时选择

能够提供最大验证性能提升的模型。此外，还会训练一个专用的路由器模型来确定在推理期间应由委员会中的哪个基础模型处理给定输入。为了缓解过拟合问题，Hellsemble 在整个训练过程中使用验证子集评估模型性能，并基于过拟合控制应用正则化策略。该框架以 Python 实现，用户必须提供候选基础模型列表和一个路由器模型。

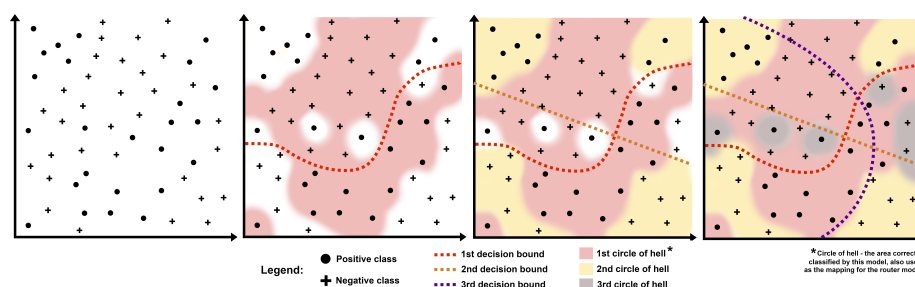


图 1. Hellsemble 的训练和数据集评估方案

训练过程以及数据集在各次迭代中的演变如图 1 所示，我们在本节中多次参考此图。

Hellsemble 构建我们所谓的“难度圈”——基于模型在每次迭代中对数据的表现。在每一轮中，算法会选择那种当加入当前 Hellsemble 组合时能带来验证分数最高提升的模型。这个分数可以是任何评估指标，如准确率或 AUC，甚至是一个用户自定义的函数。

一旦在给定的迭代中选择了最佳模型，我们就分析它正确和错误预测了哪些观测值。在基础版本中，只有分类错误的实例会被传递到下一次迭代。这些是比较有挑战性的例子，目标是找到另一个能够更好地处理它们的模型。在图 1 中，这被可视化为不同模型决策边界的重叠彩色区域。

为避免过拟合并允许难度级别之间存在重叠，我们还在下一次迭代中包含一部分正确分类的示例。此类示例的数量是根据训练数据集和验证数据集之间的性能差异来确定的——具体来说，是为了最小化它们之间的差距。这作为一种正则化的形式，确保每个模型能够推广到更广泛的区域。

每次迭代后，路由器模型都会重新训练。基于累积的难度信息——每个实例被过滤掉的迭代次数——我们将实例分配到表示其“难度圈”的类别中。这实际上将问题转换为一个多类分类任务，在这个任务中，路由器学会

了预测给定输入应使用哪个基础模型。因此，路由器模型捕获了数据集按不同难度划分的区域。

这个迭代过程会持续到不再有分类错误的实例剩余，或者没有额外的模型能提高验证分数。在训练结束时，我们获得了一组基础模型列表，每个模型都是基于数据中越来越难处理的子集进行训练的，以及一个路由模型，它将新实例映射到合适的基模型。

Hellsemble 的关键优势在于其简洁性和高效性。由于每个基础模型仅在数据的特定子集上进行训练，因此它们不需要是高度通用的模型——只要目标区域内准确，简单模型就足够了。相反，路由器模型必须具备足够的表达能力来捕捉非线性边界，因为它执行一个多类分类任务，决定将预测委托给哪个模型。这种结构使得 Hellsemble 成为一种可解释且高效的集成学习策略。

3.1 顺序 Hellsemble

顺序 Hellsemble 将用户指定的模型顺序视为固定顺序，在此顺序中，模型被训练并添加到连续的难度圈中。其细节在算法 1 中指出。

Algorithm 1 顺序 Hellensemble**Require:** 数据集 D , 基础模型 $M = \{M_1, M_2, \dots, M_k\}$, 路由模型 R

```

1: 将  $D$  分为  $D_{\text{train}}$  和  $D_{\text{val}}$ 
2:  $E \leftarrow \emptyset$  ▷ Empty ensemble
3:  $D_{\text{router}} \leftarrow \emptyset$ 
4:  $S_{\text{prev}} \leftarrow 0$ 
5: for  $i = 1$  to  $k$  do
6:   在  $D_{\text{train}}$  上训练  $M_i$ 
7:    $E \leftarrow E \cup \{M_i\}$ 
8:   用类别  $i$  标记  $D_{\text{train}}$  中的每个实例
9:    $D_{\text{router}} \leftarrow D_{\text{router}} \cup D_{\text{train}}$ 
10:   $R \leftarrow$  在  $D_{\text{router}}$  上训练路由器 (作为多类分类器)
11:   $S_{\text{curr}} \leftarrow$  使用  $R$  评估  $E$  在  $D_{\text{val}}$  上的表现
12:  if  $S_{\text{curr}} \leq S_{\text{prev}}$  then
13:    中断
14:  else
15:     $S_{\text{prev}} \leftarrow S_{\text{curr}}$ 
16:  end if
17:   $D_{\text{hard}} \leftarrow$  被  $M_i$  错分的  $D_{\text{train}}$  中的实例
18:   $D_{\text{router}} \leftarrow \emptyset$  正确预测实例的一部分  $\alpha$ 
19:   $D_{\text{router}} \leftarrow \emptyset$ 
20: end for
21:  $D_{\text{router}} \leftarrow \emptyset, R$ 

```

3.2 贪心地狱组合

贪婪地狱集成是一种计算强度更高的顺序地狱集成变体。它在每次迭代中通过评估所有可用候选者来动态选择最有前景的模型，而不是使用固定顺序的模型。

核心算法与 Sequential Hellsemble 保持一致，但有以下关键更改：

- **每次迭代的模型评估：** 代替按照预定义的顺序选择下一个模型，在每次迭代中，列表中的所有模型都在当前训练集上进行训练。
- **基于验证性能的贪婪选择：** 每个训练好的模型都会暂时添加到集成中，并在验证集上进行评估。只有能带来最佳性能提升的模型才会被选中并永久添加到集成中。

剩余步骤——包括路由器训练、训练集更新、正则化和停止准则——与算法 1 保持不变。

以下伪代码替换了算法 1 的贪婪变体中的第 5 至第 7 行：

Algorithm 2 贪婪模型选择（替换算法 1 的第 5 至 7 行）

```

1:  $best\_score \leftarrow -\infty$ 
2:  $best\_model \leftarrow \text{None}$ 
3: for each model  $M_j$  in model list do
4:    $M_j \leftarrow$  在  $D_{\text{train}}$  上训练
5:    $E \leftarrow E \cup \{M_j\}$  ▷ Temporarily add to ensemble
6:    $score_j \leftarrow$  在  $D_{\text{val}}$  上评估  $E$ 
7:   if  $score_j > best\_score$  then
8:      $best\_score \leftarrow score_j$ 
9:      $best\_model \leftarrow M_j$ 
10:  end if
11:   $E \leftarrow E \setminus \{M_j\}$  ▷ Remove temporary model
12: end for
13:  $E \leftarrow E \cup \{best\_model\}$  ▷ Permanently add best model

```

4 实验

为了评估提出的 Hellsemble 框架的性能，我们使用了两个基准二分类数据集集合进行了大量实验：**OpenML-CC18** [2] 和**标签扎布拉** [18]。这些数据集在复杂性上有所不同，Tabzilla 提供了更具挑战性的结构和更高的特征维度。总的来说，它们为评估模型的泛化能力和适应性提供了一套可靠的工具。

实验设置旨在研究不同基础模型和路由器模型组合对分类性能的影响。我们选择了：

- 基模型的 4 种不同配置（即不同的分类器集合），
- 4 种不同的路由器型号。

每个基础模型配置与每个路由器模型恰好配对一次，总共产生了 **16 种独特的 Hellsemble 配置** 种组合。表 1 总结了我们在实验中使用的基模型和路由器模型的组合。

表 1. 实验中使用的基模型和路由器模型组合的总结。

基础模型	路由模型
KNN, Logistic Regression, Decision Tree, GaussianNB	KNN ($k=3$)
	KNN ($k=5$)
	MLP
	Random Forest
Random Forest, XGBoost, MLP	KNN ($k=3$)
	KNN ($k=5$)
	MLP
	Random Forest
KNN ($k=3$), KNN ($k=5$), Decision Tree, GaussianNB	KNN ($k=3$)
	KNN ($k=5$)
	MLP
	Random Forest
All models (KNNs, Decision Tree, GaussianNB, RFs, XGBoost, KNN ($k=3$) MLPs)	KNN ($k=5$)
	MLP
	Random Forest

对于每个数据集，我们执行以下步骤：

1. 数据集使用分层抽样方法被分为训练子集和测试子集。
2. 每个 Hellsemble 配置都使用了算法的**顺序的**和**贪婪**变体进行训练。请注意，由于 Hellsemble 框架的性质，每个集成中包含的模型数量可能根据迭代过程有所不同。
3. 为了作为基线，集成配置中使用的所有单独基础模型也都分别在训练集上进行了训练。
4. 最后，训练好的 Hellensemble 模型和各个基础模型都在测试集上进行了评估。这使得可以对集成性能与独立分类器进行直接比较。

本次实验评估的目标是评估在不同配置下，Hellsemble 是否能持续优于单个模型，并观察其在顺序变体和贪婪变体之间的性能差异。

5 结果

本节介绍了实验结果的总结。我们的主要目标是比较基础模型与提出的 Hellsemble 模型（包括顺序和贪婪变体）在各种配置下的性能。对于每种基础模型套件和路由组合，我们使用平均准确性指标来评估性能。

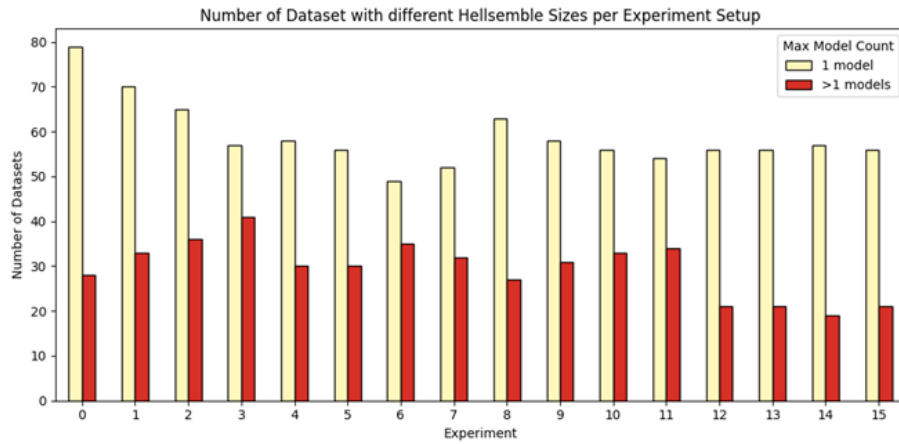


图 2. 每个配置中 Hellsemble（无论是顺序的还是贪婪的）由恰好一个模型组成与多于一个模型组成的实验比例。

在分析性能指标之前，我们首先考察 Hellsemble 构建包含多个模型的集成的频率。重要的是，在训练过程中我们并没有强制要求构造多模型——当添加额外模型无法提升验证表现时，Hellsemble 自然会停止增加模型。这意味着单模型的 Hellsembles 在某些情况下是有效且预期之内的。

图 2 表明，在大多数配置中，Hellsemble 倾向于仅由单一模型组成。特别地，使用最广泛的基模型套件的配置（最后四个设置）导致多模型 Hellsemble 的比例最低。另一方面，配置 3 和 4 在模型数量方面产生了最多样的 Hellsemble。

对于以下的准确性分析，我们将注意力限制在那些数据集和实验配对中，其中得到的 Hellsemble（无论是顺序还是贪婪）包含了一个以上的模型。这种关注点使我们能够评估 Hellsemble 的多模型特性是否有助于提高个体基础模型的表现。

为此，我们将所有模型（基础模型和 Hellsemble 变体）在所有选定数据集和配置上的准确率分数进行了汇总。结果如下图所示，按基础模型套件和路由模型分组。

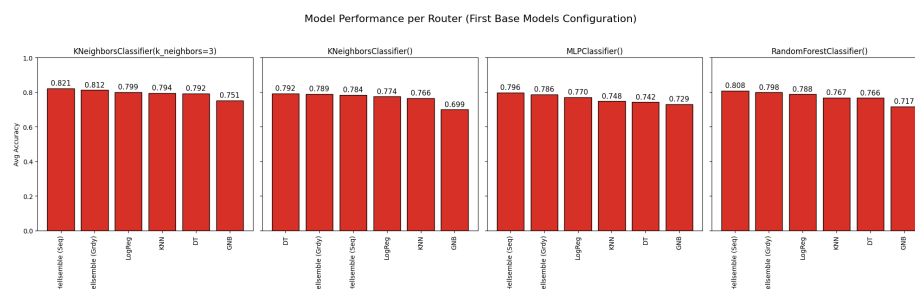


图 3. 所有数据集上不同路由模型和第一组基础模型（例如，KNN、朴素贝叶斯、逻辑回归和决策树）的平均准确率。

图 3展示了使用经典简单模型如 KNN、高斯朴素贝叶斯、逻辑回归和决策树的配置结果。在几乎所有路由场景中，Hellsemble 在平均准确率方面优于各个基础模型。唯一的例外是在将默认设置下的 KNN 用作路由器时——在这里，决策树表现最佳。然而，这些结果显示，即使使用基本分类器，Hellsemble 也能提高准确性。

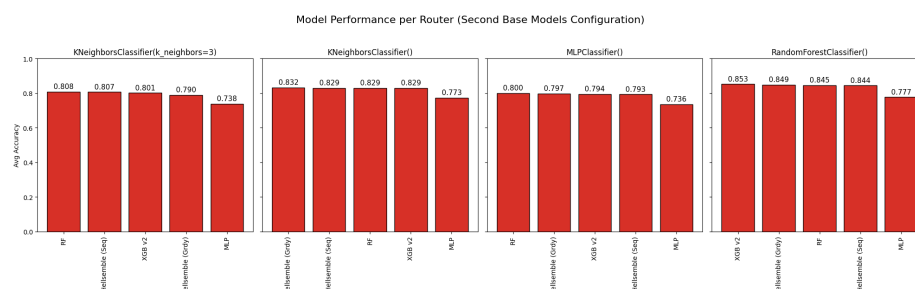


图 4. 所有数据集上不同路由模型和第二个基础模型套件（例如，随机森林和 XGBoost）的平均精度。

图 4比较了在使用更高级模型（如随机森林和 XGBoost）配置下的性能，这两种都是适用于表格数据的强大集成学习器。在这类设置中，Hellsemble

的表现略逊一筹。在多个配置中，例如使用决策树或 MLPs 作为路由器的配置下，XGBoost 超越了所有其他模型。只有一个配置（第二个路由器）中，Hellsemble（贪婪模式）取得了最佳结果。

这些发现表明，当基础模型已经是强大的集成方法时，Hellsemble 可能难以带来进一步的提升。这也可能反映出，在所有基础模型本身都很强的情况下，路由器正确分配实例到适当难度“圈层”的效果有限。然而，Hellsemble 主要是为简单模型能够专门处理数据子集的情景设计的——这一优势在基础模型已经很好地泛化时变得不那么相关。

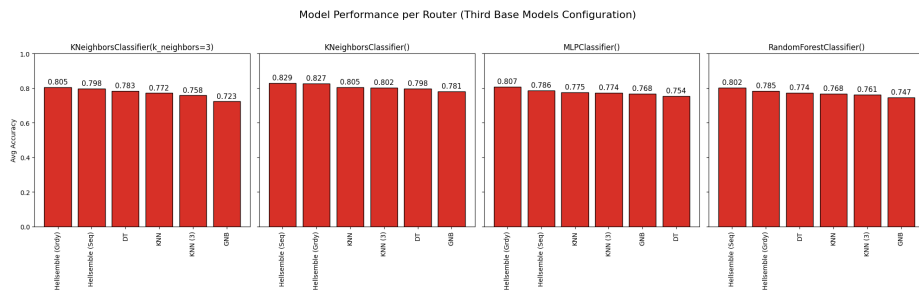


图 5. 所有数据集上的不同路由模型和第三个基础模型套件（由更简单的分类器组成）的平均准确性。

在图 5 中，我们分析了一个完全由较简单模型组成的套件的结果。在这种情况下，Hellsemble 在所有路由类型中始终优于所有基础模型。这些结果支持 Hellsemble 特别适合涉及轻量级分类器设置的假设。所观察到的性能提升为 2 – 3 个百分点，并且没有增加实质性的复杂性——只是通过将样本路由到更专业的模型来实现。

最后，图 6 展示了包含 11 个分类器的最全面基础模型套件的结果，包括基本模型、决策树、XGBoost 和 MLPs。尽管各个模型都很强大，Hellsemble 仍然在每种路由器配置中都超越了所有基础模型。顺序 Hellsemble 在使用 KNN（3 个邻居）作为路由器时表现最佳，而贪婪 Hellsemble 则在其他设置中占据主导地位。

这一结果特别有前景：即使基础模型池中包括像 XGBoost 和随机森林这样的强模型，Hellsemble 也能形成专门的集成模型，实现更优的表现。尽管改进幅度不大，但它们验证了 Hellsemble 的前提——基于数据子区域利用模型专业化可以提高整体准确性。

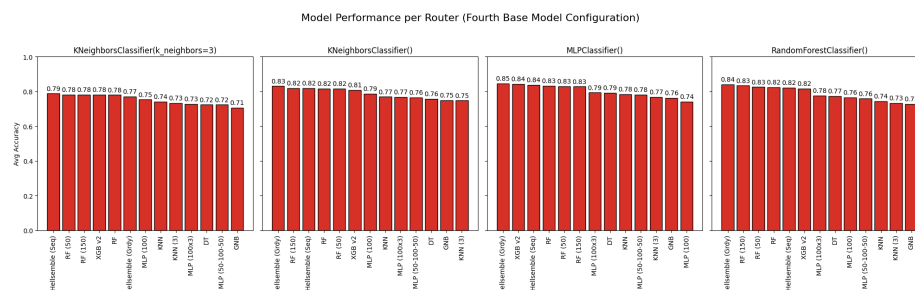


图 6. 所有数据集上不同路由模型和第四个基础模型套件（包含简单和高级模型）的平均准确率。

值得注意的是,在这种配置下,较少的数据集导致了多模型 Hellsembls。然而,当形成这样的集合时,它们的表现展示了 Hellsemble 架构的潜力。这些发现表明,通过难度感知路由实现模型专业化是提高分类器性能的一种可行策略,尤其是在使用较简单的模型时。

6 结论

Hellsemble 是一种结合动态集成选择、贪婪模型构建和基于路由的实例选择概念的新型集成学习框架。它通过在数据难度逐渐增加的数据子集上顺序训练基础模型来构建专业化的集成，提出了一种简单而有效的策略——这些难度递增的数据子集由前一个模型分类错误的实例定义。

此设计使 Hellsemble 既具有可解释性又高效。通过使用轻量级模型并将其训练在较小的、专注的数据子集上，该框架实现了具有竞争力的表现，同时保持了较低的计算成本。在推理时，仅需路由器和单一专用模型参与预测，与传统集成方法相比，这减少了延迟。

在两个基准数据集集合和一系列模型配置上的实验结果表明，Hellsemble 可以在许多设置中超越其基础模型，特别是在基础学习器是简单分类器的情况下。即使在包含像 XGBoost 和随机森林这样的高级模型的基础套件中，Hellsemble 通常也能达到或超过它们的性能。

这些结果突显了 Hellsemble 作为实用且可扩展的集成方法的潜力。尽管当前实现已经取得了有希望的结果，但仍存在改进的空间——特别是在提高路由器的准确性以及探索定义和管理难度圈的替代策略方面。

总之, Hellsemble 在集成学习中提供了一个有吸引力的新方向, 通过模块化和可解释的架构平衡了简洁性、效率和预测性能。

Acknowledgments. 此项目的工作得到了华沙理工大学的部分资助, 作为倡议: 研究型大学 (*IDUB*) 计划。计划的一部分, 通过一项针对科学俱乐部的拨款计划进行, 时间为 2024 – 2025 年。

参考文献

1. Bekku, H., Kume, T., Tsuge, A., Nakazawa, J.: A stable and efficient dynamic ensemble method for pothole detection. *Pervasive and Mobile Computing* **104**, 101973 (2024)
2. Bischl, B., Casalicchio, G., Feurer, M., Gijsbers, P., Hutter, F., Lang, M., Manton, R.G., van Rijn, J.N., Vanschoren, J.: Openml benchmarking suites (2021), <https://arxiv.org/abs/1708.03731>
3. Breiman, L.: Random forests. *Machine learning* **45**, 5–32 (2001)
4. Caruana, R., Munson, A., Niculescu-Mizil, A.: Getting the most out of ensemble selection. In: Sixth International Conference on Data Mining (ICDM'06). pp. 828–833. IEEE (2006)
5. Caruana, R., Niculescu-Mizil, A., Crew, G., Ksikes, A.: Ensemble selection from libraries of models. In: Proceedings of the twenty-first international conference on Machine learning. p. 18 (2004)
6. Cavalin, P.R., Sabourin, R., Suen, C.Y.: Dynamic selection approaches for multiple classifier systems. *Neural computing and applications* **22**, 673–688 (2013)
7. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 785 – 794. KDD '16, ACM (Aug 2016). <https://doi.org/10.1145/2939672.2939785>, <http://dx.doi.org/10.1145/2939672.2939785>
8. Cruz, R.M., Sabourin, R., Cavalcanti, G.D., Ren, T.I.: Meta-des: A dynamic ensemble selection framework using meta-learning. *Pattern recognition* **48**(5), 1925–1935 (2015)
9. Cutler, A., Cutler, D.R., Stevens, J.R.: Random forests. *Ensemble machine learning: Methods and applications* pp. 157–175 (2012)
10. Dietterich, T.G.: Ensemble methods in machine learning. In: International workshop on multiple classifier systems. pp. 1–15. Springer (2000)

11. Ding, D., Mallick, A., Wang, C., Sim, R., Mukherjee, S., Ruhle, V., Lakshmanan, L.V., Awadallah, A.H.: Hybrid llm: Cost-efficient and quality-aware query routing. arXiv preprint arXiv:2404.14618 (2024)
12. Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., Smola, A.: Autogluon-tabular: Robust and accurate automl for structured data (2020), <https://arxiv.org/abs/2003.06505>
13. Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., Smola, A.: Autogluon-tabular: Robust and accurate automl for structured data (2020), <https://arxiv.org/abs/2003.06505>
14. Feurer, M., Eggenberger, K., Falkner, S., Lindauer, M., Hutter, F.: Auto-sklearn 2.0: Hands-free automl via meta-learning. *Journal of Machine Learning Research* **23**(261), 1–61 (2022)
15. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* **55**(1), 119–139 (1997)
16. Giacinto, G., Roli, F.: Dynamic classifier selection based on multiple classifier behaviour. *Pattern Recognition* **34**(9), 1879–1881 (2001)
17. Ko, A.H., Sabourin, R., Britto Jr, A.S.: From dynamic classifier selection to dynamic ensemble selection. *Pattern recognition* **41**(5), 1718–1731 (2008)
18. McElfresh, D., Khandagale, S., Valverde, J., C, V.P., Feuer, B., Hegde, C., Ramakrishnan, G., Goldblum, M., White, C.: When do neural nets outperform boosted trees on tabular data? (2024), <https://arxiv.org/abs/2305.02997>
19. McGill, M., Perona, P.: Deciding how to decide: Dynamic routing in artificial neural networks. In: *International Conference on Machine Learning*. pp. 2363–2372. PMLR (2017)
20. Olson, R.S., Bartley, N., Urbanowicz, R.J., Moore, J.H.: Evaluation of a tree-based pipeline optimization tool for automating data science (2016), <https://arxiv.org/abs/1603.06212>
21. Sesmero, M.P., Ledezma, A.I., Sanchis, A.: Generating ensembles of heterogeneous classifiers using stacked generalization. *Wiley interdisciplinary reviews: data mining and knowledge discovery* **5**(1), 21–34 (2015)
22. Woods, K., Kegelmeyer, W.P., Bowyer, K.: Combination of multiple classifiers using local accuracy estimates. *IEEE transactions on pattern analysis and machine intelligence* **19**(4), 405–410 (1997)