Estudo de Técnicas de *Ensemble* para Classificação de Dados

Lucas Fidelis Pereira[1] **& Rafael Geraldeli Rossi**[1]

[1] Universidade Federal de Mato Grosso do Sul (UFMS)
Av. Ranulpho Marques Leal, 3484 / CEP 79613-000

{lucas.fidelis, rafael.g.rossi}@ufms.br

Resumo. *A tarefa de classificação automática de dados se tornou indispensável devido ao valor elevado de dados que os usuários e as empresas geram atualmente. Com isso, uma melhor performance de classificação de dados se torna essencial para a extração de informação e conhecimento. Técnicas, que consistem em algoritmos de aprendizado de máquina que utilizam mais de um modelo classificador para classificar determinado objeto de ensemble, podem ser utilizadas para se obter uma melhor performance de classificação em relação aos algoritmos que utilizam um único modelo. Porém, existe uma variedade de técnicas que podem ser utilizadas para combinar classificadores. Desta forma, o objetivo deste trabalho de conclusão de curso é realizar um estudo sobre diferentes técnicas de ensemble, Bagging, Boosting e Stacking, verificando a sua performance de classificação para diferentes tipos de dados quando comparadas aos algoritmos de aprendizado com modelo único comumente utilizados na tarefa de classificação automática de dados. Ao utilizar os métodos ensemble notou-se a melhora na performance de classificação na maioria das base de dados utilizadas. O Stacking foi o algoritmo que obteve a melhor performance no maior número de execuções. Também é importante ressaltar a presença de algoritmos de aprendizado único na relação dos melhores resultados.*

1. Introdução

De acordo com Tan et al. (2019), os avanços rápidos nas tecnologias de armazenamento permitem que as organizações acumulem uma grande quantidade de dados. No entanto, o ato de extrair informações úteis de grandes bases de dados se mostra um grande desafio, uma vez que tais bases possuem grandes volumes, variedade e padrões ocultos em seus dados. Tendo isso em mente, a utilização de métodos de aprendizado de máquina para organizar, gerenciar e extrair conhecimento de tais dados de maneira automática se mostra cada vez mais necessária.

A classificação de dados se mostra importante uma vez que busca aprender a relação entre um conjunto de características e determinada variável de interesse, de modo que torne possível associar tais características a uma variável e assim prever o valor da variável de interesse para novos exemplos (Aggarwal, 2014). Logo a utilização de métodos para a classificação de maneira automática se torna uma ferramenta essencial para o processamento e extração de conhecimento de grandes bases de dados, sejam dados textuais, multimídia, de redes sociais, biológicos ou ainda médicos.

Aggarwal (2014) indica que erros na classificação automática de dados ocorrem devido a variância e possível ruído presente na base de dados, assim como o *bias* presente no modelo de classificação. No entanto, (Cha Zhang, 2012) apresenta como os métodos

ensemble são eficientes e versáteis para solucionar uma grande variedade de problemas relacionados à classificação, além de reduzir a variância e aumentar a acurácia de sistemas de tomada de decisão.

Combinar os modelos de classificação em um *ensemble* não garante que o modelo resultante possuirá uma performance de classificação maior do que a performance do melhor modelo presente no *ensemble* para aquela base, mas reduz drasticamente as chances do modelo resultante possuir uma performance ruim (Cha Zhang, 2012). A combinação dos modelos resultantes em um *ensemble* pode ser realizada de diferentes maneiras: através da votação de diferentes classificadores treinados na base de dados completa, ao treinar um classificador em subconjuntos obtidos a partir da base de dados original, treinar uma sequência de classificadores que visam reduzir os erros de classificadores anteriores, ou utilizar o aprendizado obtido de um grupo de classificadores como entrada para novos classificadores.

Dada as diferentes possibilidades de combinação de classificadores no intuito de aumentar a performance de classificação, o objetivo deste trabalho trabalho de conclusão de curso é apresentar um estudo e análise sobre quais métodos e porquê utilizar os métodos de *ensemble* no aprendizado de máquina na classificação de diferentes conjuntos de dados. Serão utilizados os métodos de *ensemble* de diferentes categorias, a saber: *Bagging*, *Boosting* e *Stacking*.

Após a execução dos algoritmos, foi notada uma melhora na performance de classificação para a maioria das bases de dados com o uso de métodos *ensemble*. Particularmente, o método *Stacking* obteve o melhor resultado na maior base de dados. Também notou-se que dentre os melhores resultados gerais a presença de algoritmos de aprendizado único. Porém vale ressaltar que nessas situações, os algoritmos de *ensemble* obtiveram resultados próximos.

O restante deste trabalho está dividido da seguinte forma. Na Seção 2 são apresentados os conceitos utilizados nesse trabalho. Na Seção 3 é apresentado o método de pesquisa utilizado. Na Seção 4 são apresentados os resultados. Por fim, na Seção 5 são apresentadas as considerações finais e trabalhos futuros.

2. Conceitos

Nesta seção serão apresentados conceitos chave para o entendimento do aprendizado *ensemble* assim como detalhes dos algoritmos utilizados neste trabalho: ***Bagging***, ***Boosting*** e ***Stacking***.

2.1. Métodos de *Ensemble*

Pode-se observar que durante todo o ciclo de vida de um ser humano ele se encontrará com diversos sistemas de tomada de decisões em consenso (ou *ensemble*). Tais sistemas ou métodos de tomada de decisão estiveram presentes durante toda a história humana pelo menos no que se refere a sua vivência em sociedade (Cha Zhang, 2012). Um exemplo sobre as tomadas de decisões em consenso que pode ser observado no cotidiano é a própria democracia, um sistema no qual uma grande variedade de classificadores (indivíduos) devem avaliar possíveis candidatos, e por meio da votação se elege um desses candidatos.

De maneira análoga, métodos *ensemble* consistem na combinação dos resultados obtidos por um conjunto (ou comitê) de classificadores (Aggarwal, 2014). O

objetivo principal dos métodos *ensemble* é obter modelos com maior performance de classificação e menor variância em comparação com modelos gerados por um algoritmo de classificação de modelo único (Opitz and Maclin, 1999).

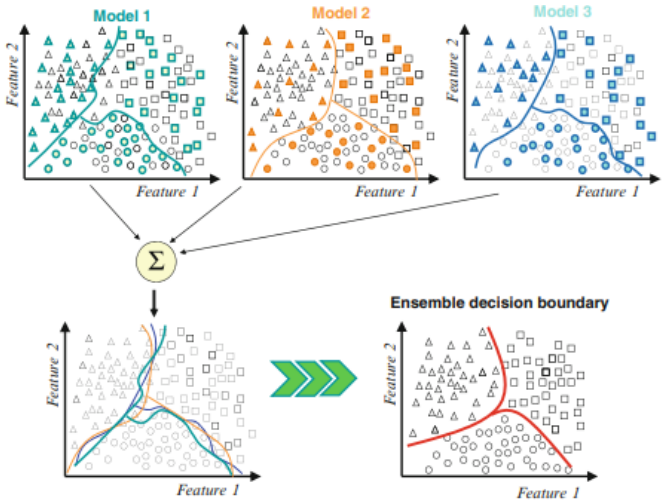

Figura 1. Exemplo de redução da variância em um *ensemble* (Cha Zhang, 2012)

Cha Zhang (2012) apresenta como a combinação de modelos de classificação em um *ensemble* pode melhorar a performance de classificação reduzindo a variância no modelo resultante, representado na Figura 1. Os três modelos apresentados classificavam os objetos da base de maneira diferente com base nas *features* (atributos) 1 e 2. Na combinação dos modelos em questão é realizada a média entre os modelos, resultando em um modelo (*ensemble*) que possui menos erros que os anteriores.

A motivação para a utilização de multiplos classificadores vem da idea de que diferentes classificadores estão sujeitos a diferentes erros e *overfitting*, isto é, quando um modelo treinado é muito eficiente e específico para a classificação da base de dados que foi treinando, tendo baixa performance de classificação para dados não vistos (Aggarwal, 2014).

Aggarwal (2014) apresenta diferentes maneiras de combinar os classificadores, sendo elas: (i) voto majoritário, na qual um mesmo exemplo é classificado por diferentes modelos e a classe com maior número de ocorrências dentre esses classificadores é atribuída ao exemplo; (ii) a combinação ponderada dos resultados dos classificadores, a qual cada classificador recebe pesos de acordo com sua performance de classificação; e (iii) meta-aprendizagem, quando os resultados de um grupo de classificadores são utilizados como base de treino para um novo classificador.

Nas subseções seguintes são apresentados o funcionamento e pseudocódigos dos diferentes métodos de *ensemble* que foram estudados e analisados neste trabalho: *Bagging*, *Boosting* e *Stacking*.

2.2. Bagging

O algoritmo conhecido como o *Bagging* busca dividir a base de treino em diferentes amostras (ou *bags*), treinar diferentes algoritmos em tais amostras, e posteriormente combinar os modelos resultantes por meio de votação. Apresentado inicialmente por Breiman

(1996), o *boostrap aggreggating*, como inicialmente foi chamado, tem como grande vantagem a sua facilidade em ser expandido, ou melhorado, uma vez que é necessário apenas incrementar o número de classificadores.

A técnica utilizada para a criação das amostras, conhecida como *bootstrapping*, consiste em criar n amostras randômicas da base de dados original podendo ou não conter repetições de exemplos da base de dados. Sendo assim, determinado exemplo da base de dados pode aparecer até m vezes, na qual m é o número máximo de exemplos na amostra, distribuídos ao longo de n amostras, ou pode não aparecer em nenhuma (Breiman, 1996).

Após a criação das *bags* e treinados os classificadores nas *bags*, os novos exemplos são classificados de maneira independente e posteriormente os modelos são combinados. Para (Breiman, 1996) uma forma óbvia e eficaz de combinar tais classificações é a realização de uma eleição por voto majoritário, no qual basicamente dado um novo exemplo, verifica-se como cada modelo o classificaria-o e a classe com maior número de votos é atribuída ao exemplo.

Algoritmo 1: *Bagging*

Entrada: Base de treinamento D
Saída : Classificador *ensemble*
1 **para** $i \leftarrow$*1 até n* **faça**
2 | *Construa uma amostra* D_i *da base* D
3 | *Aprenda um modelo de classificação* H_i *com base em* D_i
4 **fim**
5 **retorna** *Classificador ensemble* H

No Algoritmo 1 é apresentado o pseudocódigo da construção de um classificador utilizando a técnica de *Bagging*, no qual o modelo H resultante é um conjunto dos demais modelos obtidos. Na Linha 2, é criada a i-ésima amostra de treinamento D_i, a qual será utilizada para construir o i-ésimo modelo de classificação (Linha 3). As Linhas 2 e 3 são repetidas n vezes (número de amostras). Por fim, na linha 5 o classificador *ensemble* é retornado.

2.3. Boosting

Os algoritmos conhecidos como *Boosting* buscam responder a pergunta inicialmente proposta por Kearns (1988) a qual indaga se um *weak learner*, que consiste um algoritmo de único aprendizado e que tem o desempenho um pouco melhor que a escolha aleatória (Freund, 1997), poderia se tornar um *strong learner*. Inicialmente, o *Boosting* propõe a execução de uma mesma base de treinos para um *weak learner* e então distribuir os pesos da base de treinos de maneira que os exemplos classificados de maneira errada pelo modelo proposto possuem maior peso. Posteriormente, nota-se a proposta do *AdaBoost* por Freund (1997) que visa ser um algoritmo mais prático que aplica pesos adaptáveis com base na performance de cada exemplo e *weak learner*.

AdaBoost ou *Adaptive Boosting* é o algoritmo proposto por (Freund, 1997), no qual os pesos dados a cada objeto na base de dados é constantemente adaptado de modo que os exemplos mais difíceis de serem classificados possuam maior relevância. Freund (1997) define os exemplos de fácil classificação como os exemplos que forem classificados corretamente e sendo assim os exemplos difíceis são os classificados de maneira

errada. O *AdaBoost* é considerado o primeiro algoritmo de *Boosting* prático (Cha Zhang, 2012).

O funcionamento do algoritmo de *AdaBoost*, segundo seu autor Freund (1997), consiste em a inicialmente padronizar o peso de todos os objetos do conjunto de treino. Em seguida, o *weak learner* é executado em tal conjunto de treino. Os pesos dos objetos do conjunto de dados são atualizados, de modo que os exemplos de fácil classificação tenham seu peso reduzido, e os objetos mais difíceis de serem classificados ganham acréscimos no seu peso. São realizadas iterações até que não haja erros de classificação ou até atingir um limite de n iterações.

O grande diferencial do *AdaBoost* é a sua função de distribuição de pesos que tem como objetivo dar maior relevância aos exemplos mais difíceis da base de dados (exemplos que são classificados de maneira errônea com maior frequência) (Cha Zhang, 2012). Sendo assim, a cada iteração do algoritmo todos os exemplos têm seus pesos atualizados tendo como base a última execução, na qual exemplos que já foram classificados corretamente e se mantiveram assim têm seu peso reduzido, enquanto os demais têm seus pesos aumentados.

No Algoritmo 2 é apresentado o pseudocódigo do método *Boosting*. Na Linha 5 é construído o i-ésimo modelo de classificação (H_i) considerando os pesos atuais dos exemplos (W). Na Linha 6 é realizada a avaliação do modelo de classificação para a obtenção dos erros de classificação. Na Linha 7, é feita a atualização dos pesos W_i. As Linhas 4-7 repetem até que o critério de parada seja atingido. Por fim, na Linha 9 retorna-se o classificador obtido na última iteração do processo (H_n).

Algoritmo 2: *Boosting*

Entrada: Base de treinamento D
Saída : Classificador *ensemble*
1 Realize a distribuição inicial dos pesos W
2 $i \leftarrow 0$
3 **repita**
4 n++
5 Treine o *weak learner* H_i na base D
6 Avalie o modelo de classificação H_i com base em D_i
7 Atualize a distribuição de pesos W com base na taxa de erros e_i de H_i
8 **até** $i \geq n$ ***ou*** $e_i = 0$
9 **retorna** *Classificador ensemble* H_n

2.4. Stacking

Aggarwal (2014) caracteriza o *Stacking* como um meta-algoritmo de aprendizado de máquina. O meta-aprendizado consiste em aprender em um conjunto de metadados ao invés de aprender em um conjunto de dados original (Lemke et al., 2015).

Na execução da técnica de *Stacking*, um conjunto de n algoritmos, chamados de classificadores base, são treinados na base de dados gerando n modelos. Cada modelo será utilizado para prever as classes dos exemplos de treinamento. Porém, ao invés de utilizar as previsões diretamente para realizar a classificação, as previsões são utilizados para gerar uma nova representação do exemplo. Ting and Witten (1999) indica que uma

maneira de utilizar as previsões modelos é através de um vetor cuja cada posição é preenchida com a probabilidade do exemplo pertencer a uma determinada classe da coleção. Uma vez gerada as novas representações dos exemplos, um novo algoritmo de aprendizado é executado utilizando no nova base de dados, na qual as probabilidades informadas por cada algoritmo de primeiro nível se tornam as características a serem avaliadas por esse novo algoritmo (meta-classificador) (Aggarwal, 2014).

De acordo com Aggarwal (2014), o método *Stacking* também pode ser visto como um *framework*, uma vez que basta inserir um novo algoritmo no grupo de classificadores base para se obter um novo modelo final. Para gerar as representações dos novos exemplos, normalmente utiliza-se uma estratégia de validação cruzada. Nesta estratégia, a base de dados é dividida em k partes, $k - 1$ partes são utilizadas para treinar os modelos de classificação, e os exemplos contidos na parte restante é que serão submetidos aos classificadores para terem suas novas representações geradas. Isso é repetido iterativamente até que todos os exemplos tenham suas representações geradas (Ting and Witten, 1999).

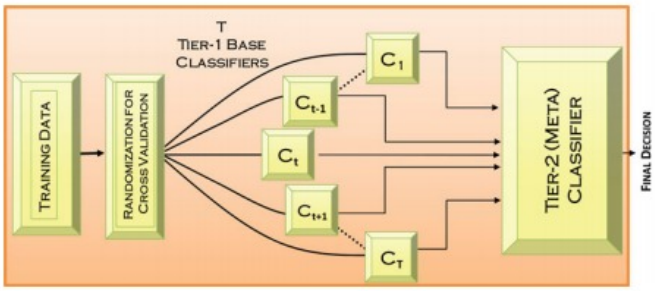

Figura 2. Funcionamento do *Stacking* (Cha Zhang, 2012)

Na Figura 2 é apresentada a ilustração do funcionamento do método *Stacking* também conhecido por *Stacked Generalization*. Como ilustrado na figura, os classificadores base (representados como *tier*-1) são treinados na base de dados e então os modelos de classificação, representados por $C_1, \ldots, C_T$ são utilizados como é ebtrada para o meta-classificador (representado por *tier*-2), que por sua vez gera o modelo/decisão final. Também é importante ressaltar que pode-se ter mais do que 2 níveis. Neste caso, classificadores de *tier*-3 utilizaram os resultados dos classificadores *tier* 2 como base e assim por diante.

No Algoritmo 3 é apresentado o pseudocódigo do método *Stacking*. A primeira diferença a ser notada se encontra nas linhas 1 a 3 nas quais os n classificadores-base h são treinados na base de treinamento. O segundo laço de repetição, que se inicia na linha 4, preenche o vetor de possibilidades para cada elemento da base D. Na linha 8 o vetor de possibilidades é adicionado à nova base de dados M que posteriormente é utilizada para o treinamento do meta-classificador H na linha 10.

De acordo com Aggarwal (2015), o método *Stacking* possui uma grande vantagem em relação aos demais métodos *ensemble* que é a flexibilidade, uma vez que quaisquer algoritmos de aprendizado de máquina podem ser utilizados em cada uma das camadas do *staking*. Além disso, mesmo diante de erros de classificação, o algoritmo na última camada é capaz de aprender com tais erros para produzir classificações corretas.

Algoritmo 3: *Stacking*

Entrada: Base de treinamento D
Saída : Classificador *ensemble*

para $i \leftarrow$ *1 até k* **faça**
    Gere as bases de treino D^i_{treino} *e teste* D^i_{teste}
    Treine os n *classificadores em* D^i_{treino}
    Gere as novas representações dos exemplos em D_{teste} *por meio dos* n *classificadores treinados e insira em* D_{meta}
fim
Treine o meta-classificador H *no conjunto* D_{meta}
retorna *Classificador ensemble* H

3. Método de Pesquisa

O método de pesquisa para este trabalho consiste em 4 etapas necessárias para uma pesquisa na área de aprendizado de máquina: (i) coleta e pré-processamento de bases de dados, (ii) definição de um conjunto de algoritmos que serão utilizados na avaliação experimental, (iii) definição do esquema e métricas de avaliação, (iv) execução dos experimentos, coleta e análise dos resultados. Nas próximas seção são apresentados os detalhes do método de pesquisa utilizado neste trabalho de conclusão de curso, e na próxima seção são apresentados os resultados bem como a análise dos mesmos.

3.1. Conjuntos de Dados e Pré-processamentos

No trabalho um total de dez conjuntos de dados foram utilizados, todas disponíveis no repositório da UCI (*University of California, Irine*) e também disponíveis na biblioteca da *Kaggle*. A bases utilizadas foram:

- *Video Games Rating By 'ESRB'*[1]: coleção de dados que avalia a classificação indicativa de jogos eletrônicos.
- *Heart Disease Data Set*[2]: coleção de sintomas e medidas que podem indicar a ocorrência de doenças cardiovasculares.
- *Phishing Websites Data Set*[3]: conjunto de dados e indicadores que indicam se um site é suspeito de *phishing*.
- *Iris Data Set*[4]: coleção de dados clássica para o reconhecimento de padrões sobre as espécies da planta íris.
- *Star dataset to predict star types*[5]: coleção de dados sobre a classificação de estrelas.
- *Malicious Server Hack*[6]: conjuto de registros sobre incidentes em um servidor que indicam se o incidente foi ou não um ataque *hacker*.
- *Audit Risk Dataset for Classifying Fraudulent Firms*[7]: conjunto de dados sobre empresas que passaram por uma auditoria com o intuito de classificar empresas como fraudulentas e não fraudulentas.

[1]Disponível em: https://www.kaggle.com/imohtn/video-games-rating-by-esrb
[2]Disponível em: https://www.kaggle.com/ronitf/heart-disease-uci
[3]Disponível em: https://archive.ics.uci.edu/ml/datasets/Phishing+Websites
[4]Disponível em: https://archive.ics.uci.edu/ml/datasets/Iris
[5]Disponível em: https://www.kaggle.com/deepu1109/star-datasets
[6]Disponível em: https://www.kaggle.com/lplenka/malicious-server-hack
[7]Disponível em: https://www.kaggle.com/sid321axn/audit-data

- *Glass Identification Dataset*[8]: conjunto de dados com informações para classificar os tipos de vidro que podem ser encontrados em cenas de crime e que podem ser utilizados como evidência.
- *Wine Quality*[9]: coleção de dados que avaliam a qualidade de vinhos.
- *Autism Screening Adult Data Set*[10]: conjunto de dados sobre testes que avaliam se os pacientes possuem Transtorno do Espectro Autista.

Para o pré-processamento dos dados foi necessário garantir que todos os conjuntos de dados estivessem no formato CSV (*Comma Separated Values*). Também foi necessária a padronização dos atributos para valores numéricos entre 0 e 1. Nota-se que para a escolha dos conjuntos foi observado que todos os conjuntos tivessem uma quantidade razoável de elementos: ao menos 100 elementos. Já os atributos categóricos foram tratados de modo que cada um dos possíveis foi utilizado como coluna e então o exemplo cujo o atributo categórico possuir determinado valor teria tal coluna com o valor 1 e as demais como 0. Na Tabela 1 são apresentadas as características das bases de dados utilizadas neste estudo, como o números de exemplos, o total de atributos, e a quantidade de classes de cada conjunto de dados após o pré-processamento.

Base de Dados	**Nº de exemplos**	**Atributos**	**Classes**
Attack	23856	15	2
Audit	773	26	2
Autism	609	97	2
Glass	214	10	7
Heart	304	13	2
Iris	150	4	3
Phishing	11055	30	2
Stars	240	24	6
Video Games	1895	32	4
Wine	6487	12	7

Tabela 1. Características das bases de dados utilizadas no estudo

3.2. Algoritmos e parâmetros

Para a execução dos experimentos, foi desenvolvido um *framework* na linguagem Python[11]. Os algoritmos de aprendizado de máquina, técnicas de *ensemble*, esquemas e medidas de validação foram obtidos por meio da biblioteca *scikit-learn*[12].

Para analisar se a combinação dos algoritmos de classificação de fato melhoram a performance em comparação com o uso de classificadores únicos, foram também executados experimentos considerando classificadores únicos. Como classificadores únicos

[8]Disponível em: https://www.kaggle.com/prashant111/glass-identification-dataset
[9]Disponível em: https://www.kaggle.com/rajyellow46/wine-quality
[10]Disponível em: https://archive.ics.uci.edu/ml/datasets/Autism+Screening+Adult
[11]Código disponível em: https://github.com/luscafidelis/estudo-ensemble
[12]Disponível em: https://scikit-learn.org/stable/

foram utilizados Árvores de Decisão[13], Rede neural (*Multilayer Perceptron Classifier*[14]), *Naive Bayes* (*Gaussian* Naive Bayes[15]), *Support Vector Machines*[16] e *k-Nearest Neighbors*[17]. Todos os algoritmos de aprendizado único foram executados com os valores padrão da biblioteca *scikit-learn*.

Já os metodos *ensemble* foram executados em cada base de dados 5 vezes para o *Bagging*[18], 3 para o *Boosting*[19] e 5 para o *Stacking*[20]. Após cada execução, os resultados foram coletados e comparados com o algoritmo de aprendizado único para verificar a performance de classificação. Tal processo foi realizado para cada uma das bases. Posteriormente os melhores resultados para cada grupo de algoritmo foi coletado e comparado, os algoritmos de aprendizado único foram comparados entre sí e denominados de *Baseline* durante a análise dos resultados.

O *Stacking* foi executado de modo que todos os algoritmos de aprendizado único estivessem no grupo de classificadores-base e em cada uma das 5 execuções, um dos algoritmos também seria utilizado como meta-classificador. No caso do *Bagging* e *Boosting* em cada execução utilizava um dos algoritmos de aprendizado único como classificador que irá gerar o *ensemble*, chamado de *base-estimator* na biblioteca *scikit-learn*. O método *Boosting* não foi executado com os algoritmos k-NN e Redes Neurais, pois os algoritmos presentes na biblioteca não ofereciam suporte à atribuição de peso às amostras da base.

3.3. Critérios para Avaliação

Os critérios para a avaliação foi utilizada a medida F_1, a qual corresponde há uma média harmônica entre as medidas precisão e revocação (Sasaki et al., 2007). Porém, dado o cenário de avaliação onde as bases de dados são compostas por múltiplas classes, foram utilizadas duas estratégias para sumarizar os resultados da medida F_1 calculada para cada uma das classes: *macro-averaging* e *micro-averaging*, as quais respectivamente irão calcular as medidas F_1 através da média da medida F_1 para cada uma das classes e a soma dos termos da medida F_1 para todas as classes.

Como esquema de avaliação foi utilizada a validação cruzada. De acordo com Refaeilzadeh et al. (2009), a validação cruzada é um método estatístico para estimar a performance de classificação de algoritmos de aprendizado de máquina, na qual a base de treino é dividida em k pastas, sendo que iterativamente uma pasta é utilizada para testar o

[13] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[14] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

[15] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

[16] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[17] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[18] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html

[19] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

[20] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html

modelo e as demais $k-1$ pastas para treinar o modelo. A validação é uma técnica importante que avalia a capacidade de generalização de um modelo. Neste trabalho foi utilizada a validação cruzada com 4 *folds*. Mais especificamente, foi utilizada a classe *StratifiedK-Fold*[21] dentro da biblioteca *scikit-learn* para garantir que todos os *folds* teriam as mesma proporção de exemplos para cada classe em relação ao conjunto de dados completo.

4. Resultados

Nesta seção serão apresentados os resultados dos experimentos com cada um dos algoritmos citados na seção anterior. Os resultados foram avaliados utilizando a medida F_1 com médias *macro* e *micro-averaging*. Serão apresentados os resultados de cada algoritmo de aprendizado único junto de suas implementações com métodos *Ensemble*.

4.1. Resultados para o Algoritmo k-NN

Nesta subseção são apresentados os resultados da execução do algoritmo k-NN, assim como os métodos de *ensemble Bagging* e *Staking*, os quais utilizaram o *k-NN* como classificador base ou classificador da última camada do *stacking* respectivamente. O *Boosting* não foi testado junto ao k-NN devido a implementação utilizada não considerar pesos para os exemplos.

Os resultados estão apresentados nas Tabelas 2 e 3 para as medidas *micro* e *macro* F_1, respectivamente. Observando a Tabela 2 (*micro-*F_1), nota-se que o método *Stacking* obteve o melhor resultado em nove das dez base de dados testadas. Já o método *Bagging* obteve o mesmo resultado que o *Stacking* nas bases *Glass* e Iris. Já o ranking base obteve o melhor resultado em duas coleções, sendo que em uma delas (*Iris*) ficou empatado com os métodos de ensemble.

Base de dados	**k-NN**	**Bagging**	**Boosting**	**Stacking**
Attack	0.9581	0.9583	-	**0.9990**
Audit	0.7987	0.7987	-	**0.9870**
Autism	0.9669	0.9752	-	**1.0000**
Glass	0.6190	**0.6429**	-	**0.6429**
Heart	**0.8000**	0.7833	-	0.7667
Iris	**1.0000**	**1.0000**	-	**1.0000**
Phishing	0.9095	0.9150	-	**0.9254**
Stars	0.8125	0.8125	-	**0.8750**
Video Games	0.8457	0.8478	-	**0.8689**
Wine	0.9669	0.9752	-	**1.0000**

Tabela 2. Resultados para o algoritmo k-NN utilizando a medida Micro-F_1

Observando a Tabela 3 (*macro-*F_1), nota-se que o método *Stacking* obteve o melhor resultado em 8 das dez base de dados testadas. Já o método *Bagging* obteve o melhor resultado para uma única base (Glass), bem como o algoritmo base k-NN obteve o melhor resultado apenas para a base *Heart*. Portanto, pode-se observar que o uso da técnica de

[21] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html`

Stacking tendo o k-NN como classificador da última camada foi benéfico para aumentar a performance de classificação. Dentre as melhorias nos resultados providas pela técnica *Stacking*, destacam-se a melhora de 18.83% na medida *micro*-F_1 para a base *Audit*, e uma melhora 40.79% na medida *macro*-F_1 considerando a base *Attack*.

Base de dados	k-NN	Bagging	Boosting	Stacking
Attack	0.5860	0.5897	-	**0.9939**
Audit	0.7582	0.7582	-	**0.9863**
Autism	0.9605	0.9706	-	**1.0000**
Glass	0.4447	**0.4521**	-	0.3897
Heart	**0.7943**	0.7727	-	0.7600
Iris	**1.0000**	**1.0000**	-	**1.0000**
Phishing	0.9081	0.9136	-	**0.9245**
Stars	0.5153	0.5153	-	**0.6269**
Video Games	0.8497	0.8515	-	**0.8728**
Wine	0.9605	0.9706	-	**1.0000**

Tabela 3. Resultados para o algoritmo k-NN utilizando a medida Macro-F_1

4.2. Resultados para o Algoritmo *Support Vector Machines*

Nesta subseção são apresentados os resultados da execução do algoritmo *Support Vector Machines*, assim como as técnicas de *ensemble* que fizeram uso do *Support Vector Machines* como classificador base. Os resultados obtidos considerando as medidas *micro*-F_1 e *macro*-F_1 são apresentados respectivamente nas Tabelas 4 e 5.

Ao observar a Tabela 4, nota-se que o método *Stacking* obteve o melhor resultado em sete das dez base de dados testadas considerando a medida *micro*-F_1. O método *Bagging* obteve o melhor resultado apenas para as bases *Iris* e *Phishing*, enquanto que o algoritmo base *Support Vector Machines* obteve o melhor resultado para três bases (*Heart*, *Iris*, *Video Games*). Já o *Boosting* obteve apenas a melhor performance em um das coleções, sendo esta um empate com as demais técnicas.

Base de dados	Support Vector Machines	Bagging	Boosting	Stacking
Attack	0.9554	0.9554	0.9554	**0.9996**
Audit	0.8636	0.8636	0.6039	**1.0000**
Autism	0.9669	0.9587	0.7025	**1.0000**
Glass	0.6905	**0.5238**	0.3571	**0.7143**
Heart	**0.7833**	0.7500	0.5500	0.7333
Iris	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Phishing	0.9349	**0.9367**	0.5572	0.9267
Stars	0.8750	0.8750	0.4792	**0.8958**
Video Games	**0.8858**	0.8774	0.3636	0.8837
Wine	0.9669	0.9587	0.7025	**1.0000**

Tabela 4. Resultados para o algoritmo *Support Vector Machines* utilizando a medida Micro-F_1

Na Tabela 5 (resultados da medida *macro*-F_1), nota-se que os melhores resultados são semelhantes aos resultados para a medida *micro*-F_1, isto é, o *Stacking* obteve o melhor resultado em 7 coleções, classificador base em 3, *Bagging* em duas, e o *Boosting* em uma coleção. Portanto, pode-se observar novamente que o uso da técnica de *Stacking* tendo o *Support Vector Machines* como classificador da última camada foi benéfico para aumentar a performance de classificação. Dentre as melhorias nos resultados providas pela técnica *Stacking*, destacam-se a melhora de 13.64% na medida *micro*-F_1 para a base *Audit*, e uma melhora 50.89% na medida *macro*-F_1 considerando a base *Attack*.

Base de dados	Support Vector Machines	Bagging	Boosting	Stacking
Attack	0.4886	0.4886	0.4886	**0.9975**
Audit	0.8453	0.8453	0.3765	**1.0000**
Autism	0.9598	0.9493	0.4126	**1.0000**
Glass	0.4099	0.3263	0.0877	**0.4222**
Heart	**0.7727**	0.7413	0.3548	0.7222
Iris	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Phishing	0.9338	**0.9357**	0.3578	0.9256
Stars	0.7275	0.7448	0.1080	**0.8027**
Video Games	**0.8892**	0.8810	0.1333	0.8876
Wine	0.9598	0.9493	0.4126	**1.0000**

Tabela 5. Resultados para o algoritmo *Support Vector Machines* utilizando a medida Macro-F_1

4.3. Resultados para o Algoritmo Árvores de Decisão

Nesta subseção são apresentados os resultados da execução do algoritmo Árvores de Decisão, assim como as técnicas de *ensemble* que fizeram uso do Árvores de Decisão como classificador base. Os resultados obtidos considerando as medidas *micro*-F_1 e *macro*-F_1 são apresentados respectivamente nas Tabelas 6 e 7.

Ao observar a Tabela 6 (*micro*-F_1), nota-se que todos as técnicas obtiveram resultados iguais em cinco das dez bases de dados. Destaca-se aqui a técnica Bagging, a qual obteve o melhor resultado em 8 bases, o a algoritmo base que obteve o melhor resultado em 7 bases.

Ao observar a Tabela 7 (*macro*-F_1), nota-se que todos os algoritmos testados obtiveram resultados iguais e com valor máximo em quatro das bases testadas. Levando em conta o resultado igual em quatro das bases, tanto o algoritmo de aprendizado único Árvores de Decisão quanto o *Boosting* obtiveram o melhor resultado em seis das 10 bases de teste. Considerando os resultados iguais, o método *Bagging* obteve o melhor resultado em sete das dez bases de dados testadas. Portanto, pode-se observar que diferente dos algoritmos analisados até aqui, o uso da técnica de *Bagging* tendo o Árvores de Decisão como classificador da última camada foi benéfico para aumentar a performance de classificação. Dentre as melhorias nos resultados providas pela técnica *Bagging*, destacam-se a melhora de 14.29% na medida *micro*-F_1 para a base *Glass*, e uma melhora 23.03% na medida *macro*-F_1 considerando a base Attack.

Base de dados	Árvores de Decisão	Bagging	Boosting	Stacking
Attack	0.9990	**0.9992**	0.9987	0.9975
Audit	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Autism	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Glass	0.5952	**0.7381**	0.5000	0.5714
Heart	**0.7167**	**0.7167**	0.7000	0.6833
Iris	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Phishing	0.9005	0.9254	**0.9439**	0.9172
Stars	**0.8958**	**0.8958**	**0.8958**	**0.8958**
Video Games	**0.8710**	0.8668	0.8541	0.8393
Wine	**1.0000**	**1.0000**	**1.0000**	**1.0000**

Tabela 6. Resultados para o algoritmo Árvores de Decisão utilizando a medida Micro-F_1

Base de dados	Árvores de Decisão	Bagging	Boosting	Stacking
Attack	0.9939	**0.9951**	0.9927	0.9853
Audit	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Autism	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Glass	0.3737	**0.6040**	0.3258	0.5209
Heart	0.6978	**0.7027**	0.6827	0.6760
Iris	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Phishing	0.8995	0.9245	**0.9431**	0.9164
Stars	**0.8630**	0.8585	**0.8630**	0.8027
Video Games	**0.8750**	0.8710	0.8584	0.8468
Wine	**1.0000**	**1.0000**	**1.0000**	**1.0000**

Tabela 7. Resultados para o algoritmo Árvores de Decisão utilizando a medida Macro-F_1

4.4. Resultados para o Algoritmo Rede Neural

Nesta subseção são apresentados os resultados da execução do algoritmo Rede Neural, assim como os métodos de *ensemble Bagging* e *Staking*, os quais utilizaram o Rede Neural como classificador base ou classificador da última camada do *stacking* respectivamente. O *Boosting* não foi testado junto ao Rede Neural devido a implementação utilizada não considerar pesos para os exemplos.

Os resultados estão apresentados nas Tabelas 8 e 9 para as medidas *micro* e *macro* F_1, respectivamente. Na Tabela 8, nota-se que o *Stacking* obteve a melhor performance de classificação, em oito das dez bases de dados, sendo que obteve o valor máximo em quatro delas. Já o método *Bagging* obteve o melhor resultado em 4 bases e o algoritmo base em 3 bases.

Observando a Tabela 9 (*macro*-F_1), nota-se que os melhores resultados são semelhantes aos da medida (*micro*-F_1). Portanto, pode-se observar novamente que o uso da técnica de *Stacking* tendo a Rede Neural como classificador da última camada foi benéfico para aumentar a performance de classificação. Dentre as melhorias nos resultados provi-

Base de dados	Rede Neural	Bagging	Boosting	Stacking
Attack	0.9813	0.9799	-	**0.9987**
Audit	0.8896	0.8701	**-**	**1.0000**
Autism	0.9256	0.9256	**-**	**1.0000**
Glass	0.6429	**0.6905**	-	0.5714
Heart	**0.7667**	**0.7667**	-	**0.7667**
Iris	**1.0000**	**1.0000**	**-**	**1.0000**
Phishing	0.9313	**0.9421**	**-**	0.9349
Stars	**0.8958**	0.875	**-**	**0.8958**
Video Games	0.8710	0.8689	-	**0.8795**
Wine	0.9256	0.9256	**-**	**1.0000**

Tabela 8. Resultados para o algoritmo Rede Neural utilizando a medida Micro-F_1

das pela técnica *Stacking*, destacam-se a melhora de 11.04% na medida *micro*-F_1 para a base *Audit*, e uma melhora 12.28% na medida *macro*-F_1 também para a base *Audit*.

Base de dados	Rede Neural	Bagging	Boosting	Stacking
Attack	0.8777	0.8710	-	**0.9927**
Audit	0.8772	0.8534	**-**	**1.0000**
Autism	0.9117	0.9117	**-**	**1.0000**
Glass	0.4525	**0.5899**	-	0.3645
Heart	**0.7569**	**0.7569**	-	0.7532
Iris	**1.0000**	**1.0000**	**-**	**1.0000**
Phishing	0.9304	**0.9413**	**-**	0.9340
Stars	**0.8080**	0.7489	**-**	0.8027
Video Games	0.8754	0.8733	-	**0.8832**
Wine	0.9117	0.9117	**-**	**1.0000**

Tabela 9. Resultados para o algoritmo Rede Neural utilizando a medida Macro-F_1

4.5. Resultados para o Algoritmo *Naive Bayes*

Nesta subseção são apresentados os resultados da execução do algoritmo *Naive Bayes*, assim como as técnicas de *ensemble* que fizeram uso do *Naive Bayes* como classificador base. Os resultados obtidos considerando as medidas *micro*-F_1 e *macro*-F_1 são apresentados respectivamente nas Tabelas 10 e 11.

Na Tabela 10 (*micro*-F_1), são apresentados os resultados da execução do *Naive Bayes* e implementações em *ensemble* utilizando-o como classificador base. Pode-se observar que o *Stacking* obteve a melhor performance de classificação em nove das dez bases de dados, sendo o valor máximo obtido em quatro delas. Já o *Boosting* obteve o melhor valor em duas coleções, e o*Bagging* obtive o melhor resultado em apenas uma coleção. Por fim, o classificador base obteve o melhor resultado em duas coleções.

De acordo com a Tabela 11 o *Stacking* também obteve a melhor performance de classificação em nove das dez bases de dados, sendo que obteve o valor máximo em

Base de dados	Naive Bayes	Bagging	Boosting	Stacking
Attack	0.9472	0.9457	0.4691	**0.9847**
Audit	0.8442	0.8442	0.9545	**1.0000**
Autism	0.3471	0.3388	0.7025	**1.0000**
Glass	0.1667	0.1905	**0.5952**	0.2143
Heart	0.7167	0.7167	0.4167	**0.7333**
Iris	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Phishing	0.5871	0.5875	0.5834	**0.9276**
Stars	**0.9167**	0.8750	0.7083	**0.9167**
Video Games	0.5497	0.5518	0.6638	**0.7632**
Wine	0.3471	0.3388	0.7025	**1.0000**

Tabela 10. Resultados para o algoritmo *Naive Bayes* utilizando a medida Micro-F_1

quatro bases. Já o *Boosting* e *Bagging* obtiveram o melhor resultado em uma coleção, e o classificador base em duas coleções. Portanto, pode-se observar que o uso da técnica de *Stacking* tendo a *Naive Bayes* como classificador da última camada foi benéfico para aumentar a performance de classificação. Dentre as melhorias nos resultados providas pela técnica *Stacking*, destacam-se a melhora de 65.29% na medida *micro*-F_1 para a base *Autism*, e uma melhora 68.45% na medida *macro*-F_1 também considerando a base *Autism*.

Base de dados	Naive Bayes	Bagging	Boosting	Stacking
Attack	0.5578	0.5642	0.3739	**0.9225**
Audit	0.8204	0.8204	0.9514	**1.0000**
Autism	0.3155	0.3094	0.4814	**1.0000**
Glass	0.2361	0.2795	0.4087	**0.4139**
Heart	0.7068	0.7068	0.3879	**0.7257**
Iris	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Phishing	0.5467	0.5473	0.4237	**0.9270**
Stars	**0.8781**	0.7823	0.5616	0.8563
Video Games	0.5367	0.5395	0.6828	**0.7736**
Wine	0.3155	0.3094	0.4814	**1.0000**

Tabela 11. Resultados para o algoritmo *Naive Bayes* utilizando a medida Macro-F_1

4.6. Comparativo Geral

Nesta seção, são apresentados os melhores resultados considerando tanto o algoritmo base, quanto às técnicas de *ensemble*, independente dos algoritmos de aprendizado utilizados. Portanto, faz-se aqui a análise do melhor caso. Nas Tabelas 12 e 13 são apresentados os resultados considerando as medidas *micro*-F_1 e *macro*-F_1 respectivamente.

Ao observar a Tabela 12 (*micro*-F_1), observa-se que considerando os melhores resultados obtidos por classificadores base, obtém-se a maior performance de classificação em 8 das 10 bases. Já utilizando as técnicas de *ensemble*, o *Stacking* obteve o melhor resultado em 7 bases, e o *Bagging* e *Boosting* em 5 bases.

Base de dados	Base	Bagging	Boosting	Stacking
Attack	0.9990	0.9992	0.9990	**0.9996**
Audit	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Autism	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Glass	0.6905	**0.7381**	0.6429	0.7143
Heart	**0.8000**	0.7833	0.7667	0.7667
Iris	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Phishing	**0.9349**	0.9421	**0.9439**	**0.9349**
Stars	**0.9167**	0.8958	0.8958	**0.9167**
Video Games	**0.8858**	0.8774	0.8795	0.8837
Wine	**1.0000**	**1.0000**	**1.0000**	**1.0000**

Tabela 12. Melhores resultados para cada algoritmo com a medida F_1-Micro

Já ao observar a Tabela 13 (*macro*-F_1), observa-se que considerando os melhores resultados obtidos por classificadores base, obtém-se a maior performance de classificação em 8 das 10 bases novamente. Já utilizando as técnicas de *ensemble*, todas obtiveram os melhores resultados para 5 das 10 bases de dados.

Base de dados	Base	Bagging	Boosting	Stacking
Attack	0.9939	0.9951	0.9927	**0.9975**
Audit	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Autism	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Glass	0.4525	**0.6040**	0.4087	0.5209
Heart	**0.7727**	0.7569	0.6827	0.7532
Iris	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Phishing	0.9338	0.9413	**0.9431**	0.9340
Stars	**0.8781**	0.8585	0.8630	0.8563
Video Games	**0.8892**	0.8810	0.8584	0.8876
Wine	**1.0000**	**1.0000**	**1.0000**	**1.0000**

Tabela 13. Melhores resultados para cada algoritmo com a medida F_1-Macro

Observa-se que a possibilidade de escolher o melhor algoritmo de aprendizado de máquina pode propiciar o melhor resultado para a maioria das vezes. Porém, nota-se que nem sempre um único algoritmo irá obter o melhor resultado, uma vez que analisando individualmente cada algoritmo, as técnicas de *ensemble*, principalmente a *stacking*, obtiveram os melhores resultados na maioria das vezes. Além disso, nota-se que mesmo quando uma técnica de *ensemble* não supera os resultados de um classificador base, os resultados são próximos, fazendo com que o uso de ensemble seja aconselhável em aplicações práticas.

5. Considerações Finais

Este trabalho se propôs a realizar um estudo sobre a performance de classificação dos métodos *ensemble*, *Bagging*, *Boosting* e *Stacking*, e compará-los com os resultados obtidos com o uso de classificadores únicos. Sendo assim foi possível notar que a utilização

de *ensembles* pode melhorar a performance de classificação de dados na maioria dos casos quando comparados aos classificadores simples.

No entanto, assim como afirmado por Cha Zhang (2012), a combinação de modelos de classificação em um *ensemble* não garante que os resultados serão superiores aos de classificadores de aprendizado único, mas reduzem a chance do modelo obtido possuir uma performance muito inferior. Ao observar todos os resultados por algoritmo, notou-se que o método *Stacking* obteve a melhor performance de classificação na maioria dos casos, sendo que em todos os experimentos realizados considerando diferentes algoritmos base, o *ensemble* obteve o melhor resultado ou resultado igual ao maior em pelo menos 4 das 10 bases. Vale ressaltar que houve casos em que a técnica de *Stacking* obteve a melhor performance em 9 das 10 bases de dados.

Quanto aos trabalhos futuros, pretende-se aprofundar os estudos no método *Stacking*, uma vez que a utilização do mesmo resultou na melhora da performance de classificação em grande parte das execuções realizadas. Para isso, pretende-se analisar a utilização de mais camadas, ou ainda considerar outros classificadores na última camada do *stacking*, como redes neurais profundas.

Referências

C. C. Aggarwal. *Data Classification: Algorithms and Applications*. Chapman & Hall, 1st edition, 2014.

C. C. Aggarwal. *Data Mining: The Textbook*. Springer, Cham, 2015. ISBN 978-3-319-14141-1. doi: 10.1007/978-3-319-14142-8.

L. Breiman. Bagging predictors. *Machine Learning, 24, 123-140 (1996)*, 1996. doi: https://link.springer.com/content/pdf/10.1007/BF00058655.pdf.

Y. M. E. Cha Zhang. *Ensemble machine learning: methods and applications.* Springer Science & Business Media, 2012.

R. E. S. Freund, Yoav. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences 55.1 (1997): 119-139*, 1997. doi: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=4BF3325D8222B3234BB95971FCAD8759?doi=10.1.1.56.9855&rep=rep1&type=pdf.

M. Kearns. Thoughts on hypothesis boosting. *Machine Learning class pro ject, Dec. 1988*, 1988. doi: https://www.cis.upenn.edu/~mkearns/papers/boostnote.pdf.

C. Lemke, M. Budka, and B. Gabrys. Metalearning: a survey of trends and technologies. *Artificial intelligence review*, 44(1):117–130, 2015.

D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

P. Refaeilzadeh, L. Tang, and H. Liu. Cross-validation. *Encyclopedia of database systems*, 5:532–538, 2009.

Y. Sasaki et al. The truth of the f-measure. 2007, 2007. URL `https://www.toyota-ti.ac.jp/Lab/Denshi/COIN/people/yutaka.sasaki/F-measure-YS-26Oct07.pdf`.

P. Tan, M. Steinbach, A. Karpatne, and V. Kumar. *Introduction to Data Mining*. What's New in Computer Science Series. Pearson, 2019. ISBN 9780133128901. URL `https://books.google.com.br/books?id=\_ZQ4MQEACAAJ`.

K. M. Ting and I. H. Witten. Issues in stacked generalization. *Journal of artificial intelligence research*, 10:271–289, 1999.