Multi-Level Stacking

Fabiana Coutinho Boldrin[1] **, Adriano Henrique Cantão**[1] **,**
Renato Tinós[1] **, José Augusto Baranauskas**[1]

[1]Departamento de Computação e Matemática,
Faculdade de Filosofia Ciências e Letras de Ribeirão Preto, Universidade de São Paulo.
Avenida dos Bandeirantes, 3900, 14040-901, Ribeirao Preto, SP, Brasil

{fabiana.boldrin, cantao}@usp.br, rtinos@ffclrp.usp.br, augusto@usp.br

Abstract. *Stacking is one of the algorithms that combine the results of different classifiers generated using the same training set. To explore some aspects of the stacking algorithm, such as the number of levels (layers) of learning, the number of classifiers per level, and the algorithms used, multi-level stacking is proposed. For this work experiments were done using three distinct types of inductors for distinct datasets and with two levels of learning.*

Resumo. *Stacking é um dos algoritmos que combina os resultados de diferentes classificadores que foram gerados utilizando o mesmo conjunto de treinamento. Com objetivo de explorar alguns aspectos com relação ao algoritmo de stacking como o número de levels (camadas) de aprendizado, o número de classificadores por level e os algoritmos de utilizados, foi proposto o multi-level stacking. Para este trabalho foram feitos experimentos utilizando três tipos diferentes de indutores para diferentes datasets com dois levels de aprendizado.*

1. Introdução

O meta-aprendizado ou *learning to learn* foi criado a partir da necessidade de combinar e reaproveitar conhecimentos já antes adquiridos em tarefas de temas e campos específicos do conhecimento [Thrun and Pratt 1998, Giraud-Carrier 2004]. Existem na literatura inúmeras definições para meta-aprendizado quando dentro da ciência da computação, como a automatização do processo de escolha dos muitos componentes de uma cadeia de aprendizado de máquina como algoritmos de aprendizado, parâmetros usados, exemplos de treinamento, etc [Anderson and Oates 2007, Dyrmishi et al. 2019, Karmaker Santu et al. 2021]. Um dos tipos de algoritmos que se enquadram como meta-aprendizado é a combinação de classificadores, conhecido como *ensemble* [Dietterich 2000, Ganaie et al. 2022]. Essa técnica parte do pressuposto que um *ensemble* geralmente tem melhor desempenho do que um único classificador que o compõe.

Nessa estratégia, inicialmente é efetuada uma amostragem do conjunto de exemplos de treinamento para induzir diferentes classificadores e então combina-se a saída destes classificadores por meio de algum mecanismo de votação ou ponderação. É possível classificar técnicas de *ensemble* como homogêneas ou heterogêneas. As homogêneas utilizam um único tipo de algoritmo de aprendizado, em geral adicionando variabilidade com técnicas de amostragem enquanto heterogêneas utilizam diferentes tipos de algoritmos de aprendizado.

Esta pesquisa se concentra em um tipo específico de *ensemble* denominado *stacking* [Wolpert 1992] aplicado a problemas de classificação. Em definição simplificada, *stacking* é um método que faz uso de um meta-classificador para combinar as saídas de diferentes classificadores. O *stacking* proposto é do tipo homogêneo e tem como objetivo comparar o *stacking*, aplicado a múltiplos níveis de aprendizado, com algoritmos de classificação e avaliar se existe algum ganho de desempenho com sua utilização. Neste estudo, o objetivo é verificar o desempenho de *multi-level stacking* homogêneo aplicado a diferentes amostragens do mesmo conjunto de exemplos em relação a apenas um único classificador, variando-se o número de níveis de aprendizado.

A organização desta pesquisa encontra-se da seguinte forma: Na Seção 2 é feita uma contextualização do temas abordados nesta pesquisa com referências da literatura. Na Seção 3 são explicados e definidos conceitos sobre o algoritmo de *stacking* necessários para o entendimento do método proposto. Na Seção 4 é descrita a proposta e a metodologia para geração de um *multi-level staking* de classificadores. Na Seção 5 encontra-se descrita a configuração dos experimentos para avaliação do algoritmo proposto. Na Seção 6 são apresentados e discutidos os resultados obtidos . Por fim, na Seção 7, são apresentados os indutores com melhores desempenhos dentro do escopo apresentado e as principais contribuições deste estudo.

2. Trabalhos Relacionados

Quando um algoritmo de aprendizado de máquina ou o modelo induzido por um algoritmo de aprendizado é analisado, tal análise está ocorrendo em um *meta-nível* [Grabczewski 2014]. Como mencionado, existem inúmeras definições para o termo meta-aprendizado e é valido ressaltar que, nesta pesquisa, o termo se refere a toda forma de aprendizado a partir de informações sobre processos e modelos de aprendizado.

Na literatura sobre meta-aprendizado é possível encontrar trabalhos como [Wang 2020, Guldogan 2022], que reforçam a relevância que o tema segue tendo em pesquisas em inteligência artificial. No primeiro trabalho a autora tem como objetivo traçar uma narrativa paralela entre linhas de pesquisa de meta-aprendizado para o campo da neurociência e para a inteligência artificial tendo como um dos métodos relevantes o de identificação automática de hiper-parâmetros utilizados em algoritmos de aprendizado [Xu et al. 2018, Jaderberg et al. 2017]. Já no segundo trabalho, os autores utilizam um *stacking* para predição, a partir de imagens de tomografia, de estágio da infecção pelo vírus SARS-CoV-2 (COVID-19) em pacientes. No primeiro *level* de aprendizado são usados modelos de *transfer learning* (TL) [Zhuang et al. 2020] e no segundo *level* um XGBoost [Chen and Guestrin 2016].

O método de meta-aprendizado *stacking*, também presente na literatura como *stacked generalization*, que esta pesquisa explora, é uma forma de *ensemble* muito presente na literatura [Caffé et al. 2012, Massaoudi et al. 2021, Wang et al. 2021]. Segundo o trabalho original, o *stacking* foi criado como forma de minimizar o erro associado a um ou mais classificadores partindo do pressuposto de que um *ensemble* geralmente é mais preciso que qualquer uma das hipóteses individuais contidas nele [Wolpert 1992]; o autor ainda compara o *stacking* com métodos clássicos de amostragem como *cross-validation*, *holdout* e *leave-one-out*, sendo possível identificar que diferentemente destes métodos que estimam o desempenho de um único algoritmo de aprendizado, o *stacking* busca

combinar algoritmos distintos. Na Seção 3 os conceitos relacionados a este algoritmo são explicados por conta de sua importância para entendimento do método de *multi-level stacking*.

Segundo o estudo de [T. Nguyen and Liew 2016], considerando apenas os *levels* em que ocorre meta-aprendizado, existem diferentes formas de combinar as saídas geradas por classificadores. Esses métodos se subdividem em dois grupos: métodos de combinação fixa e métodos dependentes de treino.

- **Métodos de combinação fixa:** Os métodos de combinação fixa não levam em consideração a saída gerada por classificadores no *level-0* de treinamento. Esses têm como vantagem a não necessidade de treino e, assim, podem ser gerados de forma mais rápida e usando menos processamento. Alguns exemplos são soma, produto, voto, média e mediana.
- **Métodos dependentes de treino:** Em contrapartida, métodos mais sofisticados e dependentes de treino são estado da arte em combinação de classificadores como *Multi Response Linear Regression* (MLR) [Witten and Ting 1997], *Decision Template* (DT) [Ludmila I. Kuncheva and Duin 2001], *Stacking Correspondence Analysis Nearest Neighbor* (SCANN) [Merz 1999] e *Variational Inference for Multivariate Gaussian Distribution* (VIG) [T. Nguyen and Liew 2016]. MLR assume que cada classificador atribui um peso diferente as classes. O método pode ser considerado uma adaptação dos mínimos quadrados de uma regressão linear e é baseado nas M combinações lineares das classes posteriores e nos pesos destas classes. Diferentemente do MLR, o DT gera uma matriz chamada *decision profile* (DP) contendo os resultados dos diferentes classificadores. O DT da classe C_1 é obtido a partir da média dos DP dos elementos de treino rotulados como C_1. Mais uma forma de combinação de classificadores é proposta em [Merz 1999], onde aplica-se *stacking* e *correspondence analysis* para modelar a relação entre os exemplos de treinamento e a classe predita. Depois um método de vizinhos mais próximos é utilizado na representação resultante para classificar novos exemplos. Mais recentemente foi proposto em [T. Nguyen and Liew 2016] o VIG, baseado no modelo Bayesiano aproximando a densidade da distribuição dos dados preditos. É utilizada *Variational Inference* para estimar a densidade de distribuição Gaussiana.

Além de diferentes formas para combinação de modelos, existe na literatura também diferentes formas de amostragem para ganho de desempenho na aplicação do *stacking*. Como mostrado no trabalho [Witten and Ting 1997], duas formas comprovadamente eficientes de amostragem são *bag-stacking* e *dag-stacking*. *Bag-stacking* consiste na aplicação do método de *bootstrap*, seleção aleatória de exemplos com reposição no conjunto de treinamento, enquanto a segunda realiza também uma amostragem aleatória de exemplos no conjunto de treinamento, porém, sem reposição.

3. O Algoritmo *Stacking*

O algoritmo *stacking* utiliza o conceito de *level* (nível) de aprendizado. Inicialmente, este conceito será fornecido, considerando apenas dois *levels* de aprendizado (*levels* 0 e 1), representado pelo Algoritmo 1. Em seguida, este conceito será estendido para mais *levels* de aprendizado, de acordo com o Algoritmo 2 proposto neste estudo.

Algorithm 1 *Stacking* - adaptado de [Aggarwal 2014, Wolpert 1992] pelos autores

Require: *dataset* $D^0 = \{x_i, y_i\}_{i=1}^{n} (x_i \in R^n, y_i \in Y)$
Require: Z, número de algoritmos no *level* zero de aprendizado
Require: $\{L_z\}_{z=1}^{Z}$ algoritmos de aprendizado de *level* 0 que geram os Z classificadores a partir de D^0
Require: M, algoritmo de aprendizado de *level* 1
Ensure: O meta-classificador h^*

1: **function** Stacking(D^0,Z, $\{L_z\}_{z=1}^{Z}$, M)
2: **for** $z \leftarrow 1, \ldots, Z$ **do** ▷ geração dos classificadores de *level* 0
3: $h_z^0(\cdot) \leftarrow L_z(D^0)$
4: **end for**
5: $h^0(\cdot) \leftarrow (h_1^0(\cdot), h_2^0(\cdot), \ldots, h_Z^0(\cdot))$
6: $D^1 \leftarrow \emptyset$ ▷ geração do dataset de *level* 1
7: **for** $i \leftarrow 1, \ldots, n$ **do**
8: $D_i^0 \leftarrow D^0 - \{(x_i, y_i)\}$ ▷ leave one out
9: **for** $z \leftarrow 1, \ldots, Z$ **do**
10: $h_{iz}(\cdot) \leftarrow L_z(D_i^0)$
11: **end for**
12: $D^1 \leftarrow D^1 \cup \{(h_{i1}(x_i), h_{i2}(x_i), \ldots, h_{iZ}(x_i), y_i)\}$
13: **end for**
14: $h^1(\cdot) \leftarrow M(D^1)$ ▷ geração do classificador de *level* 1
15: $h^*(\cdot) \leftarrow h^1(h^0(\cdot))$ ▷ meta-classificador final (*levels* 0 e 1)
16: **return** h^*

Seja $\{L_z\}_{z=1}^{Z}$ um conjunto de Z algoritmos de aprendizado (de *level* 0), M um algoritmo de aprendizado (de *level* 1). A partir do conjunto original de exemplos D^0 são gerados z classificadores $\{h_z^0(\cdot) = L_z(D^0)\}_{z=1}^{Z}$ que compõem o classificador de *level* 0, denotado por $h^0(\cdot) = (h_1^0(\cdot), h_2^0(\cdot), \ldots, h_Z^0(\cdot))$; este processo é representado em alto nível nas linhas 2-5 do Algoritmo 1. Assim, por definição, o *level*-0 de aprendizado é constituído pelos classificadores treinados no conjunto original de exemplos D^0. Em seguida, é efetuada a geração do conjunto de treinamento de *level* 1 (D^1) nas linhas 6-13 do Algoritmo 1 que permite induzir o classificador de *level* 1 (h^1) na linha 14. Originalmente, a estratégia escolhida para a geração do conjunto de treinamento de *level* 1 é *leave-one-out*, na qual um único exemplo é removido do conjunto D^0 a cada iteração [Wolpert 1992]. Desta forma, na iteração i é criado o conjunto $D_i^0 = D^0 - \{(x_i, y_i)\}$; então, são induzidos novos Z classificadores a partir de D_i^0 que são aplicados ao exemplo que ficou de fora do treinamento, x_i. Este processo se repete n vezes, uma para cada exemplo no conjunto de treinamento D^0, definindo-se o conjunto de treinamento de *level* 1, denotado por D^1. Este processo pode ser visualizado para um único exemplo (x_i, y_i) e dois algoritmos de aprendizado L_1 e L_2 na Figura 1. A partir do conjunto D^1 é gerado o classificador de *level* 1, fazendo-se $h^1(\cdot) = M(D^1)$ e o meta-classificador final então é definido como sendo $h^*(\cdot) = h^1(h^0(\cdot))$, respectivamente definidos nas linhas 14 e 15 do Algoritmo 1.

4. Multi-Level Stacking

É importante salientar os seguintes aspectos com relação ao Algoritmo 1: (i) para a criação do conjunto de treinamento de *level* 1 são necessárias $Z \times n$ execuções dos algoritmos de aprendizado, o que é custoso computacionalmente para grandes conjuntos de exemplos; (ii) são necessários $\{L_z\}_{z=1}^{Z}$ algoritmos de aprendizado distintos ou com configuração de seus parâmetros de forma distinta (para que possam produzir classificadores diferentes nos dois *levels* de aprendizado, além do meta-algoritmo M. Mesmo considerando que

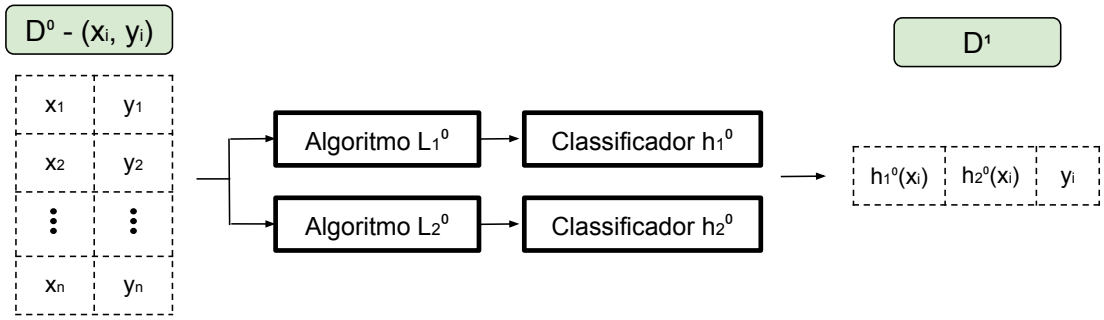


Figura 1. Combinação de classificadores em formato matricial.

o número de algoritmos não seja grande, tal escolha de diferentes algoritmos torna-se complexa na extensão para mais níveis de aprendizado.

Neste sentido, no Algoritmo 2, estes dois aspectos foram modificados da seguinte forma: (i) uso de validação cruzada com K partições em vez de *leave-one-out* (mesmo considerando que *leave-one-out* é uma validação cruzada com o número de partições (*folds*) igual ao número de exemplos de treinamento, $K = n$, há um ganho em termos de tempo quando $K \ll n$). Assim, em cada execução da validação cruzada são criados dois conjuntos D_T e D_t, representando os conjuntos de treinamento e de teste, respectivamente e (ii) um único algoritmo de aprendizado é utilizado em todos os níveis (*stacking* homogêneo), o que permite a execução de vários níveis nos experimentos reportados na Seção 5. Para tanto, é necessário que os classificadores gerados, como no algoritmo original, sejam diferentes entre si. Isto foi obtido, neste estudo, utilizando-se amostragem *bootstrap* [Beriman 1996, Mosavi 2021] no conjunto de treinamento D_T; como no algoritmo original, o conjunto de treinamento de nível seguinte utiliza apenas os exemplos em D_t em sua composição.

Utilizando o exemplo de seção anterior e partindo do momento em que foi gerado um conjunto de exemplos de *level* 1 (D^1), para o próximo *level* de aprendizado são gerados dois conjuntos distintos (D^1_T, D^1_t), respectivamente conjunto de exemplos de treinamento e conjunto de exemplos de validação. É então criado o conjunto B, composto dos conjuntos de exemplos gerados a partir da amostragem *bootstrap* aplicada ao conjunto de treinamento. Cada conjunto de exemplos é então utilizado pelo algoritmo de aprendizado L para a indução do conjunto $\{h^1_z(\cdot) = L(B)\}^Z_{z=1}$ denotado por $h^1(\cdot) = (h^1_1(\cdot), h^1_2(\cdot), \ldots, h^1_Z(\cdot))$. O conjunto de exemplos de *level* 2 (D^2) é resultado da classificação do conjunto de exemplos de validação por cada um dos indutores de $h^1(\cdot)$. Desta forma $D^2 = \{h^1_z(D^1_{tz})\}^Z_{z=1}$. A representação do *multi-level stacking* para P *levels* de aprendizado encontra-se sob a forma do Algoritmo 2 explicado a seguir.

De acordo com o Algoritmo 2 recebe-se como entrada um conjunto de n exemplos rotulados $D^0 = \{(\mathbf{x}_i, y_i), i = 1, 2, \ldots, n\}$, o valor P que corresponde ao número de *levels* do *multi-level stacking*, o valor Z que corresponde ao número de classificadores que irão compor cada *level* de aprendizado, um algorítimo de aprendizado L que gerará diferentes classificadores h e por fim K que corresponde ao número de partições que a estratégia de amostragem irá utilizar. A quesito de notação, vale observar que os números sobrescritos no contexto desta pesquisa representam o *level* ao qual a variável pertence e os números

Algorithm 2 Multi-level Stacking

Require: D^0, *dataset* com n exemplos classificados $\{x_i, y_i\}_{i=1}^{n} (x_i \in R^n, y_i \in Y)$
Require: P, número de *levels* de *stacking* (meta-aprendizado), onde $P \geq 1$
Require: L, um algoritmo de aprendizado que será utilizado nos Z *datasets* gerados a partir de D^0
Require: Z, número de classificadores gerados em cada *level* de aprendizado
Require: K, número de partições utilizadas para método de amostragem para criação de conjuntos de treinamento e validação
Ensure: O meta-classificador h^*

1: **function** multiLevelStacking(D^0, P, L, Z, K)
2: **for** $p \leftarrow 0, 1, \ldots, P-1$ **do**
3: **for** $z \leftarrow 1, 2, \ldots, Z$ **do** ▷ geração dos classificadores de *level* p
4: $B \leftarrow$ bootstrap(D^p)
5: $h_z^p(\cdot) \leftarrow L(B)$
6: **end for**
7: $h^p(\cdot) \leftarrow (h_1^p(\cdot), h_2^p(\cdot), \ldots, h_Z^p(\cdot))$ ▷ classificador de *level* p
8: **if** $p < P-1$ **then** ▷ Última iteração, não gerar D^{p+1}
9: $D^{p+1} \leftarrow \emptyset$ ▷ geração do dataset de *level* $p+1$
10: **for** $k \leftarrow 1, 2, \ldots, K$ **do** ▷ validação cruzada com K partições
11: $(D_T^p, D_t^p) \leftarrow$ partitionFold(D^p, k) ▷ treinamento e validação
12: **for** $z \leftarrow 1, 2, \ldots, Z$ **do**
13: $B \leftarrow$ bootstrap(D_T^p) ▷ amostra bootstrap do conjunto de treinamento
14: $h_{kz}(\cdot) \leftarrow L(B)$
15: **end for**
16: $D^{p+1} \leftarrow D^{p+1} \cup \{(h_{k1}(D_t^p), h_{k2}(D_t^p), \ldots, h_{kZ}(D_t^p), y_{D_t^p})\}$
17: **end for**
18: **end if**
19: **end for**
20: $h^*(\cdot) \leftarrow h^{P-1}(h^{P-2}(\ldots h^1(h^0(\cdot))\ldots))$ ▷ meta-classificador final
21: **return** h^*
22: **end function**

subscritos representam a iteração atual dos processos iterativos do Algoritmo. O processo *bootstrap* corresponde ao método de amostragem aplicado ao conjunto de exemplos de entrada e o processo *partitionDataset* corresponde ao processo de separar o conjunto de exemplos entre conjuntos de trenamento (determinado pela letra T) e de validação (determinado pela letra t).

A lógica do Algoritmo 2 se dá da seguinte maneira: recebendo as entradas anteriormente descritas, tem-se como primeiro nível de iteração os P *levels* de aprendizado descrito na linha 3. Dentro de cada iteração p tem-se o segundo nível de iteração das K partições que para cada partição irá separar o conjunto D^p entre os conjuntos de trenamento D_T^p e validação D_t^p (linha 11). Em um terceiro nível de iteração dos Z algoritmos de aprendizado, para cada z é utilizado o método de *bootstrap* no conjunto de trenamento D_T^p para a criação do conjunto B representado na linhas 12 e 13 do algoritmo. O conjunto de exemplos de trenamento B e o algoritmo de aprendizado L são então utilizados para a indução de um classificador h_{kz} representado na linha 14. Tendo então para este k o conjunto de classificadores $\{(h_{kz}), z = 1, 2, \ldots, Z\}$ é possível então construir o conjunto de exemplos $D^{(p+1)}$ a partir da classificação do conjunto de validação D_t^p e cada um dos classificadores do conjunto de Z classificadores. Este processo se repete até o *level* P do Algoritmo quando tem-se como resultado o conjunto de exemplos D^P. A partir do conjunto de exemplos D^P e do algoritmo de aprendizado L é gerado o meta-classificador

$h^*(\cdot)$ no algoritmo acima, representado na linha 20 do algoritimo.

O processo do Algoritmo 2 descrito acima tem como resultado um meta-algoritmo h^* induzido a partir de dados em um *level* P de aprendizado. Desta forma, para que um novo exemplo q seja classificado é necessário que este também esteja no mesmo *level* P de aprendizado. Para isso, durante o processo de indução de h^* também é induzido um conjunto de classificadores $h_Z^P = \{(h_z^p), z = 1, 2, \ldots, Z, p = 0, 1, \ldots, P-1\}$ (das linhas 12 a 15 do Algoritmo 2). O conjunto h_Z^P tem como primeiro nível de iteração os P *levels* de aprendizado e dentro de cada iteração p tem-se o segundo nível de iteração dos Z algoritmos de aprendizado. Para cada (p, z) é utilizado o método de amostragem *bootstrap* no conjunto D^p para criar o conjunto B. O conjunto de exemplos B então é utilizado em conjunto com o algoritmo L para a indução do classificador h_z^p (classificador z de nível p).

Para a classificação de um novo exemplo q^0 tem-se como primeiro nível de iteração os P *levels* de aprendizado e como segundo nível os Z algoritmos de aprendizado. Para cada valor (p, z) é utilizado o classificador h_z^p para classificar o exemplo q^p.

5. Metodologia Experimental

Para o processo de experimentação e avaliação dos resultados, os algoritmos de aprendizado escolhidos para cada *level* irão se repetir nos $P-1$ *levels* de aprendizado. Assim, todos os $P-1$ *levels* terão a mesma configuração. O ultimo *level* terá uma configuração diferente, sendo composto por apenas um meta-algoritmo, porém ainda utilizando o mesmo algoritmo de aprendizado que os *levels* anteriores.

Os algoritmos foram selecionados de [Fernández-Delgado et al. 2014]: (i) Árvore de decisão [Breiman et al. 1984, Quinlan 1986]; (ii) *Naive bayes* Gaussiano [Bayes 1763] e (iii) *K-nearest neighbours* (KNN) [Aha 1997]. Os algoritmos escolhidos foram utilizados com seus parâmetros *default* implementados na biblioteca `scikit-learn` [Pedregosa et al. 2011]. No caso do *K-nearest neighbours* vale salientar que sua implementação *default* tem parâmetro $K = 5$. Como métrica de avaliação do desempenho dos modelos foi utilizada área debaixo da curva ROC (AUC) [Fawcett 2006].

Os modelos analisados nos experimentos são: um único classificador (C), um *stacking* de classificadores (SC), um *multi-level stacking* com Z classificadores e P *levels* (MPC), para cada um dos algoritmos escolhidos

Para os experimentos utilizando os algoritmos *stacking* e *multi-level stacking*, foram utilizados $Z = \{5, 500, 1000\}$. Para o algoritmo *multi-level stacking* o número de *levels* utilizado no experimento foi $P = 2$. Desta forma, já é possível identificar se existe ganho em aplicar uma camada a mais de aprendizado. O valor AUC e seu desvio padrão foram obtidos a partir de uma validação cruzada utilizando de 10 *folds* em cinco *datasets* de classificação. A descrição dos *datasets* utilizados se encontra na Tabela 1.

6. Resultados

Nas Tabelas 2, 3 e 4 são apresentados os resultados dos valores AUC (média e desvio padrão) para cada um dos três algoritmos escolhidos.

Tabela 1. Resumo dos conjuntos de exemplos utilizados, no qual n indica o número de exemplos, m o número de atributos, c o número de classes, $c_1\%$ e c_2% distribuição de classes.

Dataset	n	m	c	$c_1\%$	$c_2\%$
breast-cancer-winsconsin	569	32	2	70.28	29.72
adult	48842	14	2	75.92	24.08
diabetes	768	8	2	65.10	34.90
banknote	1372	17	2	88.47	11.52
ionosphere	351	34	2	64.10	35.90

Tabela 2. Valores AUC (média e desvio padrão) obtidos a partir do algoritmo de árvore de decisão aplicado de forma simples (C), em um Stacking de árvore de decisão com diferentes valores de Z (SC$_Z$) e em um multi-level stacking com diferentes valores de Z (MPC$_Z$).

Árvore de Decisão							
Dataset	C	SC_5	SC_{500}	SC_{1000}	MPC_5	MPC_{500}	MPC_{1000}
breast-cancer-winsconsin	**0,94±0,03**	0,93±0,02	0,93±0,02	0,93±0,03	0,91±0,04	0,92±0,05	0,50±0,00
ionosphere	0,85±0,07	0,85±0,04	0,82±0,04	**0,86±0,04**	0,85±0,08	0,84±0,08	0,50±0,00
diabetes	**0,67±0,07**	0,65±0,08	0,66±0,06	0,58±0,07	0,61±0,06	0,62±0,05	0,64±0,02
adult	**0,75±0,01**	0,73±0,01	0,71±0,00	0,72±0,00	0,69±0,04	0,70±0,04	0,70±0,05
banknote	**0,67±0,03**	0,59±0,03	0,67±0,05	0,64±0,04	0,57±0,04	0,65±0,05	0,67±0,04
AUC médio	**0,77±0,12**	0,75±0,14	0,76±0,11	0,74±0,15	0,73±0,15	0,74±0,13	0,60±0,09

Na Tabela 2, referente aos resultados obtidos utilizando árvores de decisão, tem-se um melhor resultado médio entre diferentes *datasets* utilizando um classificador simples. Dentre os métodos de *ensemble*, o método de *stacking* com $z = 500$ é o que apresenta o melhor resultado. O resultado utilizando o método de *multi-level stacking* com 500 indutores em cada *level* (MPC_{500}) se aproxima de SC_{500} sendo o desempenho do SC_{500} = 0,76 e o desempenho do MPC_{500} = 0,74. Também é possível observar que dentro do *multi-level stacking* o que contem o maior numero de indutores por camada (MPC_{1000}) é o que apresenta pior desempenho (0,60). Analisando pela perspectiva dos valores médios de AUC obtidos dentro de cada *dataset*, é possível identificar que o comportamento esperado de um ganho consistente de desempenho ao passo que se adicionam classificadores nos *levels* de aprendizado só ocorre nos *datasets diabetes* e *banknote* para o *multi-level stacking*.

Em relação a Tabela 3, a qual apresenta os resultados do indutor probabilístico *naive bayes* gaussiano, tem-se um comportamento de AUC médio entre os diferentes métodos bem parecido. O melhor desempenho médio considerando todos os *datasets* foi do método de *stacking* com 1000 indutores (SC_{1000}) com AUC médio de 0,80. Mesmo assim, esse desempenho não ficou muito acima dos outros 5 métodos com o mesmo AUC médio. É possível ainda identificar um aumento no desempenho médio do método de *stacking* a medida que aumenta-se o número de indutores utilizados, com 5 indutores um desempenho de 0,77, com 500 indutores um desempenho de 0,78 e com 1000 indutores um desempenho de 0,80. É possível ver este mesmo comportamento do ponto de vista de *datasets* individuais, por exemplo o desempenho do método de *multi-level stacking* para o *dataset adult*.

Tabela 3. Valores AUC (média e desvio padrão) obtidos a partir do algoritmo *naive bayes* gaussiano aplicado de forma simples (C), em um Stacking de classificadores *naive bayes* com diferentes valores de Z (SC_Z) e em um multi-level stacking com diferentes valores de Z (MPC_Z).

Naive Bayes Gaussiano							
Dataset	C	SC_5	SC_{500}	SC_{1000}	MPC_5	MPC_{500}	MPC_{1000}
breast-cancer-winsconsin	**0,94±0,03**	0,93±0,02	**0,94±0,03**	0,93±0,03	0,92±0,03	0,91±0,03	0,91±0,04
ionosphere	0,87±0,06	0,81±0,03	0,81±0,05	0,86±0,04	**0,88±0,06**	0,84±0,05	0,84±0,06
diabetes	0,72±0,04	0,71±0,05	0,72±0,04	**0,76±0,03**	0,72±0,05	0,72±0,04	0,72±0,03
adult	0,69±0,01	0,71±0,01	0,72±0,01	0,72±0,01	0,71±0,01	0,72±0,01	**0,73±0,01**
banknote	0,68±0,04	0,70±0,04	**0,71±0,04**	**0,71±0,04**	0,69±0,04	**0,71±0,04**	**0,71±0,04**
AUC médio	0,78±0,12	0,77±0,10	0,78±0,10	**0,80±0,09**	0,78±0,11	0,78±0,09	0,78±0,09

Na Tabela 4 são apresentados os resultados para o indutor *K Nearest Neighbours*. O melhor desempenho médio, considerando todos os *datasets*, foi obtido pelo classificador simples (C). De modo geral, o indutor *Naive Bayes* Gaussiano apresentou em média o maior desempenho dentre os três indutores utilizados. Já o *K Nearest Neighbours* foi o indutor com o menor desempenho médio entre os três utilizados nesta pesquisa.

Tabela 4. Valores AUC (média e desvio padrão) obtidos a partir do algoritmo *k nearest neighbours* aplicado de forma simples (C), em um Stacking de classificadores *k nearest neighbours* com diferentes valores de Z (SC_Z) e em um Multi-level stacking com diferentes valores de Z (MPC_Z).

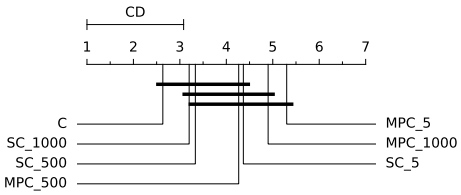
K Nearest Neighbours							
Dataset	C	SC_5	SC_{500}	SC_{1000}	MPC_5	MPC_{500}	MPC_{1000}
breast-cancer-winsconsin	**0,95±0,02**	0,91±0,04	0,93±0,02	0,91±0,04	0,92±0,04	0,92±0,02	0,50±0,00
ionosphere	0,80±0,06	**0,87±0,06**	0,85±0,04	0,86±0,07	0,85±0,08	**0,87±0,06**	0,50±0,00
diabetes	**0,68±0,05**	0,64±0,06	0,65±0,04	0,67±0,02	0,50±0,00	0,65±0,04	0,64±0,03
adult	**0,64±0,01**	0,59±0,01	0,62±0,00	0,62±0,00	0,54±0,04	0,59±0,01	0,60±0,01
banknote	**0,58±0,02**	0,55±0,02	0,55±0,01	0,55±0,04	0,50±0,01	0,54±0,01	0,53±0,02
AUC médio	**0,73±0,32**	0,71±0,16	0,72±0,16	0,72±0,16	0,66±0,21	0,71±0,17	0,55±0,06

As Figuras 2(a) e 2(b) apresentam o Diagrama de Diferença Crítica com o *ranking* médio de cada algoritmo. O teste da Figura 2(a) foi feito utilizando os vetores de valor de desempenho para todos os *datasets* e todos os algoritmos. Nele é possível ver um melhor desempenho do classificador simples (C) com relação aos *multi-level stacking* com 5 e 1000 indutores por *level* (MPC_5 e MPC_{1000}). Para além disso, é também possível ver uma desempenho melhor do *stacking* com 1000 indutores (SC_{1000}) do que o *multi-level stacking* com 5 indutores por *level* (MPC_5).

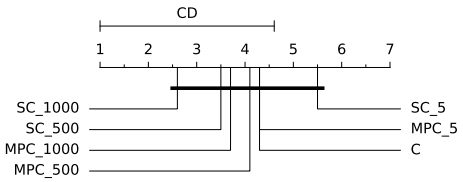
Analisando o Diagrama de Diferença Crítica, apresentado pela Figura 2(b), é possível notar que os métodos de *stacking* e *multi-level stacking* com 1000 e 500 indutores por *level* (SC_{1000}, SC_{500}, MPC_{1000} e MPC_{500}) apresentaram melhor desempenho do que o classificador simples (C) e os métodos *stacking* e *multi-level stacking* com 5 indutores por *level* (MPC_5 e SC_5).

7. Conclusão

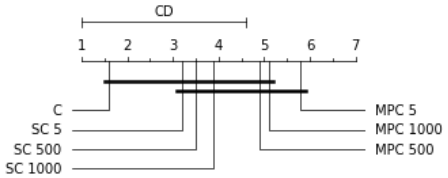
Nesta pesquisa foi apresentado um método de meta-aprendizado de múltiplos níveis que utiliza a combinação de classificadores de forma encadeada. A partir dos experimentos



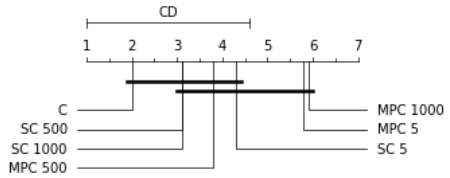
((a)) Diagrama referente aos resultados de todos os três algoritmos (Árvore de Decisão, *Naive Bayes* Gaussiano e K *Nearest Neghbours*.



((b)) Diagrama referente aos resultados do algoritmo *Naive Bayes* Gaussiano.



((c)) Diagrama referente aos resultados do algoritmo de Árvore de Decisão.



((d)) Diagrama referente aos resultados do algoritmo de K *Nearest Neghbours*.

Figura 2. Diagramas de Diferença Crítica usando o teste post-hoc de Bonferroni-Dunn para os diferentes indutores utilizados nos experimentos.

efetuados é possível concluir que um *multi-level stacking* homogêneo não teve melhor desempenho que um único classificador para os conjuntos de exemplos e algoritmos utilizados. É também possível concluir que, para alguns casos, existe certo ganho na adição de mais classificadores por *level* de aprendizado, por exemplo, o algoritmo *Naive Bayes* teve esse comportamento nos experimentos realizados. Porém, vale ressaltar que existe um limite de ganho ao adicionar mais classificadores por *level* de aprendizado dado que foi possível ver uma degradação do desempenho dos algoritmos quando Z passou de 500 para 1000. Em continuidade ao trabalho aqui apresentado, está em fase de planejamento a execução experimentos com mais conjuntos de exemplos e mais níveis de aprendizado.

Referências

Aggarwal, C. C. (2014). *Data Classification: Algorithms and Applications*. Chapman and Hall/CRC.

Aha, D. W. (1997). Lazy learning. *Artificial Intelligence Review*, 11:7–10.

Anderson, M. L. and Oates, T. (2007). A review of recent research in metareasoning and metalearning. *AI Magazine*, 28(1):12.

Bayes, T. (1763). Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s. *Philosophical transactions of the Royal Society of London*, (53):370–418.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth & Books, Pacific Grove, CA.

Caffé, M. I. R., Perez, P. S., and Baranauskas, J. A. (2012). Evaluation of stacking on biomedical data. *Journal of Health Informatics, ISSN 2175-4411*, 4(3):67–72. `http://www.jhi-sbis.saude.ws/ojs-jhi/index.php/jhi-sbis/article/view/181/119`.

Chen, T. and Guestrin, C. (2016). XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.

Dietterich, T. (2000). Janemachine learning research: Four current directions. *AI Magazine*, 18.

Dyrmishi, S., Elshawi, R., and Sakr, S. (2019). A decision support framework for automl systems: A meta-learning approach. In *2019 International Conference on Data Mining Workshops (ICDMW)*, pages 97–106.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recogn. Lett.*, 27(8):861–874.

Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.*, 15(1):3133–3181.

Ganaie, M., Hu, M., Malik, A., Tanveer, M., and Suganthan, P. (2022). Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115:105151.

Giraud-Carrier, C., V. R. . B. (2004). Introduction to the special issue on meta-learning. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning*, page 187–193.

Grabczewski, K. (2014). *Meta-Learning in Decision Tree Induction*, volume 498 of *Studies in Computational Intelligence*. Springer.

Guldogan, E. (2022). Artificial intelligence-assisted prediction of covid-19 status based on thorax ct scans using a proposed meta-learning strategy. *Machine Learning*, 38(3):1515–1521.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., Fernando, C., and Kavukcuoglu, K. (2017). Population based training of neural networks. *CoRR*, abs/1711.09846.

Karmaker Santu, S. K., Hassan, M. M., Smith, M. J., Xu, L., Zhai, C., and Veeramachaneni, K. (2021). Automl to date and beyond: Challenges and opportunities. *ACM Comput. Surv.*, 54(8).

Ludmila I. Kuncheva, J. C. B. and Duin, R. P. W. (2001). Decision templates for multiple classifier fusion: an experimental comparison. In *Pattern Recognition*, number 34, pages 299–314. https://doi.org/10.1016/S0031-3203(99)00223-X.

Massaoudi, M., Refaat, S. S., Chihi, I., Trabelsi, M., Oueslati, F. S., and Abu-Rub, H. (2021). A novel stacked generalization ensemble-based hybrid lgbm-xgb-mlp model for short-term load forecasting. *Energy*, 214:118874.

Merz, C. (1999). Using correspondence analysis to combine classifiers. In *Machine Learning*, number 36, page 33–58. https://link.springer.com/article/10.1023/A:1007559205422.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

T. Nguyen, Thi Thu Thuy Nguyen, X. C. P. and Liew, A. W.-C. (2016). A novel combining classifier method based on variational inference. In *Pattern Recognition*, number 49, page 198–212. https://doi.org/10.1016/j.patcog.2015.06.016.

Thrun, S. and Pratt, L. (1998). *Learning to Learn: Introduction and Overview*, pages 3–17. Springer US, Boston, MA.

Wang, J., Yuan, F., Chen, J., Wu, Q., Li, C., Yang, M., Sun, Y., and Zhang, G. (2021). Stackrec: Efficient training of very deep sequential recommender models by iterative stacking. *Proceedings of the 44th International ACM SIGIR conference on Research and Development in Information Retrieval*.

Wang, J. X. (2020). Meta-learning in natural and artificial intelligence. *CoRR*, abs/2011.13464.

Witten, I. H. and Ting, K. M. (1997). Stacking bagged and dagged models. In *In Proceedings of the Fourteenth International Conference on Machine Learning (ICML '97). Morgan Kaufmann Publishers Inc.*, page 367–375, San Francisco, CA, USA. https://pdfs.semanticscholar.org/ad52/dac8f267c8c75f30ac5b0c6c6bc980217285.pdf.

Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, 5(2):241–259.

Xu, Z., van Hasselt, H. P., and Silver, D. (2018). Meta-gradient reinforcement learning. 31.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2020). A comprehensive survey on transfer learning.