PUCRS

FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

ANTÔNIO DO NASCIMENTO LEÃES NETO

CLASSIFICAÇÃO COM ALGORITMO ADABOOST.M1

O MITO DO LIMIAR DE ERRO DE TREINAMENTO

Porto Alegre
2017

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CLASSIFICAÇÃO COM ALGORITMO ADABOOST.M1: O MITO DO LIMIAR DE ERRO DE TREINAMENTO

ANTÔNIO DO NASCIMENTO LEÃES NETO

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Rafael H. Bordini

Porto Alegre
2018

Ficha Catalográfica

N469 Neto, Antônio do Nascimento Leães

Classificação com algoritmo AdaBoost.M1 : O mito do limiar de erro de treinamento / Antônio do Nascimento Leães Neto . – 2017.

69 f.

Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Rafael Heitor Bordini.

1. Mineração de dados. 2. Classificação. 3. Combinação de classificadores. 4. Boosting. 5. AdaBoost.M1. I. Bordini, Rafael Heitor. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS com os dados fornecidos pelo(a) autor(a).

Antônio do Nascimento Leães Neto

Classificação com algoritmo AdaBoost.M1: O mito do limiar de erro de treinamento

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Ciência da Computação do Programa de Pós-Graduação em Ciencia da Computação, Faculdade de Informática da Pontifícia Universidade Católica do Rio Grande do Sul.

Aprovado em 20 de novembro de 2017.

BANCA EXAMINADORA:

Prof. Dr. Avelino Francisco Zordo (PPGCC/PUCRS)

Dr. Roger Leizke (PUCRS)

Prof. Dr. Rafael Heitor Bordini (PPGCC/PUCRS - Orientador)

CLASSIFICAÇÃO COM ALGORITMO ADABOOST.M1: O MITO DO LIMIAR DE ERRO DE TREINAMENTO

RESUMO

O crescimento acelerado dos repositórios de dados, nas diversas áreas de atuação, abre espaço para pesquisas na área da mineração de dados, em específico, com os métodos de classificação e de combinação de classificadores. O Boosting é um desses métodos, e combina os resultados de diversos classificadores com intuito de obter melhores resultados. O propósito central desta dissertação é responder a questão de pesquisa com a experimentação de alternativas para aumentar a eficácia e o desempenho do algoritmo AdaBoost.M1 que é a implementação frequentemente empregada pelo Boosting. Foi feito um estudo empírico levando em consideração aspectos estocásticos tentando lançar alguma luz sobre um parâmetro interno obscuro em que criadores do algoritmo e outros pesquisadores assumiram que o limiar de erro de treinamento deve ser correlacionado com o número de classes no conjunto de dados de destino e, logicamente, a maioria dos conjuntos de dados deve usar um valor de 0.5. Neste trabalho, apresentamos evidências empíricas de que isso não é um fato, mas provavelmente um mito originado pela aplicação da primeira definição do algoritmo. Para alcançar esse objetivo, foram propostas adaptações para o algoritmo, focando em encontrar uma sugestão melhor para definir esse limiar em um caso geral.

Palavras-Chave: Mineração de dados, Classificação, Combinação de classificadores, Boosting, AdaBoost.M1.

CLASSIFICATION WITH ADABOOST.M1: THE TRAINING ERROR THRESHOLD MYTH

ABSTRACT

The accelerated growth of data repositories, in the different areas of activity, opens space for research in the area of data mining, in particular, with the methods of classification and combination of classifiers. The Boosting method is one of them, which combines the results of several classifiers in order to obtain better results. The main purpose of this dissertation is the experimentation of alternatives to increase the effectiveness and performance of the algorithm AdaBoost.M1, which is the implementation often employed by the Boosting method. An empirical study was perfered taking into account stochastic aspects trying to shed some light on an obscure internal parameter, in which algorithm creators and other researchers assumed that the training error threshold should be correlated with the number of classes in the target data set and logically, most data sets should use a value of 0.5. In this paper, we present an empirical evidence that this is not a fact, but probably a myth originated by the mistaken application of the theoretical assumption of the joint effect. To achieve this goal, adaptations were proposed for the algorithm, focusing on finding a better suggestion to define this threshold in a general case.

Keywords: Data Mining, Ensemble Methods, Classification, Boosting, AdaBoost.M1.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Nos últimos anos a informatização da maioria dos processos e tarefas das diversas áreas de atuação fez com que a quantidade de informações armazenadas pelas instituições aumentasse de maneira exponencial. A diversidade e a abundância desses dados têm despertado o interesse em diversos contextos na busca do conhecimento e na extração de informações úteis desses repositórios [TSK05]. O desenvolvimento e o crescimento acelerado desses contextos abrem espaço para a utilização de técnicas das quais é possível obter vantagens estratégicas e econômicas de maneira mais ágil e segura em relação a métodos convencionais baseados em suposições. A descoberta de conhecimento em banco de dados, conhecido como KDD do inglês *Knowledge Discovery from Databases*, é um processo para extrair de forma automatizada ou conveniente o conhecimento de grandes repositórios de dados [HPK11]. O KDD é a busca pelo conhecimento, e seu processo é composto pelas etapas de limpeza, integração, seleção, transformação, mineração, avaliação de padrões e conhecimento. Essas técnicas fazem parte da mineração de dados que utiliza tarefas de diferentes áreas, tais como aprendizado de máquina e estatística.

Segundo Tan et al. [TSK05], a mineração de dados é parte integral do KDD e as técnicas que a compõe são organizadas para descobrir padrões úteis que podem estar escondidos. Elas também servem para fornecer previsões de informações futuras como estimativas de vendas, faturamento, cenários de riscos, entre outros. De acordo com Harrington [Har12], a mineração é uma tarefa desafiadora e os métodos tradicionais de consulta e análise onde é necessário explorar, conhecer e preparar os dados são ineficientes para descobrir informações relevantes dentro desses repositórios. A alternativa são algoritmos avançados capazes de extrair informações valiosas de forma automática.

Os tipos clássicos de mineração de dados são: classificação, agrupamento, regressão e associação. A classificação é uma abordagem do aprendizado de máquina (*Machine learning*) adequada para a tarefa de organização das informações em categorias. É a tarefa de aprendizado de uma função alvo que mapeia um conjunto de atributos para um atributo alvo pré-determinado [TSK05]. A classificação é uma forma de análise de dados para extrair modelos que descrevem classes de dados importantes com inúmeras aplicações incluindo a detecção de fraudes, marketing de destino, previsão de desempenho, fabricação e diagnóstico médico [HPK11]. Dentro da área da classificação encontram-se os métodos de combinação de classificadores tais como o Bagging [Bre96], Boosting [Sch90] e Random Forests [Bre01] que combinam os resultados de vários classificadores (Classificador base), com o objetivo de melhorar os resultados da classificação dos dados.

De um ponto de vista prático, esta dissertação tem como objetivo principal propor uma adaptação para a principal implementação do método Boosting, o algoritmo AdaBoost.M1. Em específico, o objetivo consiste em analisar alguns aspectos. O primeiro é avaliar

a possibilidade de aumentar a eficácia e o desempenho do algoritmo quanto aos resultados de taxa de acerto e tempo de execução. E o segundo é considerar aspectos estocásticos no que diz respeito a aleatoriedade das sementes utilizadas no processo de geração dos classificadores.

Uma questão a ser observada é o limiar do erro de treinamento que é relativamente um ponto obscuro na descrição do algoritmo original de Freund e Schapire [FS96]. Em Zhu et al. [ZZRH09], é apresentada uma proposta de extensão para o algoritmo Boosting para problemas multi-classes denominada SAMME (*Stagewise Additive Modeling using a Multi-class Exponential loss function*), que dispensa a redução de problemas de multi-classes para vários subproblemas de duas classes. Este algoritmo é diferente da implementação original, pois requer que os resultados de cada classificador base sejam melhores que escolhas aleatórias, ou seja, considera o limiar de erro como o inverso do número de classes. No entanto o trabalho de Zhu et al. [ZZRH09] considera como classificador base o Bayes rule. Assim motivado, esta dissertação segue a mesma linha exploratória de analisar outras opções de limiar de erro, mas considera como classificador base o algoritmo J48 que é semelhante ao algoritmo C4.5 empregado na definição original de Freund e Schapire [FS96].

Este trabalho tem como propósito a exploração do método Boosting de combinação de classificadores com foco no limiar de erro treinamento, parâmetro interno que que interrompe o processo de classificação caso o classificador apresente perda de desempenho. Em estudos apresentados por Fernandes et al. [FLR10], o comportamento do método Boosting é analisado quando submetido à aleatoriedade e os resultados demonstram vulnerabilidade do método, quando considerados esses aspectos. Em Fernandes et al. [FLNR13], é apresentado um estudo da adaptação do algoritmo Random Forests para que seja menos vulnerável aos impactos da aleatoriedade. Para tanto é proposta uma extensão denominada SARF (*Stochastic Aware Random Forests*) cujos resultados são comparados a outros métodos, como por exemplo, o Boosting.

Esta proposta não tem como interesse a criação de um novo método, e sim, propor uma adaptação para o algoritmo existente a fim de melhorar a taxa de acerto e o tempo de execução considerando a aleatoriedade da geração de amostras dos classificadores.

A partir disso, surge a questão de pesquisa:

- Existe algo que possa ser alterado no limiar de erro de treinamento para melhorar a eficácia e o desempenho do algoritmo AdaBoost.M1 levando em consideração aspectos estocásticos?

Baseado nisso, o objetivo geral e os objetivos específicos foram traçados buscando responder essa questão.

1.1 Objetivos

O objetivo geral desta dissertação é analisar alternativas para melhorar a eficácia e o desempenho do algoritmo AdaBoost.M1, e propor uma adaptação do algoritmo que leve em consideração o limiar de erro de treinamento e aspectos estocásticos.

Tendo em vista o objetivo geral apresentado, foram definidos os seguintes objetivos específicos:

- Implementar uma adaptação para o algoritmo AdaBoost.M1, considerando o limiar de erro de treinamento;
- Analisar a variabilidade desse algoritmo com respeito à aleatoriedade;
- Avaliar, através de experimentos práticos, os benefícios e malefícios da adaptação proposta;
- Submeter os resultados a testes estatísticos formais.

1.2 Estrutura do documento

Este documento está estruturado da seguinte maneira: o Capítulo 2 apresenta os conceitos básicos, bem como os fundamentos para a compreensão deste trabalho, tais como: Mineração de Dados, Aprendizado de Máquina, Classificação, Métodos de Combinação de Classificadores, Método Boosting, Validação Cruzada, Processos Estocásticos e Testes de Hipóteses. O Capítulo 3 apresenta os trabalhos relacionados. O Capítulo 4 apresenta a adaptação do AdaBoost.M1 utilizada para execução dos experimentos. No Capítulo 5, são apresentados os resultados da adaptação do algoritmo AdaBoost.M1. Por último, no capítulo 6 são apresentadas as considerações finais e a contribuição desta dissertação.

2. CONCEITOS BÁSICOS

Neste capítulo são apresentados alguns conceitos básicos importantes para a compreensão do trabalho desenvolvido. Dessa forma, as seções a seguir apresentam noções básicas sobre Mineração de Dados (Seção 2.1), Aprendizado de Máquina (Seção 2.2), Classificação (Seção 2.3), Árvores de Decisão (Seção 2.4), Métodos de Combinação de Classificação (Seção 2.5), Validação Cruzada (Seção 2.6), Processos Estocásticos (Seção 2.7) e Teste de Hipóteses (Seção 2.8).

2.1 Mineração de dados

A mineração de dados é a técnica que possibilita a exploração e a análise de dados utilizando-se de algoritmos com a capacidade de analisar grandes quantidades de dados. É um processo automático para descobrir padrões nos dados que sejam significativos para obter vantagens, geralmente, de alguma natureza econômica [TSK05]. De acordo com Han e Kamber [HPK11], a Figura 2.1 apresenta a adaptação das etapas do processo de descoberta de conhecimento em banco de dados (KDD) que consiste em uma sequência iterativa representada por etapas.

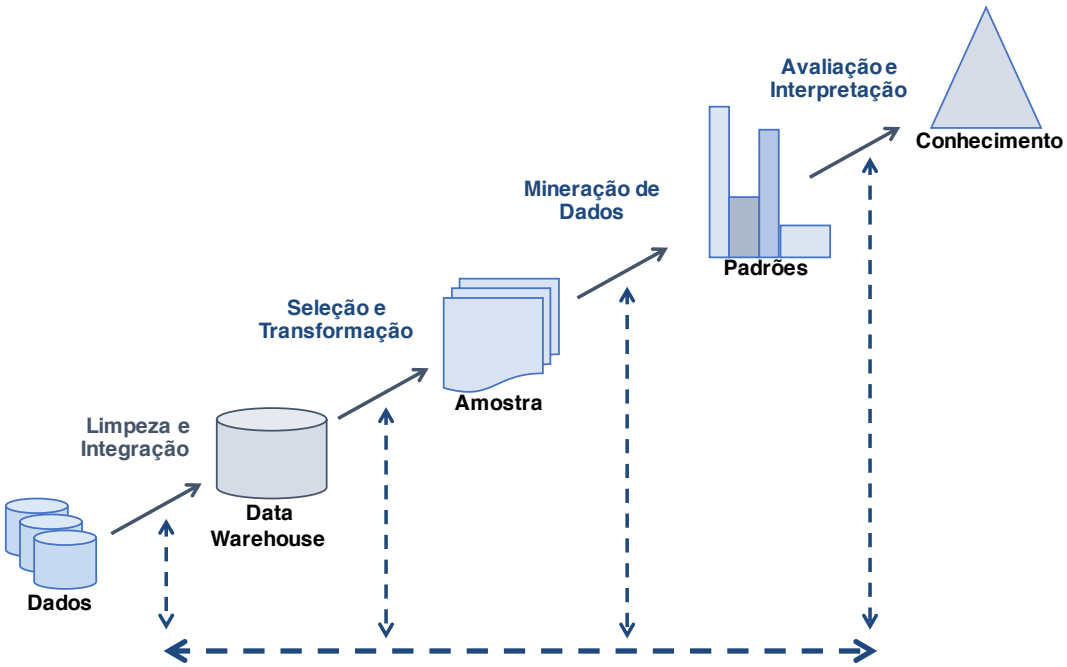


Figura 2.1 – Etapas do Processo de Descoberta de Conhecimento.

As etapas do processo estão descritas a seguir:

- Limpeza: Remoção de ruídos e de dados inconsistentes;

- Integração: Combinação de múltiplas fontes de dados;
- Seleção: Seleção dos dados relevantes para a tarefa de análise;
- Transformação: Transformação dos dados em formas adequadas para a mineração;
- Mineração de Dados: Etapa essencial em que ocorre o processo de extração de padrões e informações interessantes dos dados;
- Avaliação de Padrões: Identificação de padrões interessantes que representam o conhecimento com base em algumas medidas;
- Conhecimento: Visualização e representação do conhecimento para apresentar ao usuário.

As primeiras quatro etapas, ou seja, limpeza, integração, seleção e transformação fazem parte do pré-processamento, onde os dados são preparados para a etapa de mineração. As demais etapas fazem parte do pós-processamento [HPK11]. A mineração de dados se utiliza de técnicas da área do Aprendizado de Máquina para aprender um comportamento ou padrão na busca de possíveis informações futuras [WF11].

2.2 Aprendizado de Máquina

O Aprendizado de Máquina (*Machine learning*) é o processo de transformação dos dados em informação, sendo um instrumento que pode ser aplicado para a resolver diversos problemas [Har12]. Conforme Han e Kamber [HPK11], o Aprendizado de Máquina é a área que investiga como os computadores podem aprender automaticamente padrões e tomar decisões baseado nos dados. As principais abordagens do Aprendizado de Máquina relacionadas à mineração de dados são:

- Aprendizado supervisionado: A supervisão no aprendizado vem dos exemplos rotulados no conjunto de dados de treinamento que supervisionam o aprendizado do modelo de classificação;
- Aprendizado não supervisionado: O processo de aprendizado não é supervisionado desde os exemplos de entrada e as classes não são rotuladas. Dessa forma, o modelo precisa aprender sozinho a partir dos padrões encontrados nos dados;
- Aprendizado semi-supervisionado: O processo é feito utilizando ambas as técnicas, exemplos rotulados e não rotulados, para aprender um modelo. Os exemplos rotulados são utilizados para aprender modelos de classe e os exemplos não rotulados são utilizados para refinar os limites entre classes;

- Aprendizado ativo: Abordagem que permite aos usuários participar ativamente do processo de aprendizado. Nessa abordagem um usuário humano pode rotular um exemplo a fim de otimizar o modelo através da aquisição de conhecimento ativamente.

A Classificação e a Regressão são exemplos de Aprendizado supervisionado, pois o usuário define o que o algoritmo deve prever [Har12]. Neste trabalho, o foco são as técnicas de classificação de dados.

2.3 Classificação

A Classificação é a tarefa que visa organizar objetos em categorias pré-definidas e pode ser utilizada para diversos fins como identificação de spam, categorização de constelações, células, produtos, entre outros [TSK05]. A classificação é a tarefa de aprender uma função alvo *f* (modelo de classificação) que mapeia cada conjunto de atributos *x* para um atributo alvo *y* pré-determinado (classe). Para construir um classificador são utilizados exemplos de um conjunto de atributos definido como *x* e um atributo alvo (classe) definido como *y*, conforme apresentado na Figura 2.2:

Data de Nascimento	Local de Nascimento	Sexo	Tipo de escola	Ensino Superior
1982	Capital	M	Particular	Não
1986	Interior	F	Publica	Sim

Conjunto de atributos (*x*) Atributo alvo (*y*)

Figura 2.2 – Representação dos atributos de *x* e *y*.

Esse processo ocorre a partir da geração de um conjunto de treinamento, onde é feita a indução para a criação de um modelo de classificação e, após esse processo, o modelo é aplicado em um conjunto de teste. De acordo com Tan et al. [TSK05], existem diversas técnicas de classificação de dados, sendo as mais clássicas: Vizinhos mais Próximos, Árvores de Decisão, Naive Bayes, Redes Neurais e Vetores de Suporte.

As Árvores de Decisão são populares, pois para a construção de classificadores, não é necessário o conhecimento de um domínio ou parâmetro, sendo apropriado para descoberta do conhecimento. Elas podem lidar com grandes quantidades de dados e o aprendizado é simples e rápido, além dos classificadores terem uma boa precisão [Har12]. Com base nessa informação, este trabalho tem como foco a utilização de técnicas de classificação com Árvores de Decisão.

2.4 Árvores de Decisão

Uma árvore de decisão usa a estratégia dividir para conquistar para resolver um problema de decisão. Um problema complexo é dividido em problemas mais simples, aos quais recursivamente é aplicada a mesma estratégia. As soluções dos subproblemas podem ser combinadas, na forma de uma árvore, para produzir uma solução do problema complexo. A força dessa proposta vem da capacidade de dividir o espaço de instâncias em subespaços e cada subespaço é ajustado usando diferentes modelos. Essa é a ideia básica por trás de algoritmos baseados em árvores de decisão, tais como o ID3 e o C4.5 [Qui86, Qui96].

A classificação é feita de maneira que cada **Nodo folha** determina uma possível classe a ser escolhida em um conjunto de dados. No exemplo da Figura 2.3 o atributo alvo é definido como **sim** ou **não** para o caso de uma determinada pessoa possuir ensino **superior completo**. Para a construção de uma Árvore de Decisão são utilizados algoritmos para induzir decisões mais precisas. O algoritmo de Hunt et al. [HMS66] é um deles, sua estrutura geral é definida da seguinte forma:

- Se um conjunto de dados (D_t) só contém registros que pertencem a mesma classe (Y_t), então (t) é um nodo folha definido como (Y_t);
- Se (D_t) é um conjunto vazio, então (t) é um nodo folha definido com a classe padrão (Y_t);
- Se (D_t) contém registros que pertencem a mais de uma classe, o atributo teste é utilizado para dividir os dados em subconjuntos.

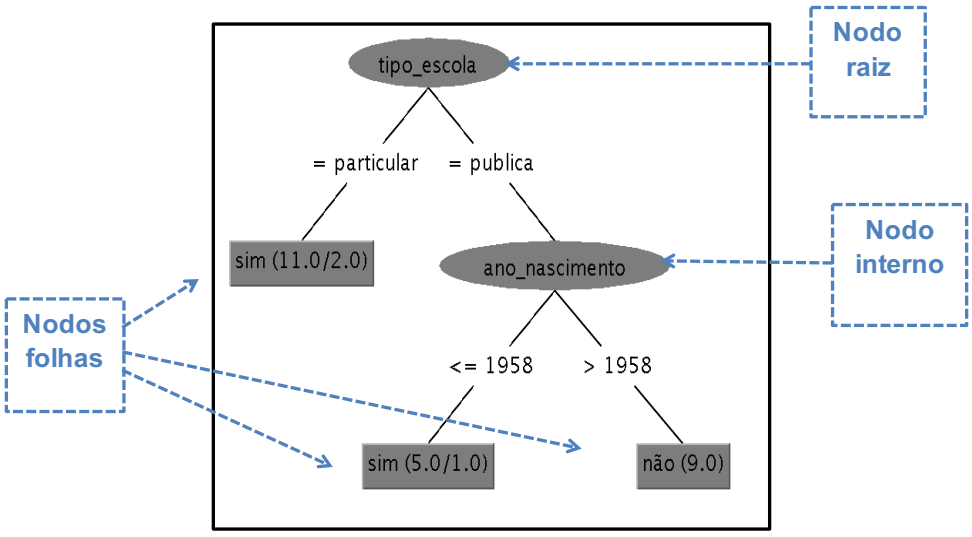


Figura 2.3 – Exemplo de Árvore de Decisão e seus tipos de nodos.

A Classificação utilizando Árvores de Decisão é uma técnica simples e muito utilizada. É constituída de um **Nodo raiz**, que possui arestas somente saindo, **Nodo interno**, que possui uma aresta chegando e duas ou mais arestas saindo, e o **Nodo folha**, que possui apenas uma aresta chegando. Na Figura 2.3 é apresentado a adaptação de um exemplo de Árvore de Decisão [TSK05]:

O algoritmo de Hunt et al. [HMS66] serve como base para o algoritmo C4.5 de Quinlan [Qui14]. Esse algoritmo cria um classificador de Árvore de Decisão constituído por uma folha que indica a classe, um nó de decisão que especifica um teste a ser realizado no valor de um atributo, e um galho para cada resposta do teste que levará para uma subárvore ou uma folha.

2.4.1 Algoritmo J48

O algoritmo J48 é um dos mais tradicionais para problemas de classificação e tem como base o algoritmo C4.5, inspirado no algoritmo ID3 [Qui86]. Ele utiliza a estratégia de divisão e conquista gerando Árvores de Decisão a partir de uma abordagem recursiva de particionamento de conjunto de dados [GP05]. Neste trabalho foi utilizada a implementação do algoritmo J48 do software Weka (*Waikato Environment for Knowledge Analysis*) versão 3.6.0 [SR14].

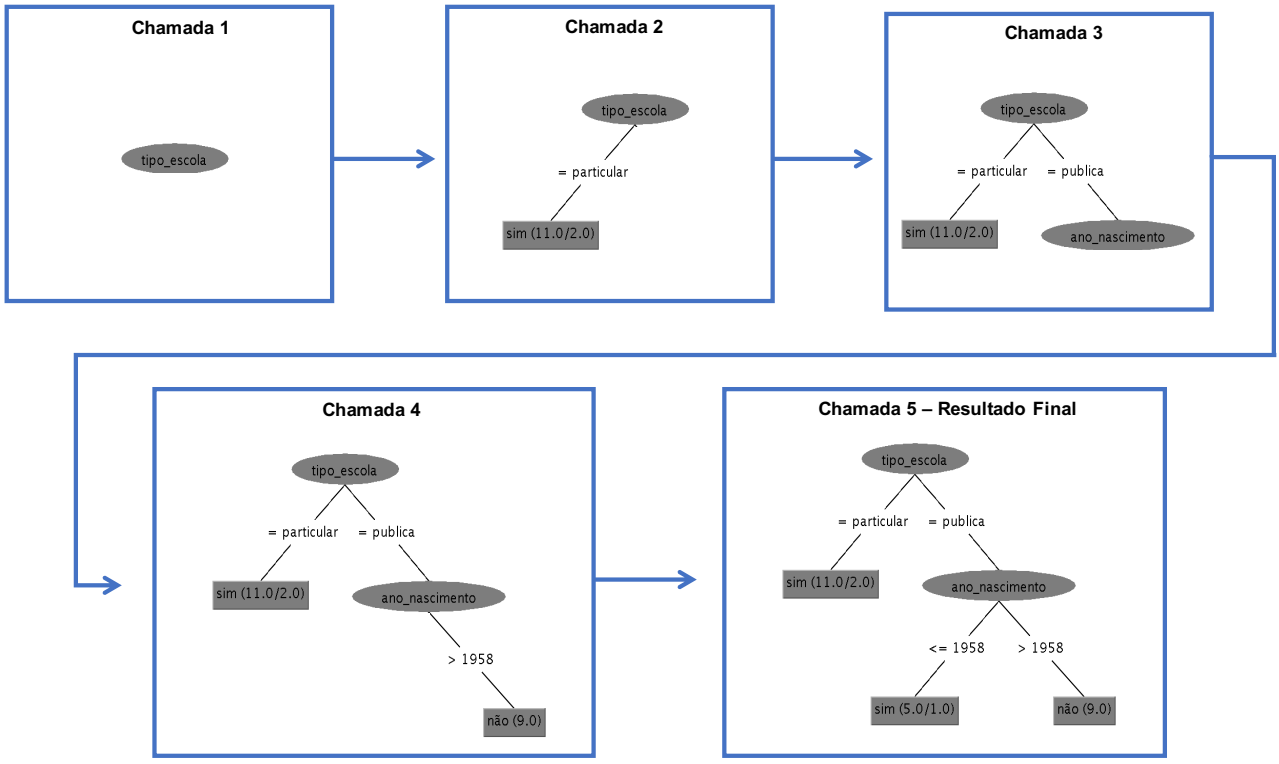


Figura 2.4 – Chamadas recursivas do método J48.

A geração da Árvore de Decisão é dada por chamadas recursivas a uma função do algoritmo C4.5 que atribuem ao nó folha uma identificação de classe. Com J48 um problema complexo é dividido em subproblemas mais simples, aplicando recursivamente a mesma estratégia a cada subproblema, associando a eles uma classe.

Esse método seleciona atributos para os nós de Árvore de Decisão pelo cálculo da entropia dos subconjuntos que escolhem os atributos que possuem um maior ganho de informação. Maiores detalhes sobre o algoritmo, fórmulas para entropia e ganho de informação podem ser encontrados em [Qui86, Qui96, LSF08, MCM13, Qui14].

Conforme Li [LC06] e Breiman [Bre96], o algoritmo C4.5 pode ser muito eficaz no processo de geração de classificadores precisos, demonstrando bons resultados em diferentes conjuntos de dados.

Na Figura 2.4 é apresentado um exemplo de funcionamento do J48 com a sequência de chamadas recursivas em que, a partir de cada chamada do C4.5, a Árvore de Decisão vai sendo montada.

2.5 Métodos de Combinação de Classificadores

Os Métodos de Combinação de Classificadores, que também são conhecidos como *Ensemble Methods*, têm como proposta melhorar os resultados da classificação dos dados [Kun04]. Trata-se de uma técnica para melhorar a precisão da classificação utilizando múltiplos classificadores em que, a partir de dados de treinamento, constrói-se um conjunto de classificadores base, com a qual seus resultados são combinados através do voto de cada classificador, sendo que a classe mais votada é a que será selecionada [TSK05]. A

O conjunto de dados original é representado por (D), as amostras geradas a partir de (D) são representadas por (D_t), (C_t) são os classificadores, e (C^*) o processo de combinação dos classificadores, que é feito através de votação.

2.5.1 Método de Boosting

O Boosting foi proposto por Schapire [Sch90] e é um método de Combinação de Classificadores que usa um procedimento iterativo para alterar de maneira adaptativa a distribuição de exemplos de treinamento de modo que os classificadores foquem em exemplos que sejam difíceis de classificar [TSK05]. De acordo com Witten et al. [WF11], é um método que combina vários modelos que complementam uns aos outros e se utiliza de votação ponderada para a classificação.

A Figura 2.5 apresenta a visão lógica do processo do método Boosting, onde inicialmente é definido um conjunto de dados de treinamento original (*D*) e, a partir dessa etapa, o processo é distribuído em três passos:

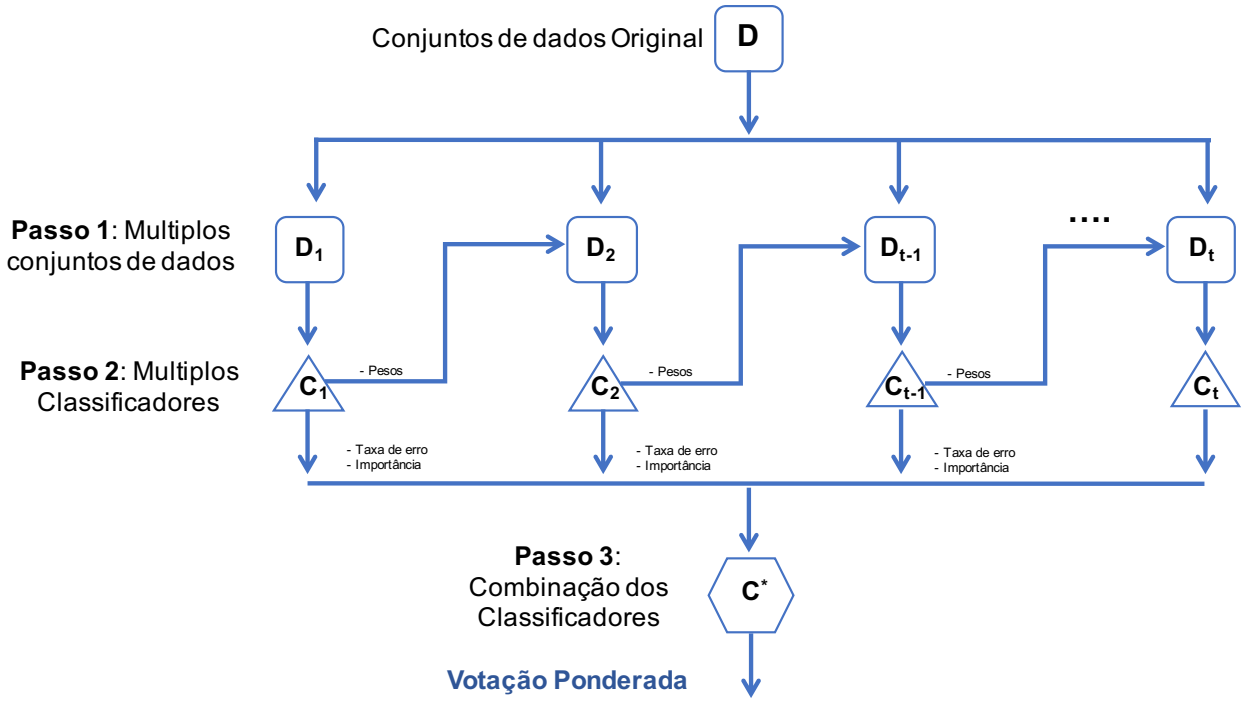


Figura 2.5 – Visão lógica do processo do método Boosting.

- No passo 1, a partir do conjunto de dados original, são geradas múltiplas amostras de dados (D_t) de tamanho igual. Inicialmente é atribuído um peso para cada instância e distribuído de maneira uniforme $1/N$;
- No passo 2, o algoritmo de aprendizado é aplicado na amostra de dados gerada no passo 1 criando um classificador (C_t), que é aplicado no conjunto de dados original. A partir disso, são calculados os valores da taxa de erro (ε), importância (α) e pesos (ω) que serão utilizados para gerar as amostras seguintes;
- No passo 3, após todas as iterações do Boosting, a combinação dos classificadores é feita a partir da importância de cada classificador (C^*), de maneira que cada voto seja dado de forma ponderada para a classificação de uma determinada classe.

De acordo com Tan et al. [TSK05], um fator importante que deve ser levado em consideração é o erro de treinamento do classificador base. Enquanto esse erro for menor que 0,5 o erro do classificador diminui exponencialmente. Por outro lado, o Boosting foca em exemplos de treinamento que são classificados incorretamente, fazendo com que a técnica do Boosting seja mais vulnerável a *overfitting*. O *overfitting* é o problema que ocorre quando um modelo de classificação tem desempenho bom no conjunto de treinamento, mas tem um desempenho ruim no conjunto de teste. Este evento é chamado de erro de generalização.

O AdaBoost (*Adaptive Boosting*) foi proposto por Freund e Shapire [FS96], sendo a implementação do método mais conhecida e utilizada para estudos. Foi projetado especificamente para classificação, podendo ser aplicado a qualquer algoritmo de Aprendizado.

Conforme Freund e Shapire [FS97], o AdaBoost utiliza múltiplos classificadores para aumentar a taxa de acerto no processo de classificação e, a partir de um conjunto de dados de treinamento, um algoritmo de aprendizado de base constrói um conjunto de classificadores base com o intuito de produzir um classificador melhor, no qual seus resultados são combinados através do voto ponderado. Uma das desvantagens do AdaBoost está na escolha do algoritmo aprendizado de base, pois se seus resultados forem ruins os resultados do AdaBoost tendem a ser piores de maneira exponencial [SF12].

O método AdaBoost foi inicialmente proposto com dois algoritmos distintos, AdaBoost.M1 e AdaBoost.M2 [FS96]. Mesmo que ambos os algoritmos tenham sido discutidos no trabalho de Freund e Schapire, apenas AdaBoost.M1 teve um grande uso em estudos pela comunidade acadêmica [FS96].

2.5.2 Algoritmo AdaBoost.M1

Em diversos contextos existe a necessidade de se trabalhar com conjuntos de dados em que o atributo alvo possui mais de duas classes. Para esses tipos de conjuntos de dados são necessários algoritmos que resolvam problemas de multi-classes, no qual o AdaBoost.M1 é um deles.

Para criar as amostras para treinamento o AdaBoost.M1 utiliza pesos para a distribuição das instâncias que inicialmente recebem valores iguais a $1/N$ e dispõe de duas implementações distintas: por reponderação e por reamostragem. Conforme Freund e Shapire [FS96], com o AdaBoost.M1 por reponderação, todo o conjunto de dados original é utilizado como amostra de tamanho fixo e, a partir dos exemplos classificados, a cada iteração são atribuídos pesos. Os exemplos classificados incorretamente têm seus pesos aumentados e os classificados corretamente têm seus pesos diminuídos. Quanto maior for a taxa de erro no classificador, maior será o peso atribuído para o próximo classificador daquele exemplo. No caso do AdaBoost.M1 por reamostragem, os exemplos são selecionados aleatoriamente para a criação de uma amostra com reposição de tamanho fixo e usados para a criação de um classificador. Os dados de treinamento são distribuídos a fim de dar uma maior probabilidade para os exemplos classificados incorretamente nas iterações anteriores.

De acordo com o algoritmo do AdaBoost.M1 de Freund e Shapire [FS96],após a primeira iteração o conjunto de dados original é classificado, então é calculada a taxa de erro (ε), importância (α) e pesos (ω), de maneira que as instâncias classificadas incorretamente têm seus pesos aumentados, e as classificadas corretamente têm seus pesos diminuídos, dando maior chance de seleção aos exemplos classificados incorretamente. No

AdaBoost.M1, caso o erro de treinamento do classificador seja superior ao limiar de 0,5, o processo de iteração é interrompido, e o voto ponderado é calculado com os classificadores criados até aquele momento.

A exemplo do proposto por Lopes [LSF08], foi descrito de forma detalhada a aplicação do algoritmo AdaBoost.M1. Supondo que um conjunto com N instâncias seja $\left\{(x_j, y_j) \mid j = 1, 2, ..., N\right\}$, os cálculos dos valores são apresentados a partir das Equações de Taxa de erro (2.1), Importância (2.2) e Pesos (2.3).

$$\varepsilon = \frac{1}{N}\left[\sum_{N}^{j=1} \omega_j I(C_t(x_j))\right] \tag{2.1}$$

$$\alpha = \frac{1}{2} \ln \frac{(1-\varepsilon)}{\varepsilon} \tag{2.2}$$

$$\omega^{(j+1)} = \frac{\omega^j}{Z_j} \times \left\{ \begin{array}{l} Euler^{-\alpha_j} \\ Euler^{+\alpha_j} \end{array} \right\} \tag{2.3}$$

No caso do cálculo dos pesos é necessário normalizá-los de forma que sua soma continue igual a 1, dividindo os resultados pela Equação (2.4).

$$Z_j = \omega^{j+1} = 1 \tag{2.4}$$

As instâncias das amostras são selecionadas aleatoriamente com reposição, podendo haver amostras repetidas. Esta função é específica do AdaBoost.M1 por reamostragem de forma que na implementação AdaBoost.M1 por reponderação o processo de reamostragem não ocorre, utilizando sempre a mesma amostra original do conjunto de dados.

2.6 Validação Cruzada

A Validação Cruzada (*Cross-validation*) é uma técnica que um conjunto de dados original é dividido em subconjuntos de tamanhos iguais, onde um subconjunto é utilizado para teste e os demais para treinamento, estimando a taxa de acerto. No caso deste trabalho, o objetivo é avaliar a taxa de acerto dos classificadores.

Segundo Tan et al. [TSK05], a Validação Cruzada é uma técnica sofisticada para avaliar o desempenho de um classificador durante a fase de treinamento. A validação Cruzada consiste na troca de papéis entre os subconjuntos iguais. No caso da divisão em dois subconjuntos, um deles é usado para teste e outro para treinamento. Após este processo os papéis são invertidos, e o conjunto de teste passa a ser o de treinamento e vice-versa.

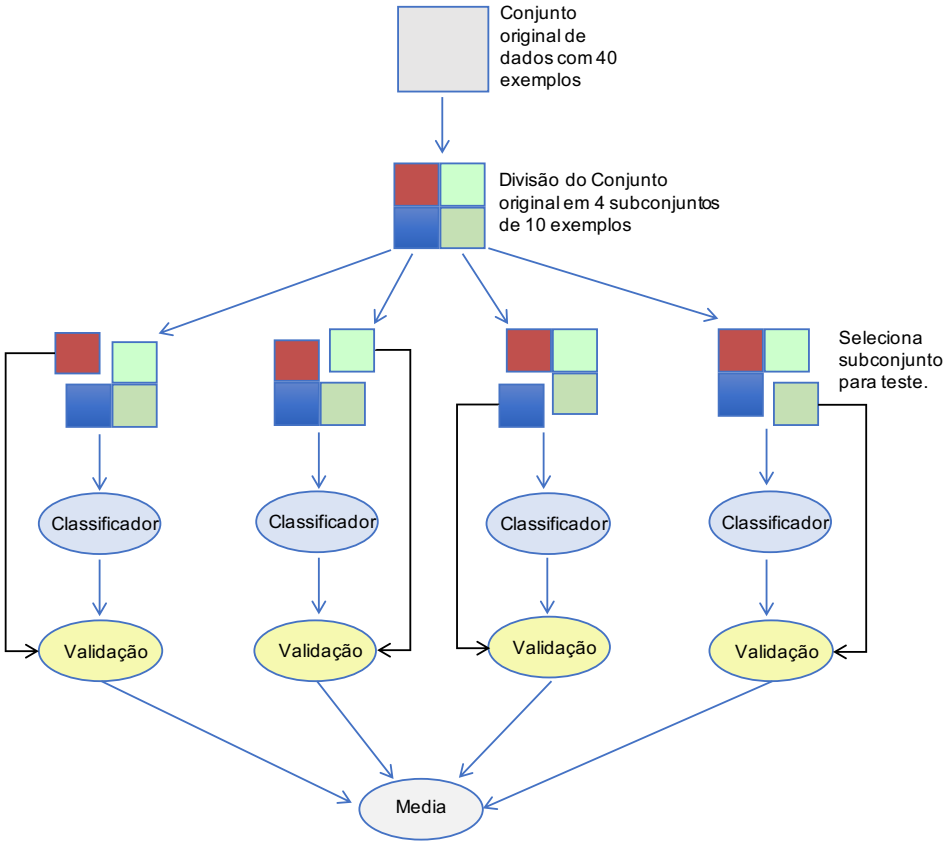


Figura 2.6 – Exemplo de Validação Cruzada para 4 folds.

Neste trabalho, utilizou-se a técnica de Validação Cruzada de k partições de tamanhos iguais (k folds), onde os subconjuntos são divididos em amostras iguais com k números de exemplos. O resultado final é dado pela média dos resultados obtidos dos subconjuntos de acordo com a Figura 2.6.

De acordo com Diamantidis et al. [DKG00] a Validação Cruzada é uma técnica utilizada para estimar a taxa de acerto de classificadores, sendo uma alternativa mais aprimorada para diminuir os impactos de técnicas mais simples. É uma alternativa que resolve o problema da geração de resultados tendenciosos que ocorrem quando a totalidade de um conjunto original de dados é utilizada tanto para teste como para treinamento. Isso também se aplica quando é empregada a uma parte do conjunto original para teste e o restante para treinamento, como ocorre na técnica de divisão por porcentagem (*Percentage Split*).

Como foi visto, a Validação Cruzada pode ser utilizada em qualquer método de classificação de dados, como por exemplo, no método de combinação de classificadores Boosting visto na seção anterior. Neste trabalho, a Validação Cruzada é a principal técnica utilizada como base para a análise dos resultados obtidos.

2.7 Processos Estocásticos

Processos Estocásticos são uma coleção de variáveis aleatórias. De acordo com Ross [Ros96], é a representação de sistemas nos quais o estado muda ao longo do processo. Estas mudanças são relacionadas à distribuição de probabilidade. Existem vários processos reais que utilizam processos estocásticos. Um exemplo comum que resulta em um processo estocástico é o processo de jogar uma moeda ou o de lançar um dado de 6 lados, pois qualquer um dos lados da moeda ou do dado tem a mesma probabilidade de ficar com uma das faces para cima.

Para Clarke e Disney [ABC79], o processo estocástico é um fenômeno que varia de maneira imprevisível na medida em que o tempo passa, produzindo uma sequência de resultados diferentes.

O processo de classificação de dados é determinado pela aleatoriedade de seleção dos dados, em específico com a utilização de números pseudoaleatórios. Conforme Tan et al. [TSK05], um experimento aleatório é a ação para avaliar um processo no qual o resultado é incerto.

Os processos físicos, tais como jogar uma moeda ou um dado podem ser considerados determinísticos pelo fato de se conhecer sua condição inicial, mas podem ser tendenciados por fenômenos físicos [Ros06].

Conforme Ross [Ros96], a geração de números aleatórios é a base para qualquer sistema de simulação estocástica. Na área da computação, são utilizados para simulação estocástica os algoritmos geradores de números pseudoaleatórios, que produzem uma sequência de números inteiros imitando uma sequência de variáveis aleatórias independentes e uniformemente distribuídas. O fundamental para a geração de uma sequência de números pseudoaleatórios é a sua imprevisibilidade, no sentido de que ninguém é capaz de visualizar seu próximo passo, evitando um processo tendencioso.

A geração de números pseudoaleatórios utilizado na seleção de instâncias para a classificação dos dados é baseada na técnica de Bagui [Bag05], que define a aleatoriedade como um número que pertence a uma sequência numérica e que não pode ser previsto a partir dos números anteriores da sequência. Para inicializar a geração dos números aleatórios é necessário utilizar uma semente de inicialização, podendo ela ser escolhida de forma arbitrária.

2.8 Testes de Hipóteses

De acordo com Faceli et al. [FLGC11], determinar se um algoritmo é melhor do que outro pelo simples exame de superioridade e inferioridade de médias não é aconselhável,

de maneira que na maioria das vezes as diferenças analisadas não são significativas, sendo necessário um teste de hipóteses para comparar os modelos estudados.

Para Devore [Dev10], se μ_1 e μ_2 são os resultados médios de dois classificadores obtidos a partir da validação cruzada, uma hipótese possível para a expressão é $\mu_1 - \mu_2 = 0$, ou seja, pode-se considerar que não existe diferença entre elas. A outra é que $\mu_1 - \mu_2 > 0$, logo μ_1 é superior a μ_2, ou dependendo da comparação necessária $\mu_1 - \mu_2 < 0$, logo μ_1 é inferior a μ_2. Nos testes de hipóteses duas suposições contrárias são levantadas, Hipótese nula H_0: $\mu_1 - \mu_2 = 0$ e Hipótese alternativa H_1: $\mu_1 - \mu_2 \neq 0$. A hipótese nula é assumida como a verdadeira, de forma que pode ser rejeitada em favor da hipótese alternativa, se algumas evidencias demonstrarem que a hipótese nula é falsa.

A avaliação estatística dos resultados experimentais é considerada uma parte essencial na validação de novos métodos de Aprendizado de Máquina [Dem06]. Os testes estatísticos devem ser executados corretamente e as conclusões resultantes devem ser extraídas cautelosamente. De acordo com as características de cada experimento deve-se escolher um teste estatístico paramétrico (*teste t*, ANOVA) ou não-paramétrico (Wilcoxon, Friedman, Teste de sinal).

Conforme Demšar [Dem06], baseado nas propriedades estatísticas conhecidas de cada teste, nos seus pressupostos, e no conhecimento dos dados do Aprendizado de Máquina, concluiu-se que os testes não-paramétrico devem ser preferidos em relação aos paramétricos, pois são mais prováveis para rejeitar a hipótese nula. Por outro lado, indica que com uma amostra maior é possível utilizar testes paramétricos como o *Teste t de Student* ou *Teste t*. Neste trabalho utilizamos para amostras relacionadas o *Teste t*.

De acordo com Magalhães e Lima [Mag00], o *Teste t* pode ser utilizado para verificar grupos avaliados sob diferentes condições extraídos de uma mesma população (hipótese nula). A hipótese alternativa deve ser definida em 3 formas:

- H_1: $\mu_1 \neq \mu_2$ (Teste bi-caudal);
- H_1: $\mu_1 > \mu_2$ (Teste uni-caudal à direita);
- H_1: $\mu_1 < \mu_2$ (Teste uni-caudal à esquerda).

Para identificar as áreas de rejeição dessas hipóteses os gráficos das respectivas alternativas são apresentados na Figura 2.7.

O valor de *t* é estimado e o *t* crítico é determinado, indicando a área de rejeição de H_0. A partir do cálculo do *t* estimado calcula-se o *p-valor*. De forma genérica, o resultado numérico do *Teste t* fornece o nível de significância da afirmação *p-valor*. *P-valores* pequenos fornecem evidências para rejeitarmos a hipótese nula em favor da hipótese alternativa. Ao contrário, *p-valores* grandes fornecem evidências para não rejeitarmos a hipótese nula. Desta forma, é necessário estabelecer um valor máximo de tolerância de aceitação, determinando o nível de significância do teste. Normalmente o valor determinado fica entre

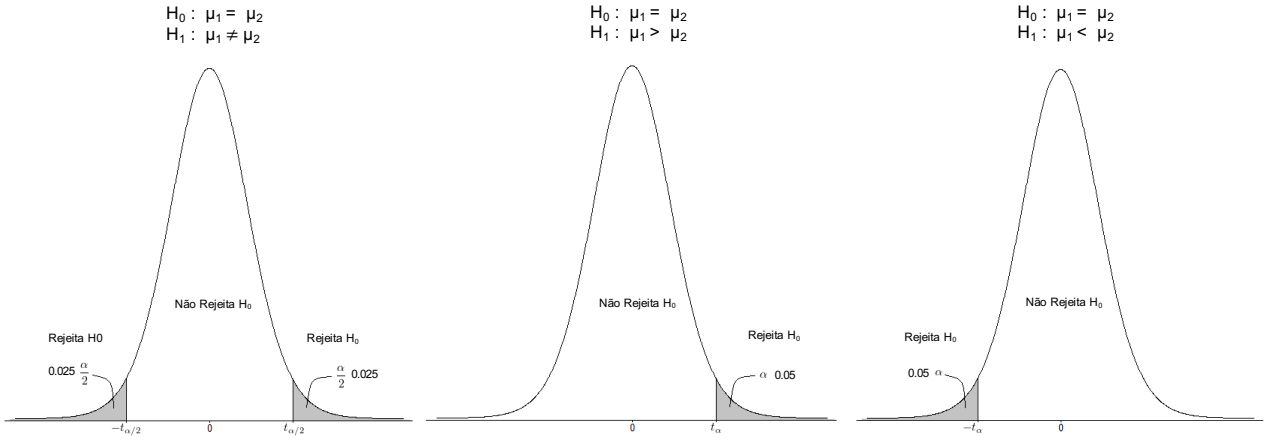


Figura 2.7 – Exemplo de gráficos das zonas de rejeição do *Teste t*

1% e 5% (α). Se Valor $p \leqslant \alpha$, a diferença entre as médias é estatisticamente significativa, rejeitando H_0. Por outro lado se o Valor $p > \alpha$, a diferença entre as médias não é estatisticamente significativa, não rejeitando H_0.

3. TRABALHOS RELACIONADOS

Neste capítulo são apresentados alguns trabalhos relacionados que detalham aplicações e experimentos utilizando técnicas de combinação de classificadores que, em alguns casos, levam em consideração aspectos estocásticos. Também são apresentadas implementações de extensões para o algoritmo AdaBoost que buscam melhorar os resultados obtidos no processo de aprendizado e de classificação. A principal relação desses trabalhos com esta dissertação está na questão da proposta de uma extensão do algoritmo que leve em consideração aspectos estocásticos.

Fernandes et al. [FLR10] analisa os métodos de combinação de classificadores Bagging e Boosting quanto à relevância da aleatoriedade das escolhas de amostras e ao impacto da falta de cuidado nos experimentos realizados. Foram utilizados como referência 32 conjuntos de dados de repositórios públicos e configurou-se os algoritmos com sementes aleatórias de 1 a 100 para verificar o impacto dessa aleatoriedade nos resultados. Para os experimentos considerou-se como parâmetros de comparação entre os métodos a taxa de acerto, o desvio padrão e a distância entre o menor e maior resultado. Este trabalho demostrou a importância de considerar os impactos da aleatoriedade nos resultados dos métodos de combinação quando são estudadas.

Fernandes et al. [FLNR13] demonstra que é possível desenvolver uma proposta de extensão do método Random Forests considerando aspectos estocásticos. Este método de combinação de classificadores é sensível à geração de número pseudoaleatórios. Baseado nisso, foi proposta uma extensão para o algoritmo, chamada de Stochastic Aware Random Forests (SARF), que visa tratar da vulnerabilidade do Random Forests quando submetido a aleatoriedade. Para os experimentos do Random Forests Estocástico foram utilizados como referência os mesmos 32 conjuntos de dados de Fernandes et al. [FLR10], assim como os parâmetros de comparação entre os métodos foram a taxa de acerto, desvio padrão e distância. Os resultados dos experimentos demonstraram, de maneira geral, que o método Random Forests Estocástico foi mais eficaz em relação ao método original do Random Forests e outros métodos de combinação, incluindo o Boosting.

Esses trabalhos têm grande relevância no desenvolvimento desta dissertação, pois consideram aspectos estocásticos utilizando métodos de combinação de classificadores. O presente trabalho propõe a mesma forma de processo de geração de amostras aleatória, mas com o foco no algoritmo AdaBoost. Além disso, propõe-se a utilização das mesmas métricas para avaliação dos métodos: a taxa de acerto, desvio padrão e distância.

Lopes et al. [LSF08] analisa a eficácia dos métodos J48, Bagging e Boosting em termos de comportamento de taxa de acerto dos classificadores gerados pelos mesmos. Para análise dos resultados foram utilizadas as técnicas de Validação Cruzada e o teste estatístico de Friedman. Nesse trabalho, o foco está em sugerir soluções eficazes para

a descoberta de conhecimento em conjuntos de dados de um seguimento específico sob uma visão de curvas de aprendizagem dos classificadores. Outro foco desse trabalho é a verificação do impacto da variação do número de classificadores na eficácia dos métodos. A partir da geração e análise de diversas curvas de aprendizagem, o método Boosting demonstrou ser mais eficaz em relação ao J48 e Bagging.

Thongkam et al. [TXZ08] propõe a combinação do método AdaBoost e Random Forests para construir um modelo de predição para análise de conjuntos de dados com informações do câncer de mama. Para o processo iterativo do AdaBoost foi utilizado como algoritmo de aprendizado base o Random Forests, com o objetivo de melhorar as taxas de acerto, estabilidade e problemas de *overfitting*. A capacidade do método foi avaliada a partir da taxa de acerto, sensibilidade e especificidade. O resultado do experimento determinou que o método proposto superou J48, AdaBoost e o Random Forests.

Esses trabalhos apresentam características de comparação de métodos de classificação e modos de utilização com o objetivo de melhorar os resultados obtidos. Sempre levando em consideração a taxa de acertos, que demonstra ser uma métrica relevante para medir a eficácia dos métodos.

Milidiú e Duarte [MD09] propõem uma extensão do algoritmo AdaBoost chamada de Boosting at Start (BAS). Este algoritmo tem como objetivo principal a distribuição inicial de exemplos das amostras de maneira arbitrária. Os resultados demonstram que é possível aumentar a taxa de acerto de forma significativa em relação a distribuição inicial uniforme, além de serem necessárias menos iterações para alcançar bons resultados. Neste trabalho, a eficácia dos métodos foi comparado e medido pelo critério de taxa de acerto.

Solomatine e Shrestha [SS04] propõem o AdaBoost.RT, extensão do algoritmo Boosting para Problemas de Regressão (AdaBoost.R) de Freund e Schapire [SF+99]. Esse algoritmo foi criado com o intuito de melhorar os resultados de problemas de regressão utilizando o AdaBoost, e tem como foco a utilização de uma constante como limiar de erro relativo para delimitar o erro de treinamento gerado pelos classificadores. Isso possibilita que, na iteração seguinte, o número de exemplos classificados incorretamente diminuam.

Friedman [Fri01] propõe uma estrutura estatística que leva a direta interpretação de Boosting como um método para estimação funcional chamado de Gradient Boosting. Esse método foi desenvolvido para problemas de classificação e regressão, e consiste em estimar uma função de predição ótima a partir de uma função de perda. Trata-se da geração de um conjunto de modelos de classificação simples, permitindo a otimização de uma função de perda no processo de construção dos classificadores a partir dos dados de entrada. Nessa abordagem, o foco é a minimização da função de perda que diminui a diferença entre os valores previstos e medidos. Para cada classe são geradas pontuações e os resultados são combinados na função, de forma que os pesos são atualizados gradualmente e atribuídos individualmente a cada uma das variáveis. Essa proposta apresenta melhora dos resultados em relação a implementação do AdaBoost Original.

Friedman [Fri02], a partir dos estudos anteriores como [FHT+00] e [Fri01], que promoveram grandes avanços do ponto de vista estatístico para o algoritmo AdaBoost, propõe um modelo chamado de Stochastic Gradient Boosting. Essa implementação tem como foco levar em consideração aspectos estocásticos de forma que, a cada iteração, uma sub-amostra dos dados de treinamento seja gerada de maneira aleatória sem substituição. Essa sub-amostra é utilizada no lugar da amostra completa e se ajusta ao classificador simples, atualizando os pesos do modelo para a iteração corrente.

Os estudos Friedman [Fri01] e Friedman [Fri02] se diferenciam deste presente trabalho por terem como foco no modelo de Boosting implementando uma estrutura estatística dentro do método. Também propõe o formato de geração de sub-amostras dentro do processo de geração dos classificadores. Esta dissertação considera aspectos estocásticos quanta à aleatoriedade da seleção de sementes, diferenciando-se desses trabalhos.

Em Zhu et al. [ZZRH09], é apresentada uma proposta de extensão para o algoritmo Boosting para problemas multi-classes denominada SAMME (*Stagewise Additive Modeling using a Multi-class Exponential loss function*), que dispensa a redução de problemas de multi-classes para vários subproblemas de duas classes. Este algoritmo é diferente da implementação original, pois requer que os resultados de cada classificador base sejam melhores que escolhas aleatórias, ou seja, considera o limiar de erro como o inverso do número de classes. No entanto o trabalho de Zhu et al. [ZZRH09] considera como classificador base o Bayes rule.

Analisando de maneira geral, todos os trabalhos relacionados apresentados mostraram diferentes possibilidades de implementação e experimentação utilizando o método Boosting, por isso, são utilizados como base para o desenvolvimento desta dissertação. Na Tabela 3.1 são apresentadas as referências e características dos trabalhos relacionados acima estudados.

Tabela 3.1 – Características dos trabalhos relacionados

Autor	Referência	Características relacionadas	Diferenças
Friedman	[Fri01]	- AdaBoost.M1 considerando aspectos estocásticos.	- Os aspectos estocásticos vem da utilização de sementes aleatória de 1 até 100.
Friedman	[Fri02]	- AdaBoost.M1 considerando aspectos estocásticos.	- Os aspectos estocásticos vem da utilização de sementes aleatórias de 1 até 100.
Solomatine e Shrestha	[SS04]	- Utilização de uma constante como limiar de erro relativo para delimitar o erro de treinamento.	- Utilização sem o limiar de erro, com o inverso do número de classes do limiar de erro e com valores absolutos.
Lopes et al.	[LSF08]	- Comparação de métodos de combinação de classificadores, incluindo o AdaBoost.M1; - Taxa de acerto como critério de comparação; - Aplicação de testes de hipóteses; - J48 como algoritmo base.	- Foco apenas no algoritmo AdaBoost.M1.
Thongkam et al.	[TXZ08]	- Comparação de métodos de combinação de classificadores, incluindo o AdaBoost.M1; - Taxa de acerto como critério de comparação; - Utilização do J48 como algoritmo base.	- Foco apenas no algoritmo AdaBoost.M1;
Milidiú e Duarte	[MD09]	- Comparação de métodos de combinação de classificadores, incluindo o AdaBoost.M1; - Taxa de acerto como critério de comparação; - Distribuição inicial dos pesos dos exemplos das amostras de maneira arbitrária.	- Distribuição inicial dos pesos dos exemplos padrão sem alteração.
Zhu et al.	[ZZRH09]	- Dispensa a redução de problemas de multiclasses para vários subproblemas de duas classes; - Limiar de erro como o inverso de número de classes.	- Aplicação de outras abordagens como a eliminação do limiar de erro e distribuição de limiares arbitrária.
Fernandes et al.	[FLR10]	- Análise dos impactos da aleatoriedade de sementes; - Comparação de métodos de combinação de classificadores, incluindo o AdaBoost.M1; - Taxa de acerto como critério de comparação; - Experimentos com 32 conjuntos de dados de repositórios públicos: University of California Irvine [AN14] e University of West Virginia [BMO14]; - J48 como algoritmo base.	- Utilização apenas do AdaBoost.M1 e do J48.
Fernandes et al.	[FLNR13]	- Análise de aspectos estocástico quanto a aleatoriedade das sementes; - Adaptação do método de combinação Random Forests; - Comparação de métodos de combinação de classificadores, incluindo o AdaBoost.M1; - Taxa de acerto como critério de comparação; - Experimentos com 32 conjuntos de dados de repositórios públicos: University of California Irvine [AN14] e University of West Virginia [BMO14]; - J48 como algoritmo base.	- Utilização apenas do AdaBoost.M1 e do J48.

4. ADAPTAÇÃO DO ALGORITMO ADABOOST.M1

Estudos anteriores foram realizados com o objetivo de propor melhorias para diversos métodos de classificação [Bot01, Fri01, Fri02, SS04, LSF08, TXZ08, SKHN08, MD09, FLR10, YM12, FLNR13, LFLA17], dos quais foram analisadas diferentes abordagens comparando seus resultados, em específico o algoritmo AdaBoost.M1, o que auxilia na construção dos experimentos. Centrada na possibilidade de melhorar os resultados com este algoritmo, a atividade de investigação realizada neste trabalho toma a direção de um estudo experimental envolvendo a implementação do algoritmo AdaBoost.M1. Tal estudo visa propor uma adaptação para este algoritmo com a finalidade de melhorar a taxa de acerto e o tempo de execução dos classificadores. A configuração deste experimento testou, sobre um grupo de conjuntos de dados, o comportamento do algoritmo adaptado comparando com o algoritmo original AdaBoost.M1, considerando aspectos estocásticos nos resultados do processo de geração dos classificadores, em específico, na seleção aleatória dos exemplos das amostras de dados.

Neste trabalho foram considerados três aspectos fundamentais:

- A escolha dos conjuntos de dados alvo;
- A escolha do ambiente de teste de software e hardware;
- O cuidado da relevância estatística dos resultados.

Neste capítulo, apresenta-se a experimentação utilizada para sugerir uma adaptação para o AdaBoost.M1. Dessa forma, as seções a seguir apresentam os conjuntos de dados utilizados para os experimentos (Seção 4.1), configuração geral dos experimentos (Seção 4.2), configuração específica dos experimentos (Seção 4.3) e definição dos Testes de Hipóteses (Seção 4.4).

4.1 Conjuntos de dados

Os trinta e dois conjuntos de dados selecionados para os experimentos deste trabalho foram retirados de repositórios públicos de diversos contextos e possuem diferentes características tais como, informação médica, softwares, reconhecimento do vinho, biologia, química, fenômenos físicos, entre outros. Esses conjuntos de dados foram escolhidos por incluírem uma grande variedade de dados de diferentes tipos, contribuindo para que os resultados dos experimentos tenham uma perspectiva geral.

A primeira coluna da Tabela 4.1 apresenta a identificação do arquivo do conjunto de dados (*ID*). A segunda coluna apresenta o nome do conjunto de dados (*BD*) e o repo-

sitório original, sendo △ para University of California Irvine [AN14] e ▽ para University of West Virginia [BMO14]. As demais colunas apresentam as informações sobre os conjuntos de dados como número de atributos (*NA*), número de instâncias (*NI*), número de classes do atributo alvo (*NC*) e a taxa de desbalanceamento (*TD*).

Tabela 4.1 – Conjuntos de dados

ID	*BD*	*NA*	*NI*	*NC*	*TD*
B01	Abalone△	9	4177	29	0,071
B02	Arrythmia△	280	452	13	0,520
B03	Audiology△	70	226	24	0,103
B04	Balance△	5	625	3	0,146
B05	Breast cancer△	10	286	2	0,165
B06	Car Evaluation△	7	1728	4	0,390
B07	CM1 software defect▽	22	498	2	0,645
B08	Datatrieve▽	9	130	2	0,690
B09	Desharnais▽	12	81	3	0,150
B10	Ecoli△	9	336	8	0,168
B11	Echo cardiongram△	12	132	3	0,054
B12	Glass△	11	214	6	0,116
B13	Heart (Cleveland)△	14	303	2	0,008
B14	Heart statlog△	14	270	2	0,012
B15	Hepatitis△	20	155	2	0,345
B16	JM1 software defect▽	22	10885	2	0,376
B17	Kr-vs-kp△	37	3196	2	0,002
B18	MW1 software defect▽	38	403	2	0,716
B19	Pima-diabetes△	9	768	2	0,091
B20	Post-operative△	9	90	3	0,366
B21	Primary-tumor△	18	339	21	0,066
B22	Reuse▽	28	24	2	0,063
B23	Solar Flare△	13	1389	8	0,682
B24	Tic-Tac-Toe Endgame△	10	958	3	0,094
B25	Thyroid (Allhyper)△	30	2800	4	0,928
B26	Thyroid (Hypothyroid)△	30	3772	4	0,807
B27	Thyroid (Sick euthyroid)△	26	3163	2	0,664
B28	Wbdc△	31	569	2	0,065
B29	Wisconsin breast cancer△	10	699	2	0,096
B30	Wine recognition△	14	178	3	0,065
B31	Yeast△	10	1484	10	0,137
B32	Zoo△	18	101	7	0,114

A taxa do desbalanceamento (*TD*) foi proposta por Fernandes et al. [FLR10], e seu valor numérico é calculado pela equação (4.1):

$$TD = \frac{\left(\frac{STD}{(NI/NC)}\right)}{\sqrt{NC}} \tag{4.1}$$

Trata-se da razão entre o desvio padrão do número de casos em cada classe (*STD*), por uma distribuição completamente equilibrada dos casos entre as classes (NI/NC) e dividida pela raiz quadrada do número de classes (*NC*). Dessa forma, a taxa de desbalanceamento varia entre 0 e 1. Por exemplo, ao se observar a Tabela 4.1, percebe-se que o conjunto B17 possui uma taxa de desbalanceamento baixa, igual a 0,002, pois suas instâncias estão divididas pelas duas classes em 1669 e 1527. No caso de B25, a taxa de desbalanceamento é alta, pois suas instâncias estão divididas pelas suas quatro classes em 62, 8, 7 e 2723. Conforme Fernandes et al. [FLR10], a taxa de desbalanceamento pode ser vista como o inverso da entropia do conjunto de dados.

Esses conjuntos são amplamente utilizados para experimentos em outros estudos na área acadêmica, de forma que não foram inventados de maneira tendenciosa. Naturalmente o algoritmo AdaBoost.M1 faz uma amostragem das instâncias para construir cada classificador, por conseguinte, cada amostra gerada, a partir do conjunto original, pode ter graus de desbalanceamento muito distintos, reduzindo ou aumentando o desbalanceamento. Esse fato reduz drasticamente a possibilidade de geração de um classificador simplista que retorna somente a classe majoritária.

4.2 Configuração geral dos experimentos

Para a execução de todos os experimentos foi utilizada a implementação do algoritmo AdaBoost.M1 do software Weka (*Waikato Environment for Knowledge Analysis*) versão 3.6.0 [SR14]. Todas as modificações no algoritmo foram feitas no código fonte, inclusive a parametrização dos experimentos sem a utilização de interface gráfica.

De acordo com Monard e Baranauskas [MB03], a avaliação de um algoritmo de aprendizado de máquina supervisionado é normalmente realizada por meio da análise do desempenho do preditor gerado por ele no processo de rotulação do atributo alvo, e isso pode ser feito através da matriz de confusão, alternativa para visualizar o desempenho de um classificador através do número de predições corretas e incorretas em cada classe. Baseado nisso, o tempo de execução da classificação dos dados foi estimado em segundos, enquanto que a taxa de acerto dos classificadores utilizou-se a matriz de confusão e a técnica de Validação Cruzada *Ten-folds cross-validation stratified*. Essa técnica permite obter estimativas mais precisas, além de reduzir a variação dos resultados.

Os testes foram executados aplicando as seguintes configurações para cada conjunto de dados:

- Classificador Base: Algoritmo J48;
- Iterações (Classificadores): 50;
- Validação Cruzada: Padrão = 10 *Folds*;
- Implementação: Reamostragem.

O número de classificadores utilizados foi determinado dessa forma para que os experimentos se mantivessem em um tempo de execução aceitável, e para que houvesse uma margem considerável para análise dos resultados, da mesma maneira que nos estudos de Lopes et al. [LSF08], Fernandes et al. [FLR10], Fernandes et al. [FLNR13] e Leães et al. [LFLA17]. Além disso, Freund e Schapire [FS96] analisaram o algoritmo quanto aos limites de probabilidades de erro nos conjuntos de treinamento e de teste. Um dos limites teóricos

demonstra que o erro de treinamento diminui exponencialmente com o número de iterações do AdaBoost.M1. Isso evidencia que, após algumas iterações, o erro de treinamento pode cair a zero. Conforme Buhlmann e Hothorn [BH07], se o algoritmo AdaBoost.M1 for executado continuamente sem um limite, o erro do conjunto de teste continua a decrescer, indicando que o algoritmo é suscetível ao overfitting.

4.3 Configurações específicas dos experimentos

Para a experimentação deste trabalho, o foco foi observar quatro cenários específicos: experimentação a fim de verificar o comportamento dos conjuntos de dados quanto à aleatoriedade (Seção 4.3.1); experimentação na tentativa de verificar se o limiar de erro de treinamento apresentava alguma relação com o número de classes (Seção 4.3.2); experimentação para verificar a existência de algum padrão relacionado aos valores absolutos do limiar de erro de algum resultado melhor (Seção 4.3.3); e a experimentação com o foco na análise do comportamento do algoritmo sem utilizar o limiar de erro (Seção 4.3.4).

4.3.1 AdaBoost.M1 submetido a aleatoriedade

Com objetivo de obter um melhor entendimento sobre os impactos da aleatoriedade na seleção das amostras para treinamento no primeiro cenário de experimentação, a adaptação do algoritmo AdaBoost.M1 foi baseada na mesma forma que os experimentos executados por Fernandes et al. [FLR10], Fernandes et al. [FLNR13] e Leães et al. [LFLA17], em que foram levados em consideração aspectos estocásticos quanto aos resultados da seleção de sementes aleatórias executadas 100 vezes, de 1 até 100, variando de 1 em 1.

Os resultados obtidos nas taxas de acerto e nos tempos de execução foram agrupados em um único valor para cada um. Para os resultados foram removidos os 5 melhores e os 5 piores resultados de cada conjunto de dados. Esse procedimento evita que algum resultado com um desvio considerável exerça um peso sobre o cálculo do percentual da média das taxas de acerto e do tempo de execução. Além disso, acredita-se que dessa forma os resultados não estão contaminados por fatores casuais, como por exemplo, alguma semente aleatória particularmente benéfica ou maléfica para a taxa de acerto. Para os 90 resultados restantes, as médias foram calculadas. Esses valores são fundamentais para este trabalho, pois é por meio deles que os resultados dos algoritmos são medidos e comparados.

Para fazer uma análise comparativa da aleatoriedade foi executada uma rodada para cada um dos conjuntos de dados utilizando os parâmetros do algoritmo AdaBoost.M1 com a configuração padrão (1) e com sementes aleatórias (1 a 100).

4.3.2 AdaBoost.M1 quanto ao número de classes

Tendo em vista que, em problemas de duas classes, a probabilidade de escolha de qualquer uma das classes é de 50%, o limiar de erro de treinamento foi determinado pelo condicional de ser maior que o valor 0,5. Baseado nisso, o limiar de erro para problemas multi-classes deve ser proporcional a probabilidade de seleção de classes de um determinado conjunto de dados, ou seja, requer que os resultados de cada classificador base sejam melhores que escolhas aleatórias, da mesma forma que no trabalho de Zhu et al. [ZZRH09], onde o limiar de erro é considerado como o inverso do número de classes.

Tendo essa questão como motivação, a tentativa inicial de experimentação foi de verificar se o erro de treinamento apresentava alguma relação com o número de classes dos conjuntos de dados. Dessa forma, foi proposta uma adaptação para o algoritmo AdaBoost.M1, considerando aspectos estocásticos definidos como AdaBoost.M1-AL. A execução dos experimentos foi realizada sem fazer nenhuma alteração no funcionamento básico do AdaBoost.M1. O código fonte do algoritmo foi modificado para considerar o número classes no processo de geração dos classificadores base, onde a relação do erro de treinamento (ε) foi dada pelo número de classes (NC) do atributo alvo, conforme equação 4.2:.

$$\varepsilon = \frac{1}{NC} \tag{4.2}$$

A proposta de implementação do AdaBoost.M1 para esse cenário é dada pelo fluxo do processo de geração das amostras (Seção 2.5.2), onde o erro de treinamento do classificador base em vez de considerar um limiar de erro fixo de 0,5, é dado pela comparação em relação à probabilidade de acerto de cada classe do conjunto. Caso o (ε) seja maior que (NC), ou o número de classificadores gerados (C) seja igual ao número de iterações (k), o processo é encerrado e a classificação dos dados é feita com o número de classificadores gerados até aquele momento.

Para cada conjunto, os resultados das taxas de acerto foram separados e organizados para serem comparados entre o algoritmo AdaBoost.M1-AL, e o algoritmo adaptado ao número de classes denominado AdaBoost.M1-NC. Com esses experimentos a ideia é verificar até que ponto a proposta AdaBoost.M1-NC pode ser mais eficaz em relação ao algoritmo original AdaBoost.M1 original.

4.3.3 AdaBoost.M1 quanto ao limiar erro de treinamento

Esse cenário de experimentos tem como objetivo analisar o comportamento do algoritmo considerando a definição arbitrária de uma série de valores absolutos de limiar de erro de treinamento gerados a cada iteração do AdaBoost, motivados pela questão proposta na seção anterior, definida na equação 4.2. Este trabalho toma foco investigativo para identificar esse fator, considerando o comportamento do limiar erro a fim de identificar algum padrão.

Para verificar o comportamento do algoritmo, considerando o erro de treinamento e os resultados dos conjuntos de dados nos experimentos deste trabalho, o AdaBoost.M1 foi modificado em seu código fonte para execuções com 9 diferentes valores de limiar de erro absoluto respeitando os intervalos de 0,1.

Os experimentos foram executados para cada um dos conjuntos de dados utilizando os mesmos parâmetros de configurações das seções anteriores. O AdaBoost.M1, para cada rodada de execuções, teve seu algoritmo modificado com os limiares de erro com os seguintes valores 0,1, 0,2, 0,3, 0,4, 0,5, 0,6, 0,7, 0,8 e 0,9. Dessa maneira, é possível avaliar a existência de algum padrão relacionado aos valores do limiar de erro, identificando se, em algum caso, os resultados apresentam alguma melhora.

4.3.4 AdaBoost.M1 considerando a eliminação do limiar de erro

Para este cenário a experimentação teve como foco principal a utilização do algoritmo AdaBoost.M1 suprimindo o limiar do erro de treinamento, além de considerar aspectos estocástico da mesma forma que as seções anteriores. A modificação consiste na eliminação da função responsável por interromper o processo interno de amostragem caso o erro de treinamento seja maior que 0,5. Sem esse parâmetro, o número de classificadores é executado por completo utilizando todas as amostras possíveis.

A proposta de adaptação do algoritmo nesta etapa foi denominada AdaBoost.M1-SL, de forma que sua representação é dada pela eliminação do limiar de erro (Seção 2.5.2). No fluxo do processo do algoritmo modificado a cada iteração (k), os classificadores são gerados. Cada classificador base criado é executado sobre o conjunto de dados original, onde seu erro de treinamento (ε) é calculado e os pesos (ω) são atualizados de acordo com este erro. Com as alterações, o critério de parada do algoritmo AdaBoost.M1 passa a ser o número de iterações configurada, sendo executado até o final independente do erro gerado a cada iteração.

O foco principal desta proposta de implementação do algoritmo, está na exploração do número máximo de classificadores de forma a permitir que o AdaBoost.M1-SL obtenha

melhores taxas de acerto quando comparado com a implementação original do algoritmo AdaBoost.M1-AL, levando em consideração aspectos estocásticos no sentido de diminuir os impactos da aleatoriedade nos resultados obtidos.

4.4 Definição dos Testes de Hipóteses

Para verificar se as diferenças encontradas nos resultados com os experimentos empíricos são significativas, não baseando-se apenas em resultados numéricos de superioridade ou inferioridade, os resultados foram submetidos ao *Teste t* e o *p-valor*. Os resultados do *Teste t* fornece o nível de significância do *p-valor*, que fornece evidências para rejeitar a hipótese nula em favor da hipótese alternativa.

Foram definidas hipóteses de comparação entre as adaptações das implementações que apresentam diferenças e o AdaBoost.M1-AL com limiar padrão no valor de 0,5. As comparações foram feitas com os resultados de taxa de acerto e tempo de execução para as implementações AdaBoost.M1-NC e AdaBoost.M1-SL, além dos limiares de erro 0,1, 0,2, 0,3, 0,4 e 0,6 que apresentam diferenças.

Com o *Teste t* executado, para todas as comparações foi definida uma Hipótese nula H_0: $\mu_1 = \mu_2$ e uma Hipótese alternativa H_1, que indica que as diferenças entre as implementações são significativas.

A partir do *Teste t* o *p-valor* é calculado e relacionado ao nível de significância α. No contexto deste estudo o nível de significância adotado foi 0,05, o que indica a possibilidade de 5% de chance de rejeitar a hipótese nula, sendo ela verdadeira.

5. RESULTADOS DA ADAPTAÇÃO DO ALGORITMO ADABOOST.M1

Neste capítulo são apresentados os resultados dos experimentos executados com algoritmo AdaBoost.M1. Para uma definição geral, levou-se em consideração diferentes pontos de vista. Dessa forma, as seções a seguir apresentam a análise dos resultados do AdaBoost.M1 submetido à aleatoriedade (Seção 5.1); a análise dos resultados do AdaBoost.M1 quanto ao número de classes (Seção 5.2); a análise dos resultados levando em consideração o limiar de erro de treinamento absoluto (Seção 5.3); a análise dos resultados, considerando a eliminação do limiar do erro de treinamento (Seção 5.4); e a análise dos resultados submetidos a testes de hipóteses (Seção 5.5).

5.1 Análise dos resultados com AdaBoost.M1 submetido à aleatoriedade

A análise dos resultados utilizando o algoritmo AdaBoost.M1 foi realizada por meio das taxas de acerto e desvio padrão das taxas. Para identificação dos experimentos o algoritmo submetido a aleatoriedade foi denominado AdaBoost.M1-AL, enquanto que o algoritmo original manteve a sua própria denominação AdaBoost.M1. A primeira observação sobre os resultados está na média da taxa de acerto entre os resultados dos conjuntos de dados, em que o AdaBoost.M1 ficou em 81,012% e AdaBoost.M1-AL em 81,496%, de forma que AdaBoost.M1 apresentou um aumento percentual na taxa de acerto de 0,484%. Para a média do desvio padrão do AdaBoost.M1-AL, o resultado foi 0,838, demonstrando que em geral os resultados ficaram mais próximos. Observando os resultados levantados com as taxas de acerto na Tabela 5.1, na maioria dos conjuntos de dados analisados o AdaBoost.M1-AL teve melhores resultados em relação ao AdaBoost.M1. Os resultados foram comparados considerando a quantidade de casos em que o percentual de taxa de acerto de um algoritmo foi maior em relação a outro, de maneira que em 22 dos 32 conjuntos de dados o AdaBoost.M1-AL obteve taxas de acerto maiores em relação ao AdaBoost.M1.

De acordo com Quinlan [Qui96], para análise de resultados de métodos de classificação, um algoritmo pode ser considerado mais eficaz que o outro, quando comparadas suas taxas de acerto e a sua diferença percentual for igual ou superior a 2%. Para os conjuntos de dados em que o AdaBoost.M1 original foi melhor, na maioria a diferença percentual não chegou a 2%, de forma que B30 foi o maior com 1,728%. No caso do AdaBoost.M1-AL, em dois conjuntos de dados a diferença superou os 2%, chegando a 6,296% em B09 e em B11 a 2,576% a mais em relação ao AdaBoost.M1 conforme gráfico da Figura 5.1. Para a maioria dos demais conjuntos a diferença nos resultados não ultrapassaram 1%.

Tabela 5.1 – Resultados do AdaBoost.M1 submetido a aleatoriedade.

Bases	AdaBoost.M1 Taxa (%)	AdaBoost.M1-AL Taxa (%)	AdaBoost.M1-AL Desvio (±)	Diferença
B01	23,103	23,473	0,398	0,370
B02	74,336	74,956	0,778	0,620
B03	84,071	84,961	0,804	0,890
B04	76,64	75,339	0,683	**1,301**
B05	65,734	67,894	1,692	2,160
B06	96,586	96,717	0,221	0,131
B07	86,747	87,662	0,711	0,915
B08	89,231	87,624	1,086	**1,607**
B09	64,198	70,494	2,414	6,296
B10	86,012	85,42	0,847	**0,592**
B11	59,848	62,424	1,73	2,576
B12	97,664	97,58	0,216	**0,084**
B13	80,528	80,033	0,992	**0,495**
B14	78,519	79,942	1,228	1,423
B15	84,516	84,093	1,344	**0,423**
B16	79,274	79,413	0,178	0,139
B17	99,625	99,639	0,053	0,014
B18	89,33	90,626	0,671	1,296
B19	72,396	73,615	0,837	1,219
B20	56,667	58	2,378	1,333
B21	39,233	40,036	1,396	0,803
B22	95,833	96,667	2,722	0,834
B23	81,857	82,718	0,281	0,861
B24	98,643	98,846	0,238	0,203
B25	99,761	99,71	0,029	**0,051**
B26	98,571	98,621	0,086	0,050
B27	97,913	97,852	0,085	**0,061**
B28	96,661	97,022	0,35	0,361
B29	96,567	96,581	0,251	0,014
B30	99,438	97,709	0,617	**1,729**
B31	47,844	46,514	0,704	**1,330**
B32	95,05	95,699	0,783	0,649
Média	81,012	81,496	0,838	0,484

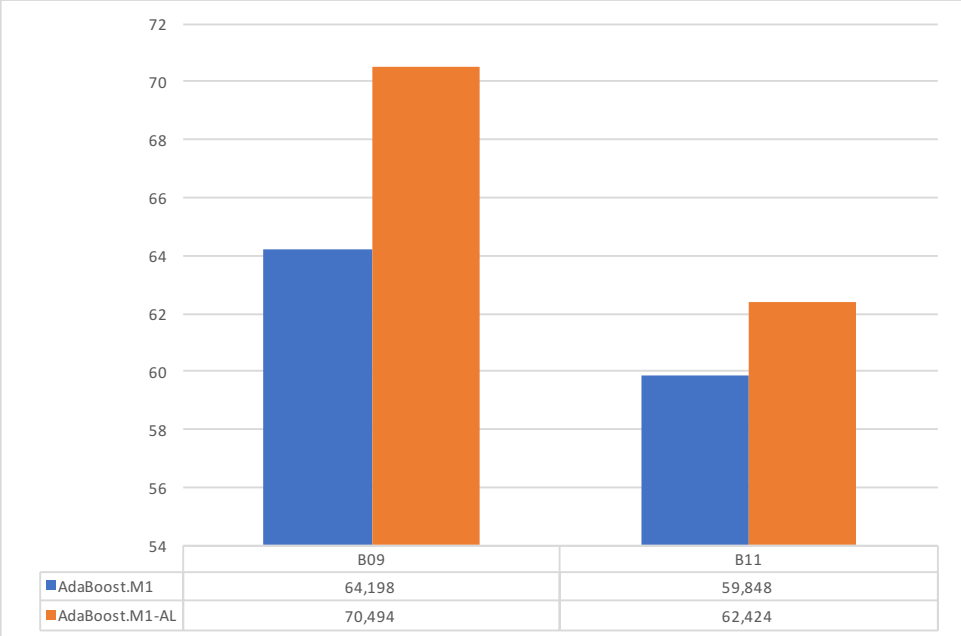


Figura 5.1 – Taxas de acerto AdaBoost.M1 e AdaBoost.M1-AL.

Levando em consideração o desvio padrão obtido com as taxas de acerto, no gráfico da Figura 5.2 os resultados são apresentados. Percebe-se que, para 23 dos 32 conjuntos de dados, as taxas de acerto ficaram mais próximas, devido ao resultado do

desvio padrão que ficou entre 0 e 1, de forma que, para esses exemplos, o algoritmo, quando submetidos à aleatoriedade, é menos sensível aos impactos gerados por ela. Para alguns casos como por exemplo B17, B25, B26 e B27 os resultados foram muito próximos a zero, isso caracteriza que obtiveram o mínimo de dispersão. Para os 6 dos 32 conjuntos, o desvio padrão ficou entre 1 e 2, de forma que, para esses conjuntos, os resultados das taxas de acerto foram um pouco mais dispersos. Para os conjuntos de dados B09, B20 e B22, os valores ficaram acima de 2, representando serem mais vulneráveis à aleatoriedade das sementes.

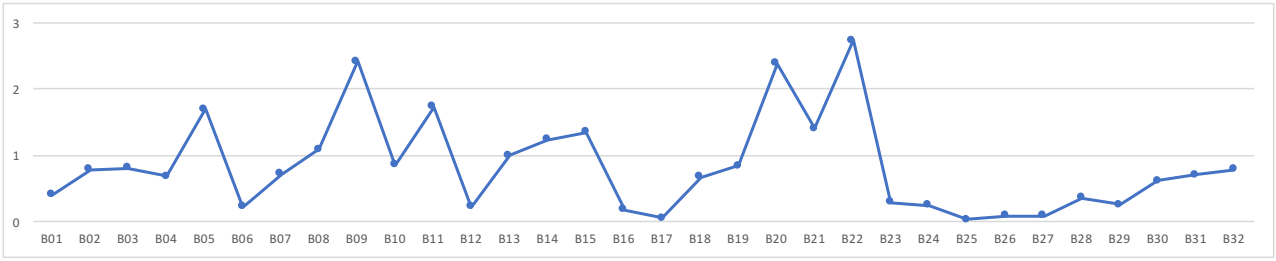


Figura 5.2 – Desvio padrão AdaBoost.M1-AL.

Os resultados com o algoritmo AdaBoost.M1 submetido a aleatoriedade mostram o impacto das sementes aleatórias no processo de geração de classificadores. Para a maioria dos casos a diferença percentual foi maior, superando 2%. Baseado neste cenário de experimentação este estudo toma a direção de adotar o critério de aleatoriedade para os demais cenários experimentados.

5.2 Análise dos resultados com AdaBoost.M1 quanto ao número de classes

A análise dos resultados utilizando o algoritmo AdaBoost.M1 foi realizada por meio das taxas de acerto, desvio padrão e tempo de execução. Conforme os resultados apresentados na seção anterior, todos os experimentos foram submetidos a aleatoriedade, e para sua identificação, o algoritmo, considerando o número de classes, foi denominado AdaBoost.M1-NC.

Na Tabela 5.2 são apresentados os resultados da taxa de acerto do algoritmo AdaBoost.M1 considerando o número de classes dos 32 conjuntos de dados analisados. Nessa etapa de análise, o foco foi dado ao processo de geração dos classificadores de forma que os resultados foram comparados entre o algoritmo AdaBoost.M1-AL, que considera o limiar do erro de treinamento no valor de 0.5, e a implementação adaptada AdaBoost.M1-NC, que considera a relação de probabilidade de seleção do número de classes para limiar de erro.

Tabela 5.2 – Resultados do AdaBoost.M1 quanto ao número de classes.

	AdaBoost.M1-AL			AdaBoost.M1-NC			Diferença		
Bases	**Taxa** (%)	**Desvio** ($\pm$)	**Tempo** (*s*)	**Taxa** (%)	**Desvio** ($\pm$)	**Tempo** (*s*)	**Taxa** (%)	**Desvio** ($\pm$)	**Tempo** (*s*)
B01	23,473	0,398	11747	20,354	0,556	247	**3,119**	0,16	11500
B02	74,956	0,778	9650	64,154	1,334	256	**10,802**	0,56	9394
B03	84,961	0,804	295	75,973	1,391	8	**8,988**	0,59	287
B04	75,339	0,683	310	75,339	0,683	310	-	-	-
B05	67,894	1,692	56	67,894	1,692	54	-	-	2
B06	96,717	0,221	300	96,717	0,221	295	-	-	5
B07	87,662	0,711	208	87,662	0,711	214	-	-	**6**
B08	87,624	1,086	36	87,624	1,086	36	-	-	-
B09	70,494	2,414	39	70,508	2,431	40	0,014	0,02	**1**
B10	85,42	0,847	208	79,421	1,229	8	**5,999**	0,38	200
B11	62,424	1,73	76	62,424	1,749	77	-	0,02	**1**
B12	97,58	0,216	64	97,580	0,216	65	-	-	**1**
B13	80,033	0,992	136	80,033	0,992	134	-	-	2
B14	79,942	1,228	156	79,942	1,228	158	-	-	**2**
B15	84,093	1,344	84	84,093	1,344	84	-	-	-
B16	79,413	0,178	23321	79,413	0,178	22501	-	-	820
B17	99,639	0,053	2260	99,639	0,053	2182	-	-	78
B18	90,626	0,671	263	90,626	0,671	261	-	-	2
B19	73,615	0,837	619	73,615	0,837	620	-	-	**1**
B20	58	2,378	17	60,395	3,105	4	2,395	0,73	13
B21	40,036	1,396	45	36,496	1,428	7	**3,540**	0,03	38
B22	96,667	2,722	6	96,667	2,722	7	-	-	**1**
B23	82,718	0,281	128	83,449	0,282	8	0,731	0,00	120
B24	98,846	0,238	167	98,846	0,238	158	-	-	9
B25	99,71	0,029	2238	99,710	0,029	2197	-	-	41
B26	98,621	0,086	1914	98,621	0,086	1880	-	-	34
B27	97,852	0,085	3315	97,852	0,085	3283	-	-	32
B28	97,022	0,35	478	97,022	0,35	480	-	-	**2**
B29	96,581	0,251	290	96,581	0,251	290	-	-	-
B30	97,709	0,617	54	97,709	0,617	54	-	-	-
B31	46,514	0,704	1305	48,803	0,843	35	2,289	0,14	1270
B32	95,699	0,783	32	95,677	0,801	29	**0,022**	0,02	3
Média	81,496	0,838	1869	80,651	0,92	1124	**0,845**	0,08	745

A primeira observação está na média da taxa de acerto entre os resultados dos conjuntos de dados em que o AdaBoost.M1-AL ficou com 81,496%, e AdaBoost.M1-NC com 80,651%, de forma que AdaBoost.M1-AL foi superior com uma diferença percentual na taxa de acerto de 0,845%. Quando comparada a média do desvio padrão para o AdaBoost.M1-AL, o resultado foi 0,838, enquanto que o algoritmo que considera o número de classes AdaBoost.M1-NC, foi 0,920, demonstrando que em média os resultados com AdaBoost.M1-AL original foram menos dispersos em relação a AdaBoost.M1-NC. Por outro lado, levando em consideração o tempo de execução, o AdaBoost.M1-NC em média foi menor, isso ocorre em função dos conjuntos de dados que possuem uma quantidade maior de classes, em que seus limiares de erro são menores, interrompendo o processo de classificação antes das 50 iterações configuradas nos parâmetros.

Para a maioria dos conjuntos de dados analisados, os resultados foram iguais, isso ocorre devido ao fato de que a maioria possui atributo alvo de 2 e 3 classes, em que o limiar de erro de treinamento é respectivamente 0,5 e 0,33. Ainda assim, no restante dos

resultados, o AdaBoost.M1-AL obteve taxas de acerto maiores em relação ao AdaBoost.M1-NC.

Observando o gráfico da Figura 5.3, para esses conjuntos de dados, o percentual nas taxas de acerto foi superior utilizando o algoritmo AdaBoost.M1-AL. Levando em consideração a margem de 2% de diferença entre os algoritmos, para alguns casos os valores foram superiores de forma considerável. No caso de B01, a diferença entre taxas de acerto chegou a 3,119%, e de B21 a 3,54%. Para B10, B03 e B02 a diferença foi maior sendo de 5,999%, 8,988% e 10,802%. O conjunto B32 não superou os 2%, apresentando uma diferença muito pequena de 0,022%. Observando as características dos conjuntos de dados B03 e B02 nota-se que possuem um número elevado de atributos e classes, e pouca quantidade de instancias, quando comparado aos demais conjuntos, possivelmente apresentando uma correlação entre esses conjuntos e o AdaBoost.M1-NC quanto ao número de classes.

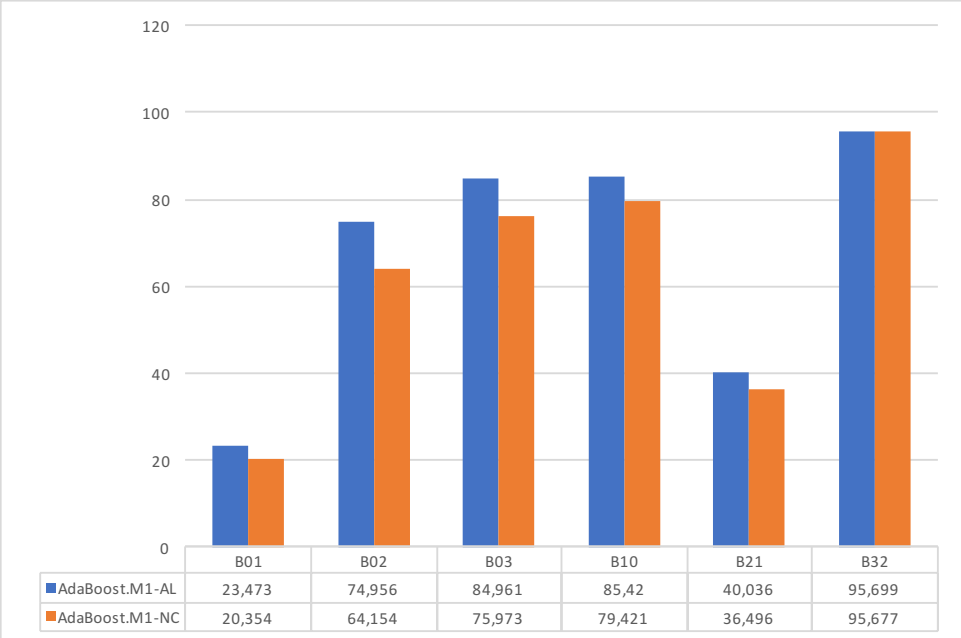


Figura 5.3 – Taxas de acerto favoráveis a AdaBoost.M1-AL.

Observando o gráfico da Figura 5.4, para esses conjuntos de dados o aumento das taxas de acerto foi maior utilizando o algoritmo AdaBoost.M1-NC. Para B09 e B23, o percentual não superou os 2% de diferença, com os resultados respectivos de 0,014% e 0,731 %. Para os casos de B20 e B31, a diferença entre as taxas foram 2,395% e 2,289%, diferença que supera os 2%.

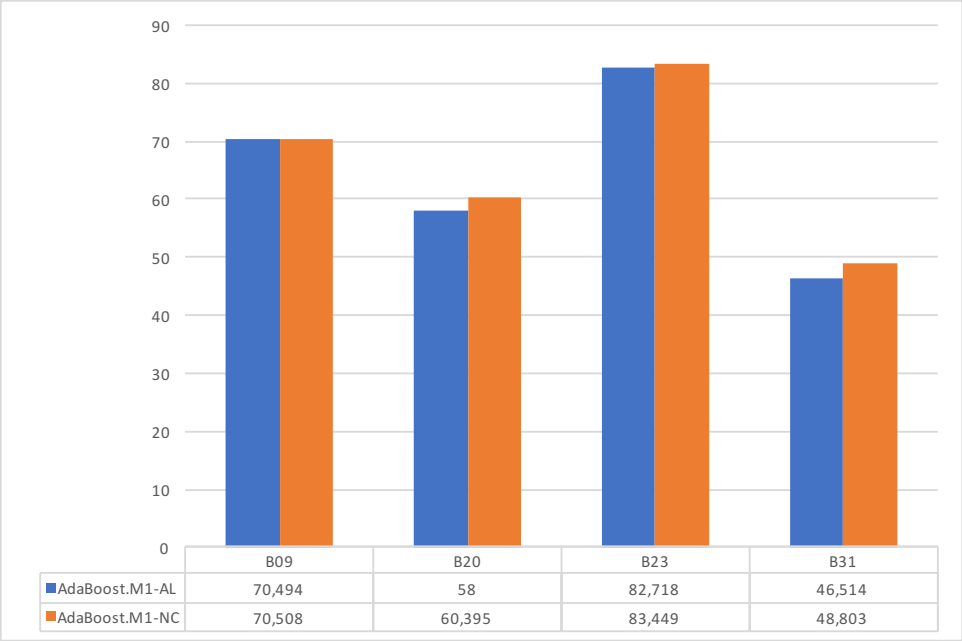


Figura 5.4 – Taxas de acerto favoráveis a AdaBoost.M1-NC.

A partir dos resultados das médias das taxas de acerto, o desvio padrão foi calculado. Para 22 dos conjuntos de dados analisados, o desvio padrão foi igual para ambas as implementações. No gráfico da Figura 5.5 são apresentados os resultados do desvio padrão dos 10 conjuntos de dados que ficaram com diferenças entre taxas de acerto. Observando os resultados percebe-se que o algoritmo AdaBoost.M1-AL demonstrou ser menos vulnerável à aleatoriedade. Analisando os conjuntos de dados B02, B03, B010 e B20, a dispersão dos resultados foi maior com AdaBoost.M1-NC.

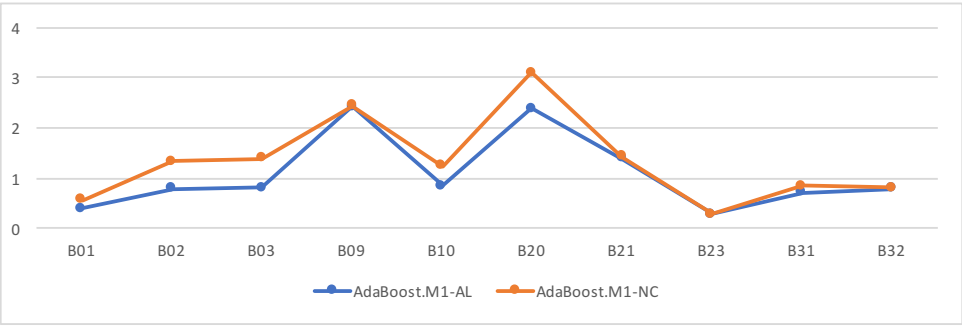
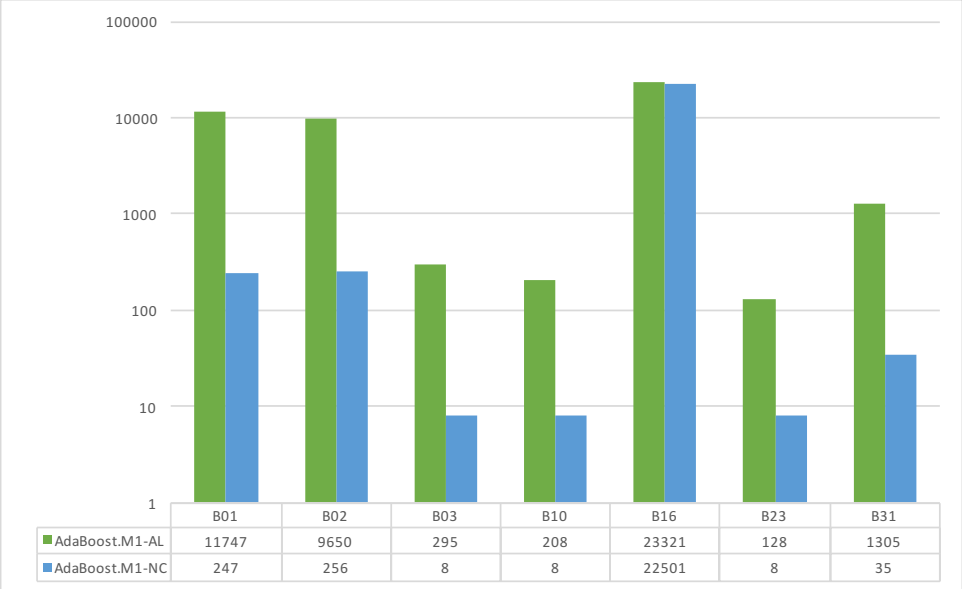


Figura 5.5 – Desvio padrão AdaBoost.M1-AL e AdaBoost.M1-NC.

Observando os resultados da Tabela 5.1, para a maioria dos conjuntos de dados o tempo de execução foi menor, quando utilizado o AdaBoost.M1-NC. Levando em consideração que o processo de iterações depende do número de classes, quando esse número é elevado, o limiar de erro tende a diminuir de forma que uma ocorrência de classificador de baixa capacidade interrompido o processo.

No gráfico da Figura 5.6 são apresentados os conjuntos de dados em que houve ocorrência de diferença considerável nos tempos de execução. Em todos esses casos o

AdaBoost.M1-NC ficou com o tempo de execução menor em relação ao AdaBoost.M1-AL. No caso de B01 a diferença entre o tempo de execução foi de 11500 segundos a menos. Em B02 a diferença foi de 9394 segundos a menos. Para B03, B10 e B23, os tempos para ambas as implementações foram menores, com uma diferença de 287, 200, 120 segundos respectivamente, sendo que o AdaBoost.M1-NC teve seu encerramento para os 3 conjuntos de dados com apenas 8 segundos. No caso de B16, o tempo de execução de ambos foi alto, enquanto que sua diferença ficou em 820 segundos a menos com AdaBoost.M1-NC. Para B31 o tempo de execução do AdaBoost.M1-NC foi de 35 segundos, 1270 segundos a menos que o AdaBoost.M1-AL. Observando as características dos conjuntos de dados, nota-se que a maioria possui um número elevado de classes, com exceção de B16, que por outro lado, possui apenas 2 classes, mas um grande número de instâncias.



	B01	B02	B03	B10	B16	B23	B31
AdaBoost.M1-AL	11747	9650	295	208	23321	128	1305
AdaBoost.M1-NC	247	256	8	8	22501	8	35

Figura 5.6 – Tempos de execução favoráveis a AdaBoost.M1-NC.

5.3 Análise dos resultados com AdaBoost.M1 quanto ao limiar do erro de treinamento absoluto

Com os resultados levantados na implementação do AdaBoost.M1-NC da seção anterior, para esta etapa de experimentação, o limiar do erro de treinamento foi experimentado com valores absolutos modificados no código fonte do AdaBoost.M1-AL, a fim de analisar o comportamento dos conjuntos de dados quanto ao limiar de erro de treinamento absoluto e de procurar algum padrão, analisando as taxas de acerto, desvio padrão e o tempo de execução.

A primeira observação sobre os resultados da Tabela 5.3 é a na média das taxas de acerto entre os resultados dos conjuntos de dados, em que o AdaBoost.M1-AL com seus

limiares modificados apresenta uma variação conforme os valores são experimentados. Com o limiar alterado para 0.1 a média geral apresenta o menor resultado de 79,415%, com limiar 0,2, 0,3, 0,4 e 0,5 a taxa média sobe com variações de forma que 0,5 alcança o resultado maior com 81,496%. Conforme os experimentos foram executados para os demais limiares 0,6, 0,7, 0,8 e 0,9 os resultados não sofreram alterações, ficando com 81,477%.

Tabela 5.3 – Resultados das Taxas de acerto com limiares de erro absoluto.

Limiar	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**
B01	20,354	20,354	20,354	20,355	**23,473**	**23,473**	**23,473**	**23,473**	**23,473**
B02	64,154	73,001	**74,956**	**74,956**	**74,956**	**74,956**	**74,956**	**74,956**	**74,956**
B03	75,988	84,287	**84,961**	**84,961**	**84,961**	**84,961**	**84,961**	**84,961**	**84,961**
B04	78,251	75,376	**75,339**	**75,339**	**75,339**	**75,339**	**75,339**	**75,339**	**75,339**
B05	70,408	70,179	67,11	67,669	67,894	**67,883**	**67,883**	**67,883**	**67,883**
B06	96,703	**96,717**	**96,717**	**96,717**	**96,717**	**96,717**	**96,717**	**96,717**	**96,717**
B07	86,156	87,262	87,432	87,53	87,662	**87,713**	**87,713**	**87,713**	**87,713**
B08	87,47	**87,624**	**87,624**	**87,624**	**87,624**	**87,624**	**87,624**	**87,624**	**87,624**
B09	61,385	62,894	70,329	**70,494**	**70,494**	**70,494**	**70,494**	**70,494**	**70,494**
B10	79,676	81,538	85,235	**85,42**	**85,42**	**85,42**	**85,42**	**85,42**	**85,42**
B11	59,082	58,906	62,357	**62,424**	**62,424**	**62,424**	**62,424**	**62,424**	**62,424**
B12	**97,58**	**97,58**	**97,58**	**97,58**	**97,58**	**97,58**	**97,58**	**97,58**	**97,58**
B13	74,807	80,011	**80,033**	**80,033**	**80,033**	**80,033**	**80,033**	**80,033**	**80,033**
B14	75,745	**79,942**	**79,942**	**79,942**	**79,942**	**79,942**	**79,942**	**79,942**	**79,942**
B15	78,781	83,792	84,057	**84,093**	**84,093**	**84,093**	**84,093**	**84,093**	**84,093**
B16	76,72	78,172	78,647	79,074	79,413	**79,444**	**79,444**	**79,444**	**79,444**
B17	**99,639**	**99,639**	**99,639**	**99,639**	**99,639**	**99,639**	**99,639**	**99,639**	**99,639**
B18	88,996	89,755	90,212	90,419	90,626	**90,664**	**90,664**	**90,664**	**90,664**
B19	70,926	73,585	73,613	**73,615**	**73,615**	**73,615**	**73,615**	**73,615**	**73,615**
B20	63,309	63,309	60,889	58,099	58	**57,778**	**57,778**	**57,778**	**57,778**
B21	36,496	36,496	36,496	36,496	40,036	**39,682**	**39,685**	**39,685**	**39,685**
B22	91,713	96,204	**96,667**	**96,667**	**96,667**	**96,667**	**96,667**	**96,667**	**96,667**
B23	83,449	83,449	83,449	82,957	82,718	**82,568**	**82,568**	**82,568**	**82,568**
B24	91,754	**98,846**	**98,846**	**98,846**	**98,846**	**98,846**	**98,846**	**98,846**	**98,846**
B25	**99,71**	**99,71**	**99,71**	**99,71**	**99,71**	**99,71**	**99,71**	**99,71**	**99,71**
B26	**98,621**	**98,621**	**98,621**	**98,621**	**98,621**	**98,621**	**98,621**	**98,621**	**98,621**
B27	**97,852**	**97,852**	**97,852**	**97,852**	**97,852**	**97,852**	**97,852**	**97,852**	**97,852**
B28	**97,022**	**97,022**	**97,022**	**97,022**	**97,022**	**97,022**	**97,022**	**97,022**	**97,022**
B29	**96,581**	**96,581**	**96,581**	**96,581**	**96,581**	**96,581**	**96,581**	**96,581**	**96,581**
B30	97,678	**97,709**	**97,709**	**97,709**	**97,709**	**97,709**	**97,709**	**97,709**	**97,709**
B31	48,803	48,803	45,754	46,877	**46,514**	**46,51**	**46,51**	**46,51**	**46,51**
B32	95,479	**95,699**	**95,699**	**95,699**	**95,699**	**95,699**	**95,699**	**95,699**	**95,699**
Média	79,415	80,966	81,295	81,282	81,496	81,477	81,477	81,477	81,477

Os resultados das taxas de acerto foram separados por conjuntos de dados conforme seu comportamento e de acordo com as variações do limiar de erro utilizado conforme identificação em negrito na Tabela 5.3. Conforme o valor do limiar de erro é aumentado, a taxa de acerto melhora até que o valor fica estabilizado, não sofrendo alterações. No gráfico da Figura 5.7, são apresentados os conjuntos B06, B08, B14, B24 e B32 que demonstram que os resultados com o limiar de erro de 0,1 são melhores e não se alteram a partir desde valor. Os conjuntos de dados analisados demonstram que o algoritmo a partir do limiar de erro 0,1 apresenta um aumento nas taxas de acerto e os valores estabilizam com o limiar

0,2. Para os conjuntos B06, B08, B30, B32 o aumento a partir do limiar 0,2 é pequeno, não chegando a 1%. Para os conjuntos B14 e B24 a diferença é maior, ficando em torno de 7% e 4%, respectivamente. Uma característica comum entre esse grupo de conjuntos de dados foi que não sofreram alterações em suas taxas de acerto conforme ocorre a variação dos demais limiares, e para a maioria dos conjuntos da taxa de acerto supera os 95%.

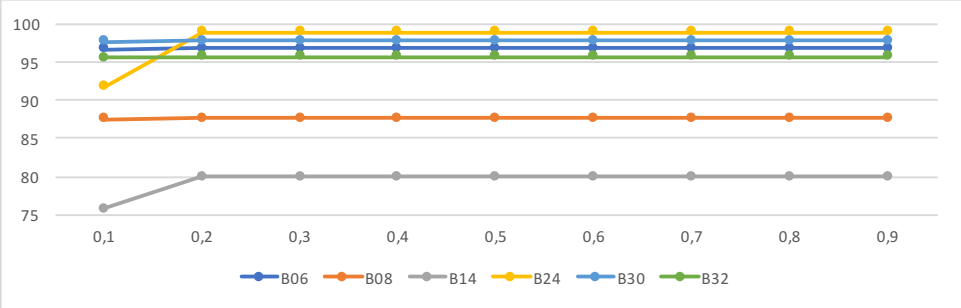


Figura 5.7 – Aumento de taxas de acerto com limiar de erro absoluto em 0,2.

No gráfico da Figura 5.8 são apresentados os conjuntos B02, B03, B13 e B22 que demonstram que os resultados com o limiar de erro de 0,3 são melhores e não se alteram a partir deste valor. Os conjuntos de dados analisados demonstram que o algoritmo com limiar de erro de 0,3 apresenta um aumento nas taxas de acerto e os valores estabilizam. A diferença percentual entre o limiar 0,1 e 0,3 para B13 e B22 fica em torno de 5%, enquanto que para B02 e B03 a diferença chega a 10%.

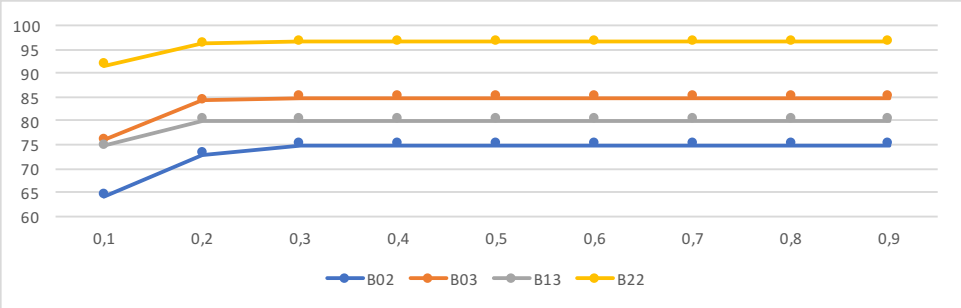


Figura 5.8 – Aumento de taxas de acerto com limiar de erro absoluto em 0,3.

Para os resultados no gráfico da Figura 5.9 são apresentados os conjuntos B09, B10, B11, B15 e B19 que demonstram que os resultados com o limiar de erro de 0,3 são melhores e não se alteram a partir desde valor. Os conjuntos de dados analisados demonstram que o algoritmo com limiar de erro de 0,3 apresenta um aumento nas taxas de acerto, e os valores estabilizam da mesma forma que o grupo anterior, mas com uma diferença menor. A diferença percentual entre o limiar 0,1 e 0,4 para B09, B11 e B19 fica em torno de 3%, enquanto que para B10 e B15 a diferença chega a 5%.

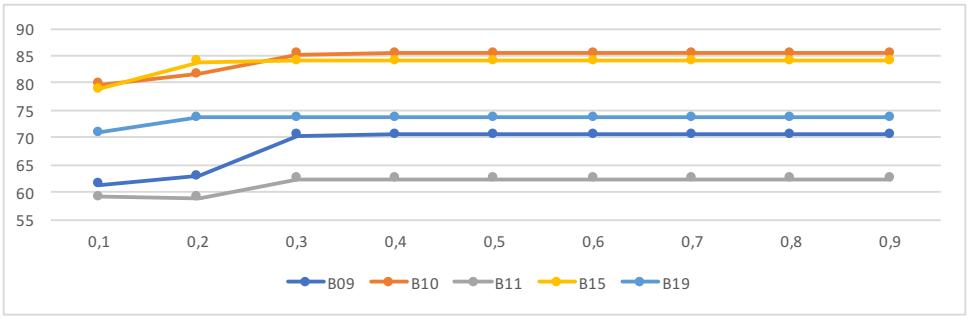


Figura 5.9 – Aumento de taxas de acerto com limiar de erro absoluto em 0,3.

No gráfico da Figura 5.10 são apresentados os conjuntos B01 e B21 que demonstram que os resultados com o limiar de erro de 0,5 são melhores e não se alteram a partir deste valor. Os conjuntos de dados analisados demonstram que o algoritmo com limiar de erro de 0,5 apresenta um aumento nas taxas de acerto e os valores estabilizam. A diferença percentual entre o limiar 0,1 e 0,5 para os conjuntos fica em torno de 3%, de forma que para esses casos o aumento do valor ocorre entre os limiares 0,4 e 0,5.

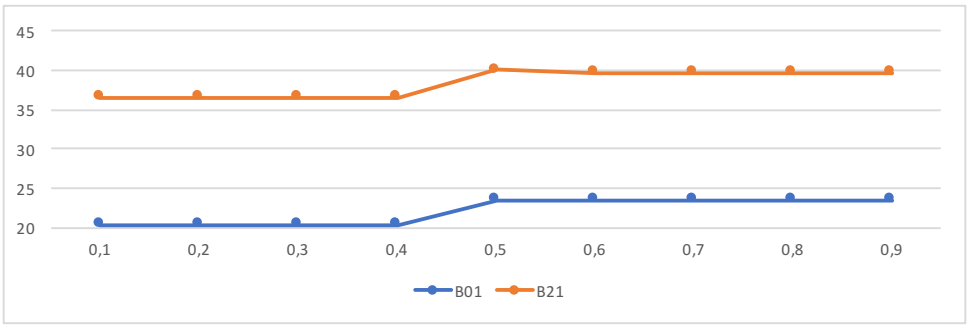


Figura 5.10 – Aumento de taxas de acerto com limiar de erro absoluto em 0,5.

Para os resultados no gráfico da Figura 5.11 são apresentados os conjuntos B07, B16 e B18 que demonstram que os resultados com o limiar de erro de 0,6 são melhores e não se alteram a partir desde valor. Os conjuntos de dados analisados demonstram que o algoritmo com limiar de erro de 0,6 apresenta um aumento nas taxas de acerto e os valores estabilizam. A diferença percentual entre o limiar 0,1 e 0,6 para B07 e B18 fica próximo de 2%, para o caso de B16 foi superior, aproximando-se dos 3% .

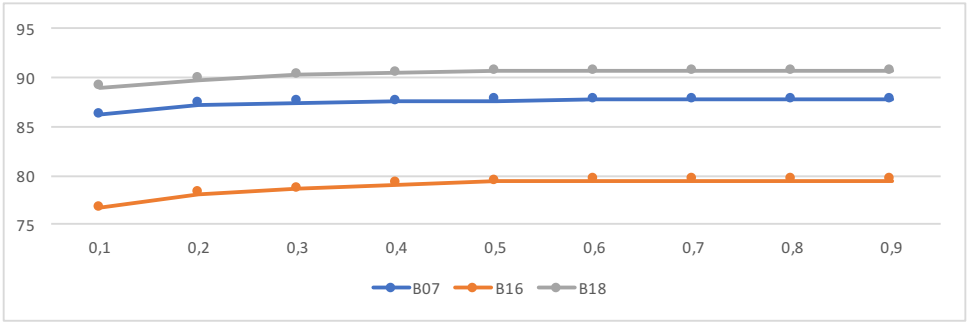


Figura 5.11 – Aumento de taxas de acerto com limiar de erro absoluto em 0,6.

Os resultados apresentados no gráfico da Figura 5.12 demonstram que para os conjuntos B12, B17, B25, B26, B27, B28 e B29 a variação dos limiares de erro absolutos não influenciam no resultado das taxas de acerto, de maneira que ficam iguais. Outra característica dos resultados desses conjuntos está nas taxas de acerto que para todos os casos ficam acima de 96%.

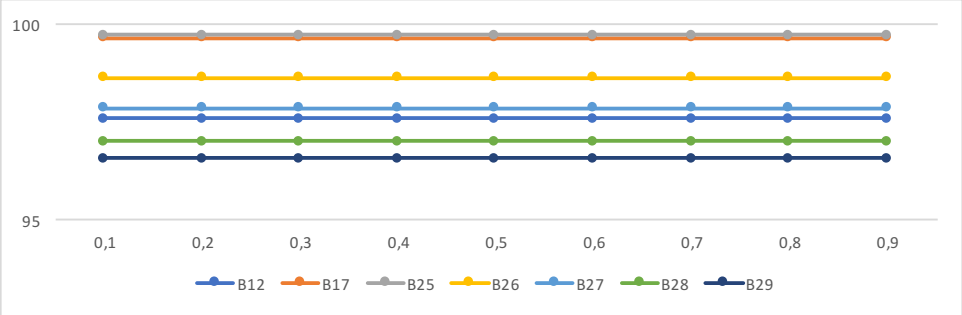


Figura 5.12 – Taxas de acerto iguais com variação de limiares de erro.

Conforme o apresentado no gráfico da Figura 5.13 o conjunto de dados B04 apresentou resultados diferentes dos anteriores, pois a taxa de acerto tende a diminuir quando o limiar de erro utilizado é de 0,2. Para B04, o percentual caiu de 78,251% para 75,376%, representando uma diminuição que fica próxima a 3%.

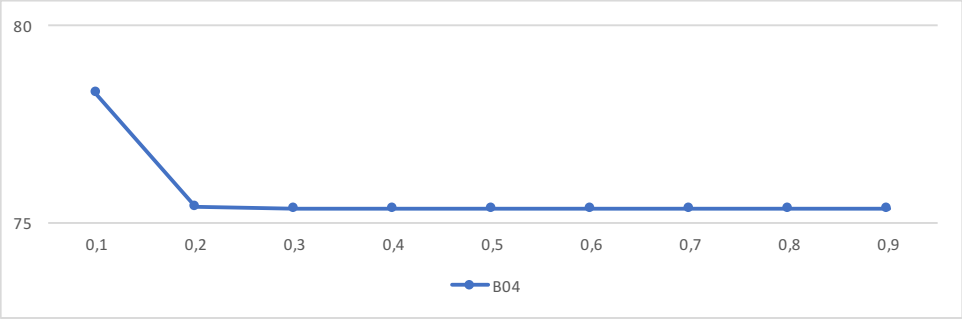


Figura 5.13 – Diminuição de taxas de acerto com limiar de erro absoluto em 0,2.

Para os resultados dos conjuntos de dados B05, B20 e B31 apresentados no gráfico da Figura 5.14, a taxa de acerto tende a diminuir quando o limiar de erro utilizado é de 0,3, com algumas variações até o limiar 0,6, até que seus resultados estabilizam. Para B05 e B31, o percentual caiu em torno de 2%. No caso de B20 a diminuição é maior, em torno de 5%.

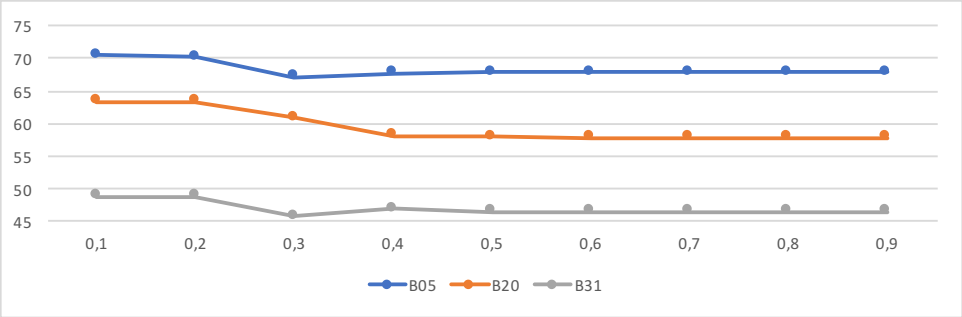


Figura 5.14 – Diminuição de taxas de acerto com limiar de erro absoluto em 0,3.

Conforme apresentado no gráfico da Figura 5.15, com o conjunto de dados B23 a taxa de acerto tende a diminuir quando o limiar de erro utilizado é de 0,4. Para B23, o percentual caiu de 83,449% para 82,568%, representando uma diminuição pequena, inferior a 1%.

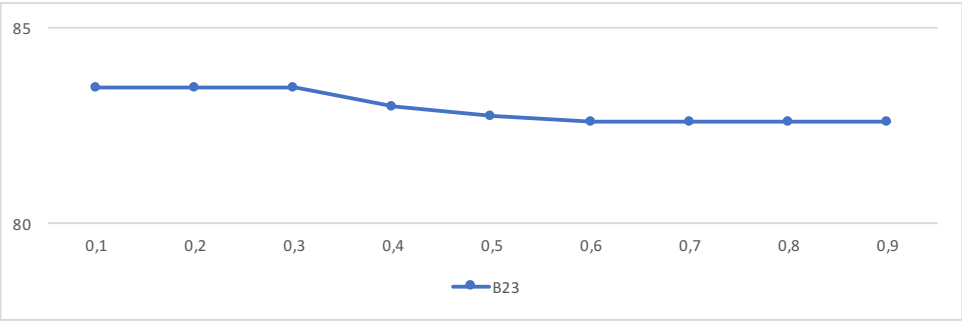


Figura 5.15 – Diminuição de taxas de acerto com limiar de erro absoluto em 0,4.

Os conjuntos foram agrupados na Tabela 5.4 e separados pelos melhores resultados de taxa de acerto de acordo com os limiares de erro. Para 9 conjuntos dos 32 analisados, a partir da utilização do limiar de erro absoluto 0,3, as taxas de acerto apresentam um comportamento melhor. Para os limiar 0,2 foram 6 conjuntos de dados com melhores resultados. Para o limiar de erro utilizado com valor 0,5 os conjuntos B01 e B21 apresentaram melhoras. Para 0,6 foram os conjuntos B07, B16 e B18. Para os 7 demais conjuntos a variação do limiar de erro não demostrou alterações nos resultados de taxa de acerto.

Tabela 5.4 – Conjuntos de dados com melhores limiares de erro para taxas de acerto

Limiar de erro ε	**Conjuntos de dados**
0,2	B06, B08, B14, B24, B30, B32
0,3	B02, B03, B09, B10, B11, B13, B15, B19, B22
0,5	B01, B21
0,6	B07, B16, B18
–	B12, B17, B25, B26, B27, B28, B29

Observando os resultados da Tabela 5.5 a média do desvio padrão das taxas de acerto é apresentada e indicadas em negrito. Com o limiar de erro alterado para 0,2 a média geral do desvio ficou em 1,188, valor mais alto entre os limiares experimentados, indicando

que com esse limiar os resultados são mais dispersos. Com os limiares 0,2, 0,3, 0,4, 0,5 e 0,6 os desvios diminuem com variações de forma que em 0,6 alcança o menor resultado de 0,828. Conforme os experimentos são executados para os demais limiares 0,7, 0,8 e 0,9 os resultados não sofreram alterações.

Tabela 5.5 – Resultados dos Desvio padrão com limiares de erro absoluto.

Limiar	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**
B01	0,556	0,556	0,556	0,556	**0,398**	**0,398**	**0,398**	**0,398**	**0,398**
B02	1,334	1,386	**0,778**	**0,778**	**0,778**	**0,778**	**0,778**	**0,778**	**0,778**
B03	1,363	1,135	**0,804**	**0,804**	**0,804**	**0,804**	**0,804**	**0,804**	**0,804**
B04	0,942	0,673	**0,683**	**0,683**	**0,683**	**0,683**	**0,683**	**0,683**	**0,683**
B05	1,383	1,53	1,673	1,492	1,692	**1,627**	**1,627**	**1,627**	**1,627**
B06	0,225	**0,221**	**0,221**	**0,221**	**0,221**	**0,221**	**0,221**	**0,221**	**0,221**
B07	0,818	0,688	0,71	0,729	0,711	**0,651**	**0,651**	**0,651**	**0,651**
B08	1,199	**1,086**	**1,086**	**1,086**	**1,086**	**1,086**	**1,086**	**1,086**	**1,086**
B09	3,107	3,12	2,606	**2,414**	**2,414**	**2,414**	**2,414**	**2,414**	**2,414**
B10	1,077	1,26	0,88	**0,847**	**0,847**	**0,847**	**0,847**	**0,847**	**0,847**
B11	3,067	3,003	1,715	**1,73**	**1,73**	**1,73**	**1,73**	**1,73**	**1,73**
B12	**0,216**	**0,216**	**0,216**	**0,216**	**0,216**	**0,216**	**0,216**	**0,216**	**0,216**
B13	1,82	0,995	**0,992**	**0,992**	**0,992**	**0,992**	**0,992**	**0,992**	**0,992**
B14	1,903	**1,228**	**1,228**	**1,228**	**1,228**	**1,228**	**1,228**	**1,228**	**1,228**
B15	1,888	1,293	**1,347**	**1,344**	**1,344**	**1,344**	**1,344**	**1,344**	**1,344**
B16	0,273	0,246	0,2	0,191	0,178	**0,186**	**0,186**	**0,186**	**0,186**
B17	**0,053**	**0,053**	**0,053**	**0,053**	**0,053**	**0,053**	**0,053**	**0,053**	**0,053**
B18	0,934	0,739	0,72	0,709	0,671	**0,684**	**0,684**	**0,684**	**0,684**
B19	1,132	0,851	0,836	**0,837**	**0,837**	**0,837**	**0,837**	**0,837**	**0,837**
B20	3,022	3,022	3,427	2,903	2,378	**2,342**	**2,342**	**2,342**	**2,342**
B21	1,428	1,428	1,428	1,428	1,396	1,204	**1,202**	**1,202**	**1,202**
B22	4,585	3,083	**2,722**	**2,722**	**2,722**	**2,722**	**2,722**	**2,722**	**2,722**
B23	0,282	0,282	0,282	0,356	0,281	**0,317**	**0,317**	**0,317**	**0,317**
B24	2,198	**0,238**	**0,238**	**0,238**	**0,238**	**0,238**	**0,238**	**0,238**	**0,238**
B25	**0,029**	**0,029**	**0,029**	**0,029**	**0,029**	**0,029**	**0,029**	**0,029**	**0,029**
B26	**0,086**	**0,086**	**0,086**	**0,086**	**0,086**	**0,086**	**0,086**	**0,086**	**0,086**
B27	**0,085**	**0,085**	**0,085**	**0,085**	**0,085**	**0,085**	**0,085**	**0,085**	**0,085**
B28	**0,35**	**0,35**	**0,35**	**0,35**	**0,35**	**0,35**	**0,35**	**0,35**	**0,35**
B29	**0,251**	**0,251**	**0,251**	**0,251**	**0,251**	**0,251**	**0,251**	**0,251**	**0,251**
B30	0,644	**0,617**	**0,617**	**0,617**	**0,617**	**0,617**	**0,617**	**0,617**	**0,617**
B31	0,843	0,843	1,166	0,865	0,704	**0,7**	**0,7**	**0,7**	**0,7**
B32	0,922	**0,783**	**0,783**	**0,783**	**0,783**	**0,783**	**0,783**	**0,783**	**0,783**
Média	1,188	0,981	0,899	0,863	0,838	**0,828**	**0,828**	**0,828**	**0,828**

Os resultados do desvio padrão foram separados em grupos por resultados e por conjuntos de dados conforme a variação do limiar de erro absoluto utilizado e apresentado em gráficos. O gráfico da Figura 5.16 apresenta os resultados do primeiro grupo de conjuntos de dados, com o desvio padrão. Nota-se que para os conjuntos B06, B08, B14, B24, B30 e B32 com o limiares de erro maiores que 0,1 os resultados foram mais próximos de zero, indicando que são menos sensíveis aos impactos da aleatoriedade, e seus resultados são mais agrupados. Para B24, os resultados com o limiar 0,1 ficou em 2,198, indicando que esse conjunto teve resultados mais dispersos.

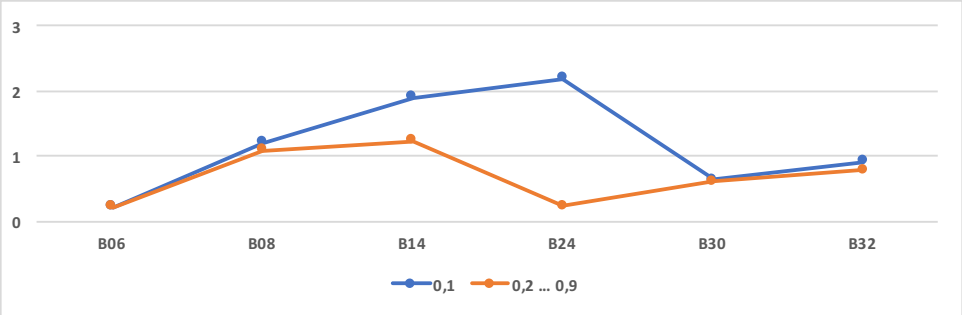


Figura 5.16 – Desvio padrão menores com limiar 0,2.

Observando o gráfico da Figura 5.17, os resultados desse grupo de conjuntos de dados com o desvio padrão, nota-se que para os conjuntos B02, B03, B13 e B22, com o limiares de erro maiores que 0,2 os resultados foram mais próximos de zero, indicando que são menos sensíveis aos impactos da aleatoriedade e seus resultados são mais agrupados. Para B22, os resultados dos desvios padrão foram superiores, com o limiar 0,1 foi de 3,083 e com 0,2 foi de 4,585, indicando que esse conjunto de dados teve resultados mais dispersos em relação aos demais limiares.

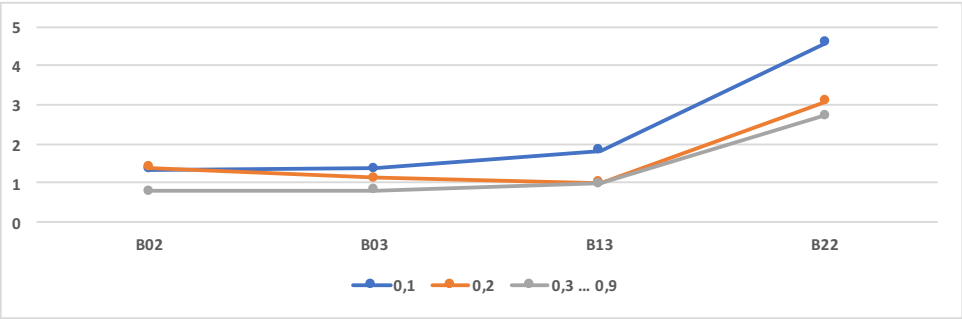


Figura 5.17 – Desvio padrão menores com limiar 0,3.

Observando o gráfico da Figura 5.18, os resultados desse grupo de conjuntos de dados com o desvio padrão, nota-se que para os conjuntos B09, B10, B11, B15 e B19 com os limiares de erro maiores que 0,3 os resultados foram mais próximos de zero, indicando que são menos sensíveis aos impactos da aleatoriedade e seus resultados são mais agrupados. Para esse grupo de conjuntos de dados a diferença entre os desvios padrão foi pequena.

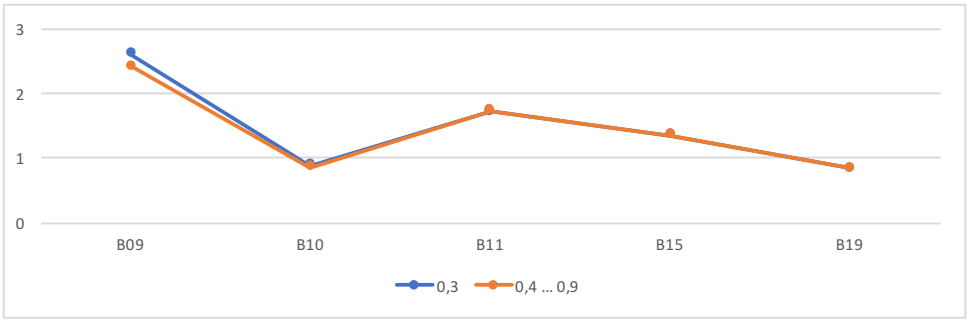


Figura 5.18 – Desvio padrão menores com limiar 0,4.

Para os resultados do gráfico da Figura 5.19, nota-se que para os conjuntos B01 e B21 com o limiares de erro superior a 0,4, os resultados foram mais próximos de zero, indicando que são menos sensíveis aos impactos da aleatoriedade e seus resultados foram mais agrupados. Para esse grupo de conjuntos de dados a diferença entre os desvios padrão também foi pequena. O conjunto B21 apresentou desvios mais distantes de zero, de forma que seus resultados foram mais dispersos.

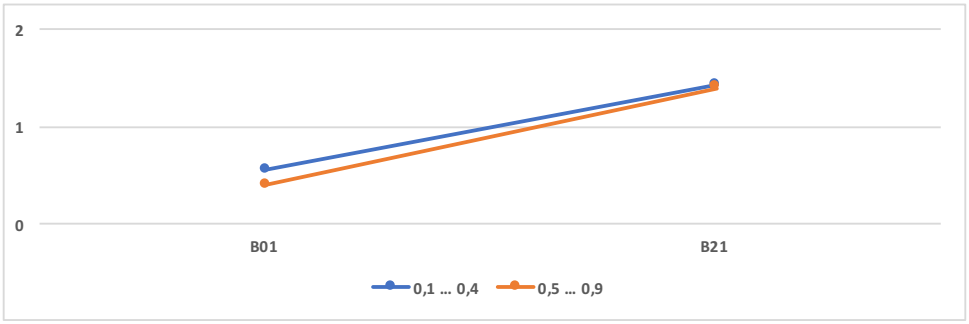


Figura 5.19 – Desvio padrão menores com limiar 0,5.

Para os resultados do gráfico da Figura 5.20, nota-se que para os conjuntos B04, B05, B07 e B20, com o limiares de erro superior a 0,5, os resultados foram mais próximos de zero, indicando que são menos sensíveis aos impactos da aleatoriedade e seus resultados foram mais agrupados. Para esse grupo de conjuntos de dados a diferença entre os desvios padrão foi pequena. O conjunto B04 ficou com desvios padrão iguais com limiares superiores a 0,6.

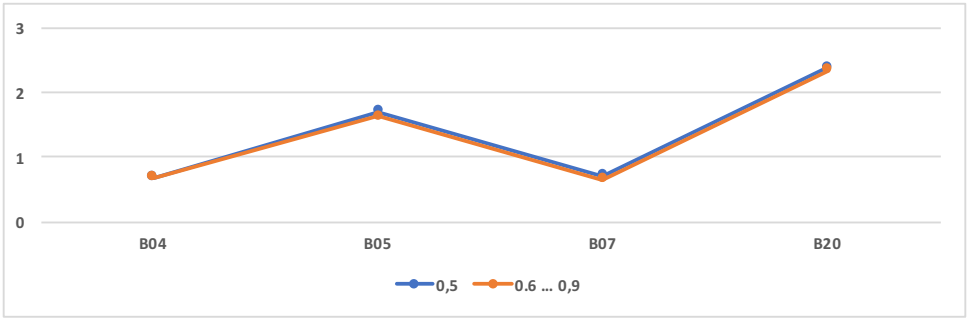


Figura 5.20 – Desvio padrão menores com limiar 0,6.

Observando o gráfico da Figura 5.21, os resultados desse grupo de conjuntos de dados B12, B17, B25, B26, B27, B28 e B29 não sofreram alterações nos desvios padrão com a variação dos limiares entre 0,1 e 0,9. Para todos os conjuntos os valores ficaram próximos de zero, sendo B28 o maior, com 0,350, indicando que seus resultados foram mais agrupados.

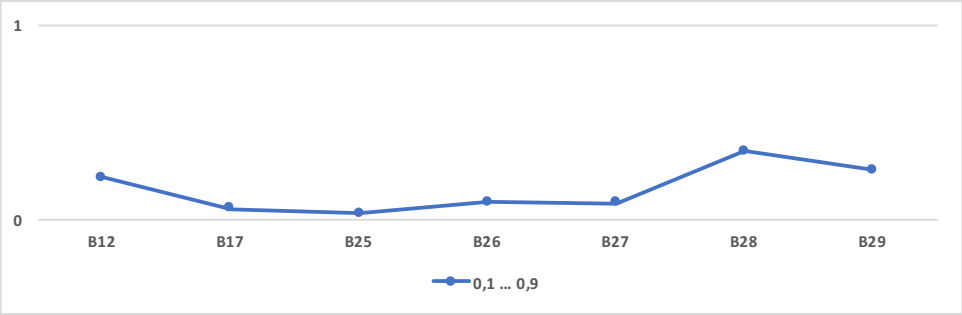


Figura 5.21 – Desvio padrão iguais.

Observando o gráfico da Figura 5.22, nota-se que para os conjuntos B16, B18, B023 e B31 com o limiares de erro superior a 0,6 os resultados foram mais distantes de zero, ao contrário dos resultados apresentados anteriormente. Nesse grupo com limiares até 0,5 os conjuntos foram menos sensíveis aos impactos da aleatoriedade, e seus resultados foram mais agrupados.

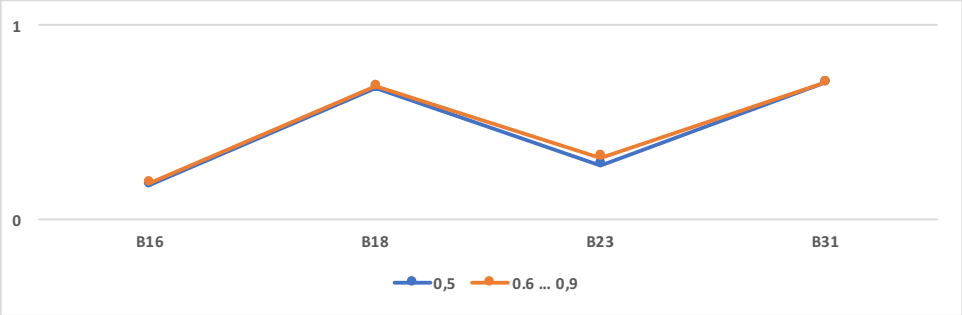


Figura 5.22 – Desvio padrão maiores com limiar 0,6.

Observando os resultados da Tabela 5.6, a média do tempo de execução é apresentada. Conforme o limiar de erro absoluto é alterado de 0,1 até 0,9 o tempo médio geral aumenta. Isso ocorre pois quanto menor for o limiar de erro maior a probabilidade de interrupção do processo de geração dos classificadores.

Os resultados do tempo de execução foram normalizados entre 0 e 100%, e organizados e separados em grupos por resultados indicados em negrito e por conjuntos de dados conforme a variação do limiar de erro absoluto utilizado e apresentados em gráficos.

Tabela 5.6 – Resultados de Tempo de execução com limiares de erro absoluto.

Limiar	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
B01	452	438	435	445	11747	**12406**	**12422**	**12421**	**12466**
B02	282	7492	9971	10089	**9650**	**9789**	**9829**	**9795**	**9959**
B03	14	258	**324**	**323**	**295**	**300**	**299**	**299**	**300**
B04	15	**318**	**319**	**317**	**310**	**310**	**311**	**310**	**311**
B05	8	8	19	30	**56**	**59**	**59**	**59**	**59**
B06	332	335	334	332	**300**	**303**	**303**	**303**	**315**
B07	143	169	181	190	208	**276**	**277**	**277**	**290**
B08	18	**37**	**37**	**37**	**36**	**36**	**36**	**36**	**39**
B09	4	8	**40**	**41**	**39**	**40**	**40**	**41**	**43**
B10	24	60	**199**	**210**	**208**	**208**	**208**	**208**	**220**
B11	5	8	**77**	**78**	**76**	**77**	**77**	**77**	**81**
B12	**66**	**66**	**65**	**65**	**64**	**65**	**65**	**66**	**71**
B13	8	**141**	**141**	**140**	**136**	**138**	**138**	**138**	**145**
B14	8	**160**	**161**	**159**	**156**	**158**	**159**	**158**	**165**
B15	6	59	**85**	**87**	**84**	**85**	**85**	**85**	**89**
B16	957	9611	12392	15395	23321	25003	25118	25227	25223
B17	2619	2621	2630	2607	**2260**	**2261**	**2268**	**2261**	**2264**
B18	59	76	96	122	263	**331**	**330**	**330**	**330**
B19	18	**620**	**645**	**635**	**619**	**620**	**620**	**621**	**621**
B20	3	4	4	8	17	**24**	**25**	**24**	**24**
B21	11	11	12	11	45	**309**	**308**	**309**	**310**
B22	**3**	**5**	**7**	**7**	**6**	**7**	**6**	**7**	**6**
B23	19	28	33	53	128	**476**	**476**	**476**	**476**
B24	97	**178**	**179**	**178**	**167**	**167**	**166**	**166**	**166**
B25	2385	2391	2391	2375	**2238**	**2241**	**2241**	**2241**	**2236**
B26	2039	2041	2039	2026	**1914**	**1917**	**1915**	**1919**	**1919**
B27	3536	3538	3543	3520	**3315**	**3308**	**3322**	**3316**	**3316**
B28	**484**	**485**	**488**	**479**	**478**	**479**	**479**	**480**	**479**
B29	**294**	**294**	**294**	**292**	**290**	**290**	**290**	**290**	**290**
B30	**53**	**55**	**54**	**54**	**54**	**54**	**54**	**54**	**54**
B31	36	36	375	978	**1305**	**1309**	**1308**	**1315**	**1312**
B32	20	**34**	**34**	**33**	**32**	**32**	**32**	**32**	**32**
Média	438	987	1175	1291	**1869**	**1971**	**1977**	**1979**	**1988**

Nos gráficos das Figura 5.23 e Figura 5.24 são apresentados os conjuntos B02, B03, B04, B08, B09, B10, B11, B13, B14, B15, B19, B22, B24 e B32 que demonstram que o tempo de execução com o limiar de erro de 0,2 é menor em relação aos demais limiares. Com o limiar alterado para 0,3 o tempo tende a subir com poucas variações, ficando com valores próximos a 100%.

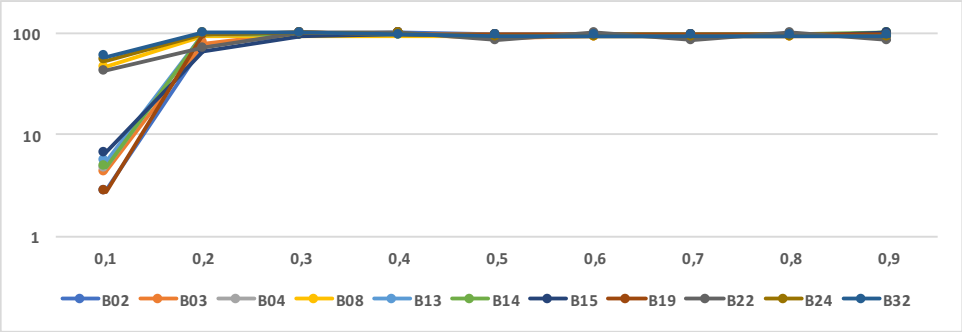


Figura 5.23 – Aumento dos tempos de execução com limiar de erro absoluto em 0,3.

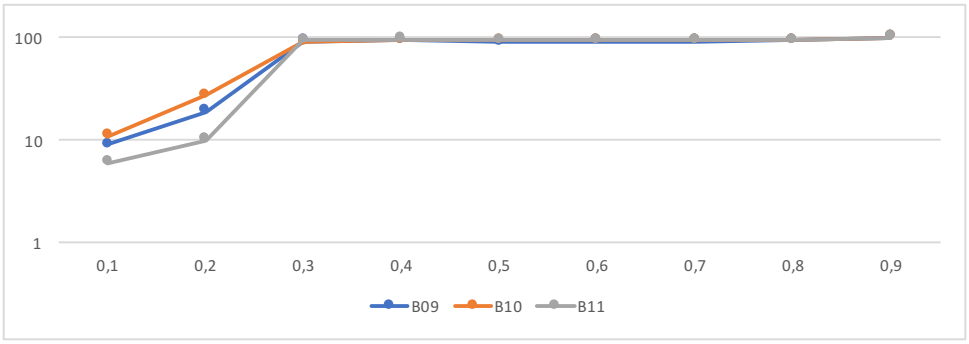


Figura 5.24 – Aumento dos tempos de execução com limiar de erro absoluto em 0,3.

Para os resultados do gráfico da Figura 5.25 são apresentados os tempos de execução dos conjuntos B01, B05, B16 e B31, que demonstram que com o limiar alterado para 0,5 ou mais, o tempo tende a subir com poucas variações ficando com valores próximos a 100%.

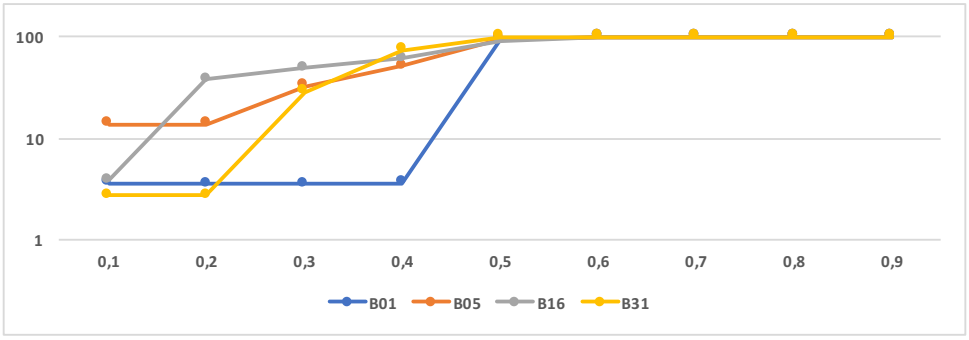


Figura 5.25 – Aumento dos tempos de execução com limiar de erro absoluto em 0,5.

Para os resultados do gráfico da Figura 5.26 são apresentados os tempos de execução dos conjuntos B07, B18, B20, B21 e B23 que demonstram que com o limiar alterado para 0,6 ou mais, o tempo tende a subir com poucas variações, ficando com valores próximos a 100%.

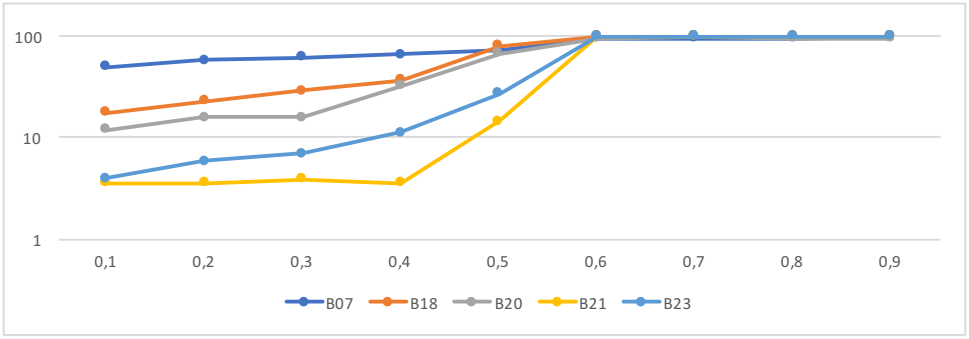


Figura 5.26 – Aumento dos tempos de execução com limiar de erro absoluto em 0,6.

Para os resultados do gráfico da Figura 5.27 são apresentados os tempos de execução dos conjuntos B06, B12, B25, B26, B27, B28, B29 e B30 que demonstram que com

o limiar de erro entre 0,1 e 0,9 não apresentam variações consideráveis, de forma que os valores ficam todos próximos a 100%.

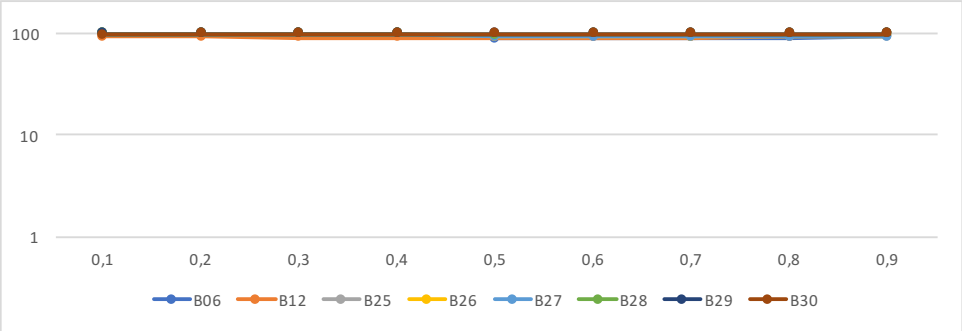


Figura 5.27 – Tempos de execução iguais.

Para os resultados do gráfico da Figura 5.28 o tempo de execução do conjunto de dados B17 demonstra que com o limiar de erro em 0,5 o tempo tende a cair.

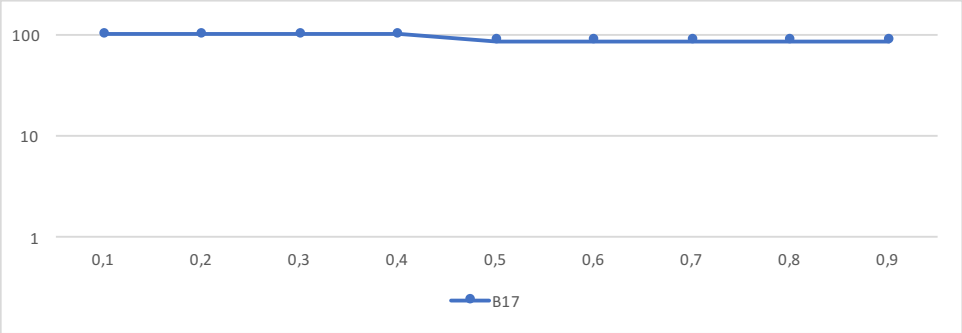


Figura 5.28 – Diminuição do tempo de execução com limiar de erro absoluto em 0,5.

Os conjuntos foram agrupados na Tabela 5.7 e separados pelos menores resultados de tempo de execução de acordo com os limiares de erro. Para 14 conjuntos dos 32 analisados, a partir da utilização do limiar de erro absoluto 0,3 os tempos de execução apresentam um comportamento melhor. Para o limiar de erro utilizado com valor 0,5 os conjuntos B01, B05, B16, e B31 apresentaram os tempos de execução menores. Para 0,6 os tempos de execução menores foram para os conjuntos B07, B18, B20, B21 e B23. Para os 9 demais conjuntos o limiar de erro não demostrou alterações nos resultados de tempo de execução, de maneira que os resultados foram próximos ao 100%.

Tabela 5.7 – Conjuntos de dados com melhores limiares de erro para tempo de execução.

Limiar de erro ε	**Conjuntos de dados**
0,3	B02, B03, B04, B08, B09, B10 e B11, B13, B14, B15, B19, B22, B24 e B32
0,5	B01, B05, B16 e B31
0,6	B07, B18, B20, B21 e B23
–	B06, B12, B17, B25, B26, B27, B28, B29 e B30

5.4 Análise dos resultados com AdaBoost.M1-AL quanto a eliminação do limiar do erro de treinamento

A análise dos resultados, considerando a eliminação do limiar de erro, foi realizada por meio das taxas de acerto, desvio padrão e tempo de execução. Com o objetivo de identificar o comportamento do algoritmo, o limiar do erro de treinamento foi suprimido, permitindo a utilização do máximo de iterações (classificadores) possíveis. Para identificação dos experimentos, o algoritmo sem o limiar de erro foi denominado AdaBoost.M1-SL, enquanto que o algoritmo original com sua própria denominação AdaBoost.M1-AL.

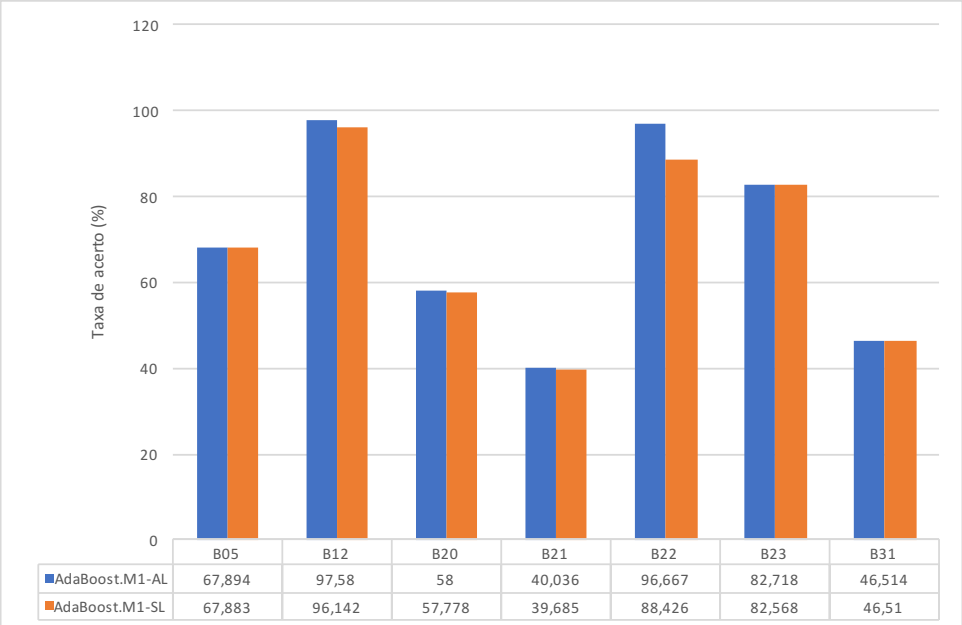
Tabela 5.8 – Resultados do AdaBoost.M1-AL com a eliminação do limiar de erro.

	AdaBoost.M1-AL			**AdaBoost.M1-SL**			**Diferença**		
Bases	**Taxa** (%)	**Desvio** ($\pm$)	**Tempo** (*s*)	**Taxa** (%)	**Desvio** ($\pm$)	**Tempo** (*s*)	**Taxa** (%)	**Desvio** ($\pm$)	**Tempo** (*s*)
B01	23,473	0,398	11747	23,473	0,398	12409	-	-	**662**
B02	74,956	0,778	9650	74,956	0,778	9791	-	-	**141**
B03	84,961	0,804	295	84,961	0,804	300	-	-	**5**
B04	75,339	0,683	310	75,339	0,683	311	-	-	**1**
B05	67,894	1,692	56	67,883	1,627	59	**0,011**	**0,06**	**3**
B06	96,717	0,221	300	96,717	0,221	304	-	-	**4**
B07	87,662	0,711	208	87,713	0,651	276	0,051	**0,06**	**68**
B08	87,624	1,086	36	87,624	1,086	36	-	-	0
B09	70,494	2,414	39	70,494	2,414	40	-	-	**1**
B10	85,42	0,847	208	85,420	0,847	208	-	-	0
B11	62,424	1,73	76	62,424	1,73	77	-	-	**1**
B12	97,58	0,216	64	96,142	0,705	7	**1,438**	0,49	57
B13	80,033	0,992	136	80,033	0,992	141	-	-	**5**
B14	79,942	1,228	156	79,942	1,228	162	-	-	**6**
B15	84,093	1,344	84	84,093	1,344	85	-	-	**1**
B16	79,413	0,178	23321	79,444	0,186	25724	0,031	0,01	**2403**
B17	99,639	0,053	2260	99,639	0,053	2340	-	-	**80**
B18	90,626	0,671	263	90,664	0,684	332	0,038	0,01	**69**
B19	73,615	0,837	619	73,615	0,837	643	-	-	**24**
B20	58	2,378	17	57,778	2,342	24	**0,222**	**0,04**	**7**
B21	40,036	1,396	45	39,685	1,202	317	**0,351**	0,19	**272**
B22	96,667	2,722	6	88,426	5,582	6	**8,241**	2,86	0
B23	82,718	0,281	128	82,568	0,317	446	**0,150**	0,04	**318**
B24	98,846	0,238	167	98,846	0,238	155	-	-	12
B25	99,71	0,029	2238	99,710	0,029	2216	-	-	22
B26	98,621	0,086	1914	98,621	0,086	1892	-	-	22
B27	97,852	0,085	3315	97,852	0,085	3344	-	-	**29**
B28	97,022	0,35	478	97,022	0,35	458	-	-	20
B29	96,581	0,251	290	96,581	0,251	286	-	-	4
B30	97,709	0,617	54	97,709	0,617	52	-	-	2
B31	46,514	0,704	1305	46,510	0,7	1294	**0,004**	-	11
B32	95,699	0,783	32	95,699	0,783	31	-	-	1
Média	81,496	0,838	1869	81,174	0,933	1993	**0,322**	0,10	**124**

A primeira observação sobre os resultados da Tabela 5.8 está na média da taxa de acerto entre os resultados dos conjuntos de dados, em que o AdaBoost.M1-AL ficou em 81,496%, e AdaBoost.M1-SL em 81,174%, de forma que AdaBoost,M1 foi superior com

uma diferença percentual na taxa de acerto de 0,322%. Quando comparada a média do desvio padrão para o AdaBoost.M1-AL, o resultado foi 0,838, enquanto o algoritmo que considera o número de classes AdaBoost.M1-SL, foi 0,933, demonstrando que em média os resultados com AdaBoost.M1-AL foram menos dispersos em relação a AdaBoost.M1-SL. O tempo de execução em média com o AdaBoost.M1-AL foi de 1869 segundos, sendo 123 segundos a menos em relação a AdaBoost.M1-SL, que ficou em 1993 segundos. Para maioria dos conjuntos de dados analisados, os resultados foram iguais.

Observando o gráfico da Figura 5.29, para esses conjuntos de dados o percentual nas taxas de acerto foi superior utilizando o algoritmo AdaBoost.M1-AL. Levando em consideração a margem de 2% de diferença entre os algoritmos para alguns casos, os valores foram superiores de forma consideráveis. Apenas B22 obteve um valor considerável de 8,241% de diferença. Para os demais casos, a maioria ficou com diferença menor de 1%.



	B05	B12	B20	B21	B22	B23	B31
AdaBoost.M1-AL	67,894	97,58	58	40,036	96,667	82,718	46,514
AdaBoost.M1-SL	67,883	96,142	57,778	39,685	88,426	82,568	46,51

Figura 5.29 – Taxas de acerto favorável a AdaBoost.M1-AL.

Observando o gráfico da Figura 5.30, para esses conjuntos de dados, o percentual nas taxas de acerto foram superiores utilizando o algoritmo AdaBoost.M1-SL, mas em nenhum caso a diferença foi superior a 1%.

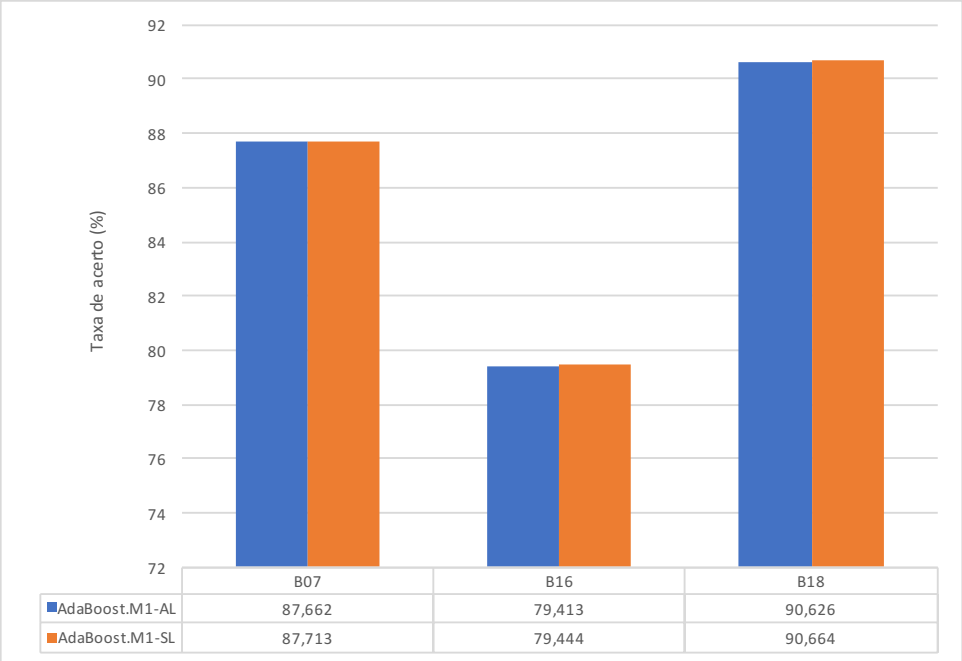


Figura 5.30 – Taxas de acerto favorável a AdaBoost.M1-SL.

A partir dos resultados das médias das taxas de acerto o desvio padrão foi calculado. Para os conjuntos de dados analisados, o desvio padrão foi igual para ambos os algoritmos na maioria dos casos. No gráfico da Figura 5.31, são apresentados os resultados do desvio padrão dos 10 conjuntos de dados que ficaram com diferenças entre taxas de acerto. Observando os resultados percebe-se que o algoritmo AdaBoost.M1-AL demonstrou ser menos vulnerável à aleatoriedade. Analisando o conjunto de dados B22, a dispersão dos resultados foi maior com AdaBoost.M1-SL, com um desvio padrão de 5,5582, o mesmo conjunto em que a diferença percentual foi maior em favor do AdaBoost.M1-AL para taxa de acerto.

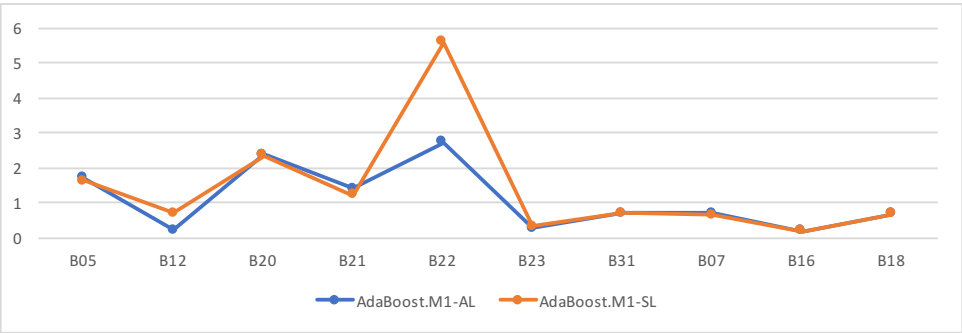


Figura 5.31 – Desvio padrão AdaBoost.M1-AL e AdaBoost.M1-SL.

No gráfico da Figura 5.32 são apresentados os conjuntos de dados em que houve ocorrência de diferença considerável nos tempos de execução. Na maioria dos casos o AdaBoost.M1-AL ficou com o tempo de execução menor em relação ao AdaBoost.M1-SL.

No caso de B01 a diferença entre o tempo de execução foi de 662 segundos a menos. Em B02 a diferença foi de 141 segundos a menos. Para B07, B21 e B23 os tempos para ambas as implementações foram menores, com uma diferença de 68, 272 e 318 segundos, respectivamente. No caso de B16 o tempo de execução de ambos foi alto, enquanto que sua diferença ficou em 2403 segundos a menos com AdaBoost.M1-AL. Para B12 o tempo de execução AdaBoost.M1-SL foi 7 segundos, 57 segundos a menos que o AdaBoost.M1-AL. O algoritmo AdaBoost.M1-SL somente encerra a execução no final de todas as iterações tendo uma disposição a demorar mais em seu processo em alguns casos.

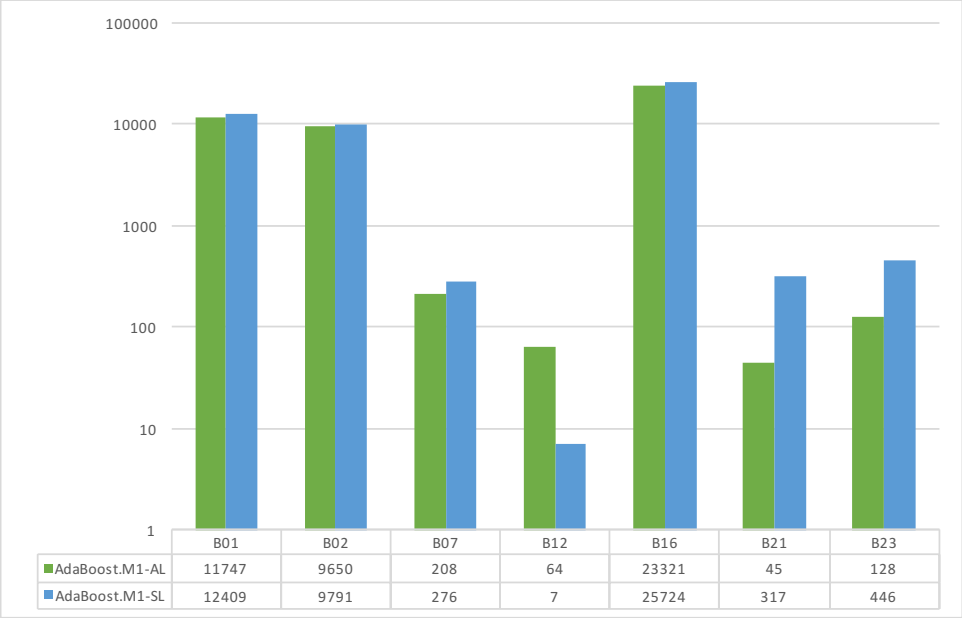


Figura 5.32 – Tempo de execução de AdaBoost.M1-AL e AdaBoost.M1-SL.

5.5 Análise dos resultados submetidos à Teste de Hipóteses

De acordo com os resultados empíricos apresentados nas seções anteriores as hipóteses foram levantadas com o objetivo de identificar estatisticamente se existe diferença significativa entre as implementações e os diferentes limiares de erro testados. Os resultados de taxa de acerto e tempo de execução foram submetidos a testes de hipóteses. O *Teste t* e o *p-valor* foram calculados para fornecer evidências para rejeitar ou não uma hipótese nula em favor da hipótese alternativa.

A Tabela 5.9 apresenta os resultados do *Teste t* para a comparação das taxas de acerto de AdaBoost.M1-AL com as diferentes propostas de implementações de adaptação do algoritmo. O valor numérico de *t* foi calculado e relacionado ao *t* crítico para o teste uni-caudal para verificar se estava dentro da zona de rejeição da hipótese nula. Para as comparações apenas as implementações AdaBoost.M1-NC e AdaBoost.M1 com Limiar 0.1 rejeitaram H_0, de forma que ao um nível de significância de 5% há evidências que o

AdaBoost.M1-AL é superior em relação a estas implementações. Com o *p-valor* uni-caudal calculado considerando um nível de significância de 5%, logo $p < \alpha = 0{,}05$, há indicativa que a diferença entre essas implementações é significativa.

Tabela 5.9 – *Teste t* para taxas de acerto.

Implementações		Médias		t estimado	P(T<=t) uni-caudal	t crítico uni-caudal
AdaBoost.M1-AL	AdaBoost.M1-NC	81,496	80,651	1,703	0,049	1,696
	AdaBoost.M1-SL		81,174	1,239	0,112	
	AdaBoost.M1 limiar 0.1		79,415	3,182	0,002	
	AdaBoost.M1 limiar 0.2		80,966	1,388	0,087	
	AdaBoost.M1 limiar 0.3		81,295	1,123	0,135	
	AdaBoost.M1 limiar 0.4		81,282	1,466	0,076	
	AdaBoost.M1 limiar 0.6		81,477	1,402	0,085	

A Tabela 5.10 apresenta os resultados do *Teste t* para a comparação dos Tempos de execução de AdaBoost.M1-AL com as diferentes propostas de implementações de adaptação do algoritmo. O valor numérico de *t* foi calculado e relacionado ao *t* crítico para o teste uni-caudal para verificar se estava dentro da zona de rejeição da hipótese nula. Para as comparações apenas as implementações AdaBoost.M1 com Limiar 0.1 e AdaBoost.M1 com Limiar 0.2 rejeitaram H_0, de forma que ao um nível de significância de 5% há evidências que o AdaBoost.M1-AL é superior em relação a estas implementações. Com o *p-valor* uni-caudal calculado considerando um nível de significância de 5%, logo $p < \alpha = 0{,}05$, há indicativa que a diferença entre essas implementações é significativa.

Tabela 5.10 – *Teste t* para tempos de execução.

Implementações		Médias		t estimado	P(T<=t) uni-caudal	t crítico uni-caudal
AdaBoost.M1-AL	AdaBoost.M1-NC	1869	1124	1,638	0,056	1,696
	AdaBoost.M1-SL		1993	-1,594	0,060	
	AdaBoost.M1 limiar 0.1		438	1,765	0,044	
	AdaBoost.M1 limiar 0.2		987	1,611	0,059	
	AdaBoost.M1 limiar 0.3		1175	1,432	0,081	
	AdaBoost.M1 limiar 0.4		1291	1,356	0,092	
	AdaBoost.M1 limiar 0.6		1971	-1,808	0,040	

6. CONCLUSÃO

Este trabalho apresentou uma proposta de adaptação para o algoritmo AdaBoost.M1 com o propósito de melhorar a taxa de acerto e o tempo de execução considerando aspectos estocásticos. Cabe lembrar que o presente trabalho não teve ambição de propor um novo método, e sim, testar possibilidades de melhoria no algoritmo original AdaBoost.M1. Para alcançar o objetivo principal foram analisados diversos cenários em cima da implementação do algoritmo AdaBoost.M1, de forma que sua eficácia e seu desempenho foram avaliados no decorrer da pesquisa. Buscando uma perspectiva de análise fez-se uma comparação dos resultados empíricos, levando em consideração a margem de 2% para as taxas de acerto.

Inicialmente esta pesquisa seguiu por uma linha exploratória, com o objetivo de identificar os impactos das sementes aleatórias com o algoritmo AdaBoost.M1, a exemplo dos trabalhos de Fernandes et al. [FLR10] e Fernandes et al. [FLNR13]. No que diz respeito a aspectos estocásticos, com relação aos resultados encontrados, o AdaBoost.M1-AL (Submetido a aleatoriedade) é melhor de uma maneira geral na maioria dos conjuntos de dados em comparação ao AdaBoost.M1 original, demonstrando que o algoritmo é sensível a aleatoriedade das sementes.

Para o desenvolvimento do trabalho o objetivo foi identificar uma possível adaptação para o algoritmo. Desta forma implementou-se o algoritmo AdaBoost.M1-NC, considerando o limiar de acordo com número de classes, e o AdaBoost.M1-SL, considerando a eliminação do limiar de erro. Além disso, para uma visão mais ampla do comportamento do algoritmo, o limiar foi implementado com valores de erro absoluto entre 0,1 e 0,9. Para avaliar o comportamento do AdaBoost.M1 foram realizadas diversas comparações sobre os resultados dos experimentos.

Com relação aos resultados encontrados do primeiro cenário de experimentação, o AdaBoost.M1 é melhor de uma maneira geral nos resultados de taxa de acerto, em comparação ao AdaBoost.M1-NC. As observações sobre os estudos mostram que o AdaBoost.M1 obteve notória melhoria em relação ao AdaBoost.M1-NC quando analisados os valores de desvio padrão, demonstrando que os resultados sofreram menos com os impactos da aleatoriedade, ficando mais próximos de zero. Comparando os resultados de tempo de execução, identificou-se que o AdaBoost.M1-NC foi mais rápido nos casos onde houve diferenças em relação a AdaBoost.M1. Na comparação, o AdaBoost.M1 foi melhor quando considerada a margem de 2% ao comparar a taxa de acerto, enquanto que o AdaBoost.M1-NC foi superior em alguns casos, mas os resultados não ultrapassaram 1%.

Com os resultados encontrados no segundo cenário, a taxa de acerto do AdaBoost.M1 com limiares definidos arbitrariamente entre 0,1 e 0,9, foi melhor na maioria dos conjuntos de dados a partir do limiar 0,3, não sofrendo alterações em seus resultados com

outros limiares maiores. Para uma quantidade significativa de conjuntos de dados, o limiar entre 0,1 e 0,9 não teve interferência nos resultados de taxa de acerto. Para o tempo de execução, o limiar 0,3 também foi que obteve maior representatividade na execução do algoritmos, sendo mais rápido. Da mesma forma que para as taxas de acerto para uma quantidade significativa de conjuntos de dados o limiar entre 0,1 e 0,9 não teve interferência nos resultados do tempo de execução.

Com os resultados encontrados do terceiro cenário, o AdaBoost.M1 é melhor de uma maneira geral nos resultados de taxa de acerto em comparação ao AdaBoost.M1-SL. As observações sobre os estudos mostram que o AdaBoost.M1 obteve melhoria em relação ao AdaBoost.M1-SL, quando analisados os valores de desvio padrão, demonstrando que os resultados sofreram menos com os impactos da aleatoriedade, ficando mais próximos de zero. Comparando os resultados de tempo de execução, identificou-se que o AdaBoost.M1 foi mais rápido nos casos onde houve diferenças em relação a AdaBoost.M1-SL.

Para a experimentação adotada foram observados quatro cenários: o comportamento dos conjuntos dos dados quanto à aleatoriedade; a relação do limiar de erro de treinamento com o número de classes, a exemplo do trabalho de Zhu et al. [ZZRH09]; a utilização arbitrária de valores absolutos do limiar de erro; o comportamento do algoritmo sem utilizar o limiar de erro. Destes, os resultados empíricos que apresentaram uma melhora consistente foi a quando implementado o limiar de erro absoluto com o valor de 0,3.

Para que a análise dos resultados não fosse baseada somente pelo exame de superioridade e inferioridade entre as taxas de acerto e tempos de execução, os resultados em que foram encontradas diferenças foram submetidos a teste de hipóteses, a fim de determinar se essas diferenças entre as implementações são significativas. Para as comparações de taxas de acerto para maioria das hipóteses estudadas, a hipótese nula (H_0) não foi rejeitada, logo não há evidências que as diferenças sejam significativas. Apenas para a comparação entre AdaBoost.M1 e AdaBoost.M1-NC, e para a comparação entre AdaBoost.M1 e AdaBoost.M1 com limiar 0,1, que a hipótese nula foi rejeitada em favor da hipótese alternativa (H_1). Logo, para esses casos o AdaBoost.M1 é superior, e a diferença entre as implementações é significativa. Para as comparações de tempo de execução para maioria das hipóteses estudadas a hipótese nula (H_0) não foi rejeitada. Logo, não há evidências que as diferenças sejam significativas. Apenas para a comparação entre AdaBoost.M1 e AdaBoost.M1 com limiar 0,1 e para a comparação entre AdaBoost.M1, e AdaBoost.M1 com limiar 0,6, que a hipótese nula foi rejeitada em favor da hipótese alternativa (H_1). Logo para esses casos o AdaBoost.M1 é mais rápido e a diferença entre as implementações é significativa.

Todos os resultados apresentados até aqui permitem concluir que, dentro dos experimentos deste presente trabalho, o AdaBoost.M1 com o limiar 0,3 apresentou ser menos vulnerável à aleatoriedade quando comparado ao AdaBoost.M1 original e que, em alguns casos, obteve melhores resultados quanto a taxa de acerto e tempo de execução. Quando

os resultados foram submetidos a testes de hipóteses, o limiar 0,3 é uma boa escolha para adaptação do algoritmo. Por outro, com relação a um nível de significância estatística de 5% outros resultados em comparação com o AdaBoost.M1 original não apresentam evidências de uma diferença significativa, demonstrando que o limiar de erro em 0,5 trata-se de um mito. É possível que o limiar de erro em 0,5 venha da primeira definição do algoritmo para problemas de duas classes ou do conhecido como efeito conjunto onde o resultado deve ser melhor que escolhas aleatórias, apresentado por Hansen e Salamon [HS90].

Este trabalho deixou de aprendizado um bom entendimento sobre Mineração de Dados, técnicas de Aprendizado de Máquina e Combinação de Classificadores. Além de permitir uma compreensão mais profunda sobre o AdaBoost.M1 e o entendimento de que, além de obter melhoras em seus resultados, outras abordagens podem ser estudadas.

6.1 Contribuição do trabalho

A contribuição principal do trabalho foi mostrar, através das implementações propostas e experimentadas com o algoritmo AdaBoost.M1, que é possível adaptá-lo para ser mais eficaz e menos vulnerável à aleatoriedade. O objetivo geral e os objetivos específicos serviram para responder a questão de pesquisa do presente trabalho que, somados ao estudo inicial sobre o AdaBoost.M1, produziram quatro contribuições que claramente podem ser apontadas:

- Experimentos que mostram a variabilidade dos resultados do AdaBoost.M1 quando submetido à aleatoriedade;
- Experimentos que mostram os resultados com a adaptação do algoritmo AdaBoost.M1-NC;
- Experimentos que mostram os resultados com a adaptação do algoritmo AdaBoost.M1-SL;
- Experimentos que mostram os resultados com a adaptação do algoritmo com limiares absolutos entre 0,1 e 0,9;

A primeira das contribuições serviu como apoio e justificativa para o desenvolvimento desta dissertação, cujos experimentos e estudos dos resultados foram analisados por diferentes aspectos. Considerando o número de sementes utilizadas, os resultados do AdaBoost.M1 quanto à aleatoriedade foram melhores em relação ao AdaBoost.M1 com semente padrão 1. Nesse cenário, foi aplicada uma comparação em que foram contabilizados o número de vezes que um algoritmo foi melhor considerando a taxa de acerto e desvio padrão.

A implementação do AdaBoost.NC foi a segunda contribuição. O desenvolvimento do algoritmo não implicou em alterar nenhuma funcionalidade interna do AdaBoost.M1, e sim, utilizar as funcionalidades já existentes para o seu desenvolvimento. Nesse caso, foi apresentada uma forma de geração de classificadores, considerando como limiar do erro de treinamento o inverso do número de classes, conforme sugerido por Zhu et al. [ZZRH09], de forma que não demonstrou ser uma estratégia interessante, uma vez que estatisticamente o AdaBoost.M1 original demonstrou evidências de superioridade em relação a implementação proposta.

A terceira contribuição foi a implementação do AdaBoost.M1 com valores de limiares de erro absolutos distribuídos arbitrariamente entre 0,1 e 0,9. Nesse caso, foi apresentada uma variação de limiares de forma a observar seu comportamento e essa estratégia demonstrou-se interessante por permitir identificar a existência de possibilidades de outros limiares com resultados melhores.

A implementação do AdaBoost.SL foi a quarta contribuição. Nesse caso, foi apresentada uma forma de geração de classificadores, eliminando o limiar do erro de treinamento, para permitir que o algoritmo utilizasse o número máximo de classificadores para o voto ponderado da classe. O funcionamento do algoritmo foi comentado passo a passo para um melhor entendimento. Os resultados estatísticos não demonstraram evidências de diferenças significativas entre as implementações.

6.2 Trabalhos Futuros

Dentro do contexto deste trabalho, o AdaBoost.M1 e os experimentos executados mostraram os ganhos obtidos, pôde-se identificar que diversas questões se encontram em aberto e que podem gerar trabalhos futuros a esta dissertação. Dentre as diversas possibilidades, sugere-se as seguintes atividades:

- Experimentação de outras abordagens, utilizando outros algoritmos de aprendizado de base em vez do J48, como por exemplo, o Bayes rule utilizado no estudo de Zhu et al. [ZZRH09], ou até mesmo, o algoritmo base padrão Decision Stump;
- Análise do comportamento da AdaBoost.M1 frente conjuntos de dados com características específicas e correlacionadas.

O desenvolvimento desses itens poderão contribuir para novas possibilidades de melhorias para o algoritmo AdaBoost.M1.

REFERÊNCIAS BIBLIOGRÁFICAS

[ABC79] A. B. Clarke, R. L. D. “Probabilidade e Processos e Estocásticos”. LTC, 1979, 338p.

[AN14] Asuncion, A.; Newman, D. “Uci machine learning repository”. Capturado em: https://ergodicity.net/2013/07/, Maio 2014.

[Bag05] Bagui, S. C. “Combining pattern classifiers: Methods and algorithms”, *Technometrics*, vol. 47–4, 2005, pp. 517–518.

[BH07] Bühlmann, P.; Hothorn, T. “Boosting algorithms: Regularization, prediction and model fitting”, *Statistical Science*, vol. 22–4, 2007, pp. 477–505.

[BMO14] Boetticher, G.; Menzies, T.; Ostrand, T. “The promise repository of empirical software engineering data”. Capturado em: http://promisedata.org/, Junho 2014.

[Bot01] Botta, M. “Resampling vs reweighting in boosting a relational weak learner”. In: AI*IA 2001: Advances in Artificial Intelligence, 7th Congress of the Italian Association for Artificial Intelligence, 2001, pp. 70–80.

[Bre96] Breiman, L. “Bagging predictors”, *Machine Learning*, vol. 24–2, 1996, pp. 123–140.

[Bre01] Breiman, L. “Random forests”, *Machine Learning*, vol. 45–1, 2001, pp. 5–32.

[Dem06] Demšar, J. “Statistical comparisons of classifiers over multiple data sets”, *Journal of Machine learning research*, vol. 7–1, 2006, pp. 1–30.

[Dev10] Devore, J. L. “Probabilidade e Estatística para Engenharia e Ciências”. Cengage Learning Edições, 2010, 712p.

[DKG00] Diamantidis, N.; Karlis, D.; Giakoumakis, E. A. “Unsupervised stratification of cross-validation for accuracy estimation”, *Artificial Intelligence*, vol. 116–1, 2000, pp. 1–16.

[FHT+00] Friedman, J.; Hastie, T.; Tibshirani, R.; et al.. “Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)”, *The annals of statistics*, vol. 28–2, 2000, pp. 337–407.

[FLGC11] Faceli, K.; Lorena, A. C.; Gama, J.; Carvalho. “Inteligência Artificial: Uma abordagem de aprendizado de máquina”. LTC, 2011, 375p.

[FLNR13] Fernandes, P.; Lopes, L.; Normey, S.; Ruiz, D. D. A. “Stochastic aware random forests-a variation less impacted by randomness.” In: The Florida Artificial Intelligence Research Society, 2013, pp. 425–426.

[FLR10] Fernandes, P.; Lopes, L.; Ruiz, D. D. A. “The impact of random samples in ensemble classifiers”. In: ACM Symposium on Applied Computing, 2010, pp. 1002–1009.

[Fri01] Friedman, J. H. “Greedy function approximation: a gradient boosting machine”, *Annals of statistics*, vol. 29–5, 2001, pp. 1189–1232.

[Fri02] Friedman, J. H. “Stochastic gradient boosting”, *Computational Statistics & Data Analysis*, vol. 38–4, 2002, pp. 367–378.

[FS96] Freund, Y.; Schapire, R. E. “Experiments with a new boosting algorithm”. In: International Conference on Machine Learning, 1996, pp. 148–156.

[FS97] Freund, Y.; Schapire, R. E. “A decision-theoretic generalization of on-line learning and an application to boosting”, *Journal of computer and system sciences*, vol. 55–1, 1997, pp. 119–139.

[GP05] Goldschmidt, R.; Passos, E. “Data Mining: Um Guia Prático”. Campus, 2005, 256p.

[Har12] Harrington, P. “Machine Learning in Action”. Manning Publications, 2012, 347p.

[HMS66] Hunt, E. B.; Marin, J.; Stone, P. J. “Experiments in induction”. Academic Press, 1966, 247p.

[HPK11] Han, J.; Pei, J.; Kamber, M. “Data mining: concepts and techniques”. Elsevier, 2011, 673p.

[HS90] Hansen, L. K.; Salamon, P. “Neural network ensembles”, *Transactions on Pattern Analysis and Machine Intelligence*, vol. 12–10, 1990, pp. 993–1001.

[Kun04] Kuncheva, L. I. “Combining pattern classifiers: methods and algorithms”. John Wiley & Sons, 2004, 384p.

[LC06] Li, J.; Cercone, N. “Assigning missing attribute values based on rough sets theory”. In: International Conference on Granular Computing, 2006, pp. 607–610.

[LFLA17] Leães, A.; Fernandes, P.; Lopes, L.; Assunção, J. “Classifying with adaboost.m1: The training error threshold myth”. In: The Florida Artificial Intelligence Research Society, 2017, pp. 198–201.

[LSF08] Lopes, L.; Scalabrin, E. E.; Fernandes, P. “An empirical study of combined classifiers for knowledge discovery on medical data bases”. In: Asia-Pacific Web Conference, 2008, pp. 110–121.

[Mag00] Magalhães, Marcos Nascimento, L. A. C. P. “Noções de Probabilidade e Estatística”. EDUSP, 2000, 428p.

[MB03] Monard, M. C.; Baranauskas, J. A. “Conceitos sobre aprendizado de máquina”, *Sistemas Inteligentes-Fundamentos e Aplicações*, vol. 1–1, 2003, pp. 39–56.

[MCM13] Michalski, R. S.; Carbonell, J. G.; Mitchell, T. M. “Machine learning: An artificial intelligence approach”. Springer Science & Business Media, 2013, 738p.

[MD09] Milidiú, R.; Duarte, J. “Boosting at start”. In: International Conference on Artificial Intelligence and Applications, 2009, pp. 17–29.

[Qui86] Quinlan, J. R. “Induction of decision trees”, *Machine Learning*, vol. 1–1, 1986, pp. 81–106.

[Qui96] Quinlan, J. R. “Bagging, boosting, and C4.5”. In: Conference on Innovative Applications of Artificial Intelligence, 1996, pp. 725–730.

[Qui14] Quinlan, J. R. “C4.5: programs for machine learning”. Morgan Kaufmann Publishers, 2014, 299p.

[Ros96] Ross, M. S. “Stochastic Processes”. John Wiley & Sons, 1996, 510p.

[Ros06] Ross, M. S. “Simulation”. Academic Press, 2006, 274p.

[Sch90] Schapire, R. E. “The strength of weak learnability”, *Machine Learning*, vol. 5, 1990, pp. 197–227.

[SF+99] Schapire, R. E.; Freund, Y.; et al.. “A short introduction to boosting”, *Journal of Japanese Society for Artificial Intelligence*, vol. 14–5, 1999, pp. 771–780.

[SF12] Schapire, R. E.; Freund, Y. “Boosting: Foundations and algorithms”. MIT press, 2012, 544p.

[SKHN08] Seiffert, C.; Khoshgoftaar, T. M.; Hulse, J. V.; Napolitano, A. “Resampling or reweighting: A comparison of boosting implementations”. In: International Conference on Tools with Artificial Intelligence, 2008, pp. 445–451.

[SR14] Scuse, D.; Reutemann, P. “Weka experimenter tutorial for version 3-6-0”. Capturado em: http://www.cs.waikato.ac.nz/ml/weka/, 2014.

[SS04] Solomatine, D. P.; Shrestha, D. L. “Adaboost.RT: a boosting algorithm for regression problems”. In: International Joint Conference on Neural Networks, 2004, pp. 1163–1168.

[TSK05] Tan, P.; Steinbach, M.; Kumar, V. “Introduction to Data Mining”. Addison-Wesley, 2005, 900p.

[TXZ08] Thongkam, J.; Xu, G.; Zhang, Y. “Adaboost algorithm with random forests for predicting breast cancer survivability”. In: International Joint Conference on Neural Networks, 2008, pp. 3062–3069.

[WF11] Witten, I. H.; Frank, E. “Data Mining: Practical machine learning tools and techniques”. Morgan Kaufmann, 2011, 664p.

[YM12] Yuan, B.; Ma, X. “Sampling + reweighting: Boosting the performance of adaboost on imbalanced datasets”. In: International Joint Conference on Neural Networks, 2012, pp. 1–6.

[ZZRH09] Zhu, J.; Zou, H.; Rosset, S.; Hastie, T. “Multi-class adaboost”, *Statistics and its Interface*, vol. 2–3, 2009, pp. 349–360.

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Graduação
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: prograd@pucrs.br
Site: www.pucrs.br