UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
DEPARTAMENTO DE ESTATÍSTICA E INFORMÁTICA
PROGRAMA DE PÓS–GRADUAÇÃO EM INFORMÁTICA APLICADA

THIAGO RODRIGUES DE FRANÇA

ABORDAGEM EVOLUCIONÁRIA COM MÚLTIPLOS OBJETIVOS PARA GERAÇÃO DE BASES DE DADOS SINTÉTICOS

RECIFE – PE

2021

THIAGO RODRIGUES DE FRANÇA

ABORDAGEM EVOLUCIONÁRIA COM MÚLTIPLOS OBJETIVOS PARA GERAÇÃO DE BASES DE DADOS SINTÉTICOS

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Informática Aplicada da Universidade Federal Rural de Pernambuco, como parte dos requisitos necessários para obtenção do grau de Mestre.

ORIENTADOR: Prof. Dr. Péricles Barbosa Cunha de Miranda

RECIFE – PE

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

F814a França, Thiago Rodrigues
ABORDAGEM EVOLUCIONÁRIA COM MÚLTIPLOS OBJETIVOS PARA GERAÇÃO DE BASES DE DADOS SINTÉTICOS / Thiago Rodrigues França. - 2021.
74 f. : il.

Orientador: Pericles Barbosa Cunha de Miranda.
Inclui referências.

Dissertação (Mestrado) - Universidade Federal Rural de Pernambuco, Programa de Pós-Graduação em Informática Aplicada, Recife, 2022.

1. Geração de dados sintéticos. 2. Algoritmos de otimização. 3. Medidas de complexidade. 4. Computação evolutiva. 5. Otimização de muitos objetivos. I. Miranda, Pericles Barbosa Cunha de, orient. II. Título

CDD 004

THIAGO RODRIGUES DE FRANÇA

ABORDAGEM EVOLUCIONÁRIA COM MÚLTIPLOS OBJETIVOS PARA GERAÇÃO DE BASES DE DADOS SINTÉTICOS

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Informática Aplicada da Universidade Federal Rural de Pernambuco, como parte dos requisitos necessários para obtenção do grau de Mestre.

Aprovada em: 20 de 08 de 2021.

BANCA EXAMINADORA

Prof. Dr. Péricles Barbosa Cunha de Miranda (Orientador)
Universidade Federal Rural de Pernambuco
Departamento de Computação

Prof. Dr. Rafael Ferreira Mello
Universidade Federal Rural de Pernambuco
Departamento

Prof. Dr. Ricardo Bastos Cavalcante Prudêncio
Universidade Federal de Pernambuco
Centro de Informática

Dedico esse trabalho a Deus, a meus familiares e amigos, a todos os bons professores que passaram na minha vida e a Dona Mirian "Im Memorian" que sempre torceu e se alegrou com minhas conquistas.

Agradecimentos

Agradeço a Deus por me dar força e saúde para seguir, mesmo em momentos difíceis, agradeço aos meus pais por sempre me apoiarem no caminho do estudo, aos meus irmãos por todo o apoio e por sempre torcerem por mim, agradeço aos meus amigos Leonardo e Lairson pelo grande incentivo e ajuda.

Agradeço ao Prof. Péricles Miranda por todo o conhecimento a mim passado e por toda a orientação e motivação em momentos difíceis, pela compreensão e parceria em todos os momentos da pesquisa. Não poderia ter um melhor orientador.

Agradeço a minha esposa Thuany por sempre me apoiar e incentivar em meus sonhos e objetivos, a meus filhos eu agradeço pela paciência e compreensão nos momentos em que me ausentei ou que, mesmo presente fisicamente, estava longe deles no meio das pesquisas.

A todos os que contribuíram direta ou indiretamente com esse trabalho, o meu muito obrigado!

Uma máquina consegue fazer o trabalho de 50 homens ordinários. Nenhuma máquina consegue fazer o trabalho de um homem extraordinário. (Elbert Hubbard)

Resumo

A avaliação de algoritmos de aprendizado de máquina supervisionados em uma tarefa particular geralmente é feita por meio de avaliação empírica em dados observacionais do mundo real. No entanto, às vezes não há dados anotados anteriormente disponíveis. Conjuntos de dados sintéticos têm ganhado destaque como alternativa para avaliação eficaz de classificadores, uma vez que são acessíveis, rotulados e de complexidade parametrizável. A geração de tais conjuntos de dados sintéticos com características conhecidas trás muito valor para o processo de avaliação de algoritmos. Esses conjuntos de dados podem ser gerados levando em consideração uma variedade de métricas de descrição, tais como, como número de atributos, número de instâncias e distribuição dos dados. A geração de bases de dados com diferentes níveis de complexidades, bem anotados, é uma tarefa que pode trazer bastante benefício na avaliação de desempenho de classificadores em cenários diversos.

O presente trabalho trata a tarefa de geração de dados sintéticos com atributos parametrizáveis. Entre esses atributos podemos citar a distribuição dos pontos e seu ruído, número de instâncias, número de atributos e de classes. Além disso, como maior contribuição deste trabalho, tratamos a geração de dados de complexidade parametrizável como um problema de otimização de muitos objetivos, pois é possível escolher quantos atributos de complexidade serão otimizados e quais serão considerados.

Os resultados mostram que a proposta é capaz de gerar dados sintéticos com valores de atributos especificados pelo usuário, bem como gerar dados com diferentes níveis de complexidades, mesmo o presente problema de otimização apresentando objetivos conflitantes. Para fins experimentais, a abordagem foi avaliada na geração de dados sintéticos considerando-se quatro cenários diferentes de parametrização de complexidade. A abordagem foi comparada com a estratégia exaustiva e aleatória de geração de dados sintéticos, apresentando resultados aderentes à abordagem exaustiva, ou seja, satisfazendo a parametrização desejada pelo especialista. Além disso, para complementar a análise, algoritmos de classificação foram treinados e testados nas bases geradas com o intuito de analisar seus comportamentos em diferentes níveis de complexidade.

Os resultados mostraram que a geração de dados sintéticos com complexidades conhecidas são úteis para verificar o comportamento e desempenho dos algoritmos, inclusive, podendo a comparação dos resultados de desempenho de um determinado algoritmo para cada

nível de problema (fácil, médio, difícil e muito difícil) serem usadas como apoio para recomendação do algoritmo para um tipo de problema.

Palavras-chave: Geração de dados sintéticos. Algoritmos de otimização. Medidas de complexidade. Computação evolutiva. Otimização de muitos objetivos.

Abstract

The evaluation of supervised machine learning algorithms on a particular task is usually done through empirical evaluation on real-world observational data. However, sometimes previously annotated data is not available. Synthetic datasets have gained prominence as an alternative for the effective evaluation of classifiers, since they are accessible, labeled and with parameterizable complexity.

The generation of such synthetic datasets with known characteristics brings a lot of value to the algorithm evaluation process. These datasets can be generated taking into account a variety of description metrics such as number of attributes, number of instances and data distribution. The generation of well-annotated databases with different levels of complexity is a task that can be very beneficial in evaluating the performance of classifiers in different scenarios.

The present work deals with the task of generating synthetic data with parameterizable attributes. Among these attributes we can mention the distribution of points and their noise, number of instances, number of attributes and classes. Furthermore, as a major contribution of this work, we treat the generation of parameterizable complexity data as an optimization problem with many objectives, as it is possible to choose how many complexity attributes will be optimized and which will be these attributes.

The results show that the proposal is capable of generating synthetic data with attribute values specified by the user, as well as generating data with different levels of complexity, even with the present optimization problem presenting conflicting objectives. For experimental purposes, the approach was evaluated in the generation of synthetic data considering four different complexity parameterization scenarios.

The approach was compared with the exhaustive and random strategy for generating synthetic data, presenting results that adhere to the exhaustive approach, that is, satisfying the parameterization desired by the specialist. Furthermore, to complement the analysis, classification algorithms were trained and tested on the generated bases in order to analyze their behavior at different levels of complexity.

The results showed that the generation of synthetic data with known complexities are useful to verify the behavior and performance of algorithms, including the comparison of performance results for a given algorithm for each level of problem (easy, medium, difficult

and very difficult) be used as support for recommending the algorithm for a type of problem.

Keywords: Synthetic Dataset generation. Optimization Algorithms. Complexity measures. Evolutionary computation. Many-objectives Optimization.

Lista de Figuras

Figura 1 – Exemplo de um conjunto de soluções com objetivos conflitantes e sua frente de Pareto traçada em vermelho. 24
Figura 2 – Distribuição de 15 pontos de referência em um hiperplano tridimensional com 4 divisões para casa eixo de objetivo. 25
Figura 3 – Exemplo da métrica F1 para um conjunto de dados de duas classes. . . 27
Figura 4 – Pipeline da geração de dados sintéticos baseados em complexidade. . . 31
Figura 5 – Arquitetura da geração de dados sintética baseada em complexidade. . 36
Figura 6 – Pipeline da geração da base não rotulada. 38
Figura 7 – Distribuição dos pontos usando a função make_blobs. 39
Figura 8 – Distribuição dos pontos usando a função make_moons. 40
Figura 9 – Distribuição dos pontos usando a função make_circles. 40
Figura 10 – Processo de otimização. 42
Figura 11 – Representação do indivíduo para problemas de três classes. 42
Figura 12 – Exemplo de figura resultante do módulo de testes. 45
Figura 13 – Posição de cada instância em bases de dados geradas com diferentes níveis de complexidade. 56
Figura 14 – Posição de cada instância em bases de dados geradas com diferentes níveis de complexidade, disposição dos dados em círculos com ruído em 0,1. 58
Figura 15 – Posição de cada instância em bases de dados geradas de forma aleatória. 59
Figura 16 – Posição de cada instância em bases de dados geradas de forma aleatória. 60
Figura 17 – Comparação da acurácia obtida usando os classificadores SVM, *Naive Bayes*, *Random Forest*, e *Decision Tree* usando as bases de dados geradas com 100 instâncias e 2 atributos para cada nível de problema. 61
Figura 18 – Comparação da revocação obtida usando os classificadores SVM, *Naive Bayes*, *Random Forest*, e *Decision Tree* usando as bases de dados geradas com 100 instâncias e 2 atributos para cada nível de problema. 62
Figura 19 – Comparação da precisão obtida usando os classificadores SVM, *Naive Bayes*, *Random Forest*, e *Decision Tree* usando as bases de dados geradas com 100 instâncias e 2 atributos para cada nível de problema 63

Figura 20 – Comparação da medida F1 obtida usando os classificadores SVM, *Naive Bayes*, *Random Forest*, e *Decision Tree* usando as bases de dados geradas com 100 instâncias e 2 atributos para cada nível de problema. 64
Figura 21 – Comparação da acurácia obtida usando os classificadores SVM, *Naive Bayes*, *Random Forest*, e *Decision Tree* usando as bases de dados geradas com 100 instâncias e 10 atributos para cada nível de problema. 65
Figura 22 – Comparação da acurácia obtida usando os classificadores SVM, *Naive Bayes*, *Random Forest*, e *Decision Tree* usando as bases de dados geradas com 500 instâncias e 10 atributos para cada nível de problema. 66
Figura 23 – Telas para fornecimento de informações para a geração de bases sintéticas. 68

Lista de Tabelas

Tabela 1 – Categorias das principais medidas de complexidade 26
Tabela 2 – Trabalhos que usam otimização para geração de dados sintéticos levando em consideração a complexidade dos dados. 35
Tabela 3 – Configuração do algoritmo e valores iniciais do conjunto de dados sintéticos. 47
Tabela 4 – Valores de complexidade para cada nível de problema. 48
Tabela 5 – Valores de L2, N1, N2 e C2 dos conjuntos de dados gerados com 100 instâncias considerando 2 e 10 atributos. 50
Tabela 6 – Valores de L2, N1, N2 e C2 dos conjuntos de dados gerados com 500 instâncias considerando 2 e 10 atributos. 52
Tabela 7 – Comparação da geração de base de dados usando o a abordagem evolucionária e uma abordagem aleatória. 54

Lista de Siglas

MaOA	*Many-objective Optimization Algorithm*
MOOP	*Multi-objective Otimization Problem*
SPEA	*Strength Pareto Evolutionary Algorithm*
GA	*Genetic Algorithm*
MST	*Minimum Spanning Tree*
SVM	*Support Vector Machine*
SOOP	*single-objective otimization problem*
MOOP	*multi-objective otimization problem*
NSGA	*Non-Dominated Sorting Genetic Algorithm*
DEAP	*Ditributed Evolutionary Algorithms in Python*
Mhz	*Megahertz*
GB	*Gigabyte*
RAM	*Random Access Memory*
DDR	*Double-Data-Rate*
GNU	*GNU's Not Unix*
CbDGEN	*Complexity Based Dataset Generator*

Sumário

1 Introdução 17

1.1 Problema de Pesquisa 19

1.2 Proposta 19

1.3 Objetivos 20

1.3.1 Objetivo Geral 20

1.3.2 Objetivo Específico 20

1.4 Estrutura do Trabalho 21

2 Fundamentação Teórica 22

2.1 Algoritmos Genéticos 22

2.2 Otimização 23

2.2.1 NSGA-III 25

2.3 Complexidade dos dados 26

2.3.1 Medidas de Sobreposição de Atributos 26

2.3.2 Medidas de Linearidade 27

2.3.3 Medidas de Vizinhança 28

2.3.4 Medidas de Rede 29

2.3.5 Medidas de Dimensionalidade 29

2.3.6 Medidas de Balanceamento de Classes 30

3 Revisão da Literatura 31

4 Proposta 36

4.1 Arquitetura 36

4.1.1 Módulo de Criação de base não rotulada 37

4.1.2 Módulo de Otimização de Complexidade 37

4.1.3 Módulo de Testes 38

4.2 Implementação da Arquitetura 38

4.2.1 Criação da base não-rotulada 38

4.2.2 Módulo de Otimização 41

4.2.3 Módulo de Testes 45

5 Metodologia 46

5.1 Ambiente Experimental 46

5.1.1 Linguagem de Desenvolvimento . . . 46
5.1.2 Bibliotecas Utilizadas . . . 46
5.1.3 Ambiente de Desenvolvimento . . . 46
5.2 Execução . . . 47
5.3 Metodologia de Avaliação . . . 48
6 Resultados . . . 50
6.1 Análise da Otimização . . . 50
6.1.1 Conjunto de dados com 100 instâncias . . . 50
6.1.2 Conjunto de dados com 500 instâncias . . . 52
6.2 Proposta X Abordagem de geração aleatória . . . 54
6.3 Visualização dos dados . . . 55
6.3.1 Abordagem evolucionária . . . 55
6.3.2 Abordagem aleatória . . . 59
6.4 Análise do Desempenho dos Classificadores . . . 60
7 Conclusão . . . 67
8 Considerações Finais . . . 68
8.1 Contribuições . . . 68
8.2 CbDGEN . . . 68
8.3 Artigos Aceitos . . . 69
8.4 Limitações e Trabalhos Futuros . . . 69
Referências . . . 71
GLOSSÁRIO . . . 73

1 Introdução

Atualmente, vários algoritmos de classificação estão sendo usados em uma grande variedade de tarefas distribuídas em muitas áreas como medicina, segurança, negócios e educação (XIA et al., 2011),(HARPER, 2005), (KAUR et al., 2015). Devido à grande heterogeneidade de tais tarefas e às particularidades que tornam um algoritmo específico melhor do que outros, é necessário analisar a qualidade do classificador em termos de desempenho de classificação (acurácia, precisão, *recall* e outros) e com relação aos custos computacionais (LORENA et al., 2012). Essas análises empíricas auxiliam o especialista na seleção do classificador mais adequado para seu problema específico (LORENA et al., 2012).

Uma prática comum, para avaliar o desempenho de um ou mais classificadores, é treiná-los e testá-los em diferentes bancos de dados, a maioria destes, publicamente disponíveis. No entanto, observou-se que tal prática pode ter suas limitações (AMANCIO et al., 2014).

A primeira é que a estimativa de desempenho do classificador pode estar associada a sua própria limitação ou com a complexidade do próprio conjunto de dados (por exemplo, dados não representativos ou ausentes, desequilíbrio de classe e outros). Testar diferentes algoritmos nos mesmos conjuntos de dados pode parecer superar esse problema, pois a complexidade do conjunto de dados é compartilhada por todos os algoritmos considerados. No entanto, isso também pode levar a conclusões enganosas, pois há complexidades no conjunto de dados que podem afetar os algoritmos de aprendizagem de diferentes maneiras (AMANCIO et al., 2014).

A segunda limitação é que, mesmo que os resultados obtidos sejam consistentes, quando comparados com centenas de conjuntos de dados do mundo real os resultados ainda permanecem específicos para os conjuntos de dados considerados na avaliação experimental. Tentar estender as informações obtidas a partir dessas análises para um conjunto de dados diferente pode ser ineficaz (SOARES, 2009). Além disso, obter mais bancos de dados com características diferentes para avaliar classificadores em diferentes aspectos, é uma tarefa complicada e muitas vezes cara (SOARES, 2009).

Diante dos aspectos apresentados acima, bancos de dados sintéticos têm ganhado destaque como alternativa para avaliação classificadores, uma vez que são acessíveis e

parametrizáveis (LORENA et al., 2012). Embora os bancos de dados sintéticos não estejam diretamente associados a conjuntos de dados reais específicos, eles podem ser usados como representações de grandes classes de dados (AMANCIO et al., 2014).

Devido à natureza parametrizável das bases sintéticas, é possível criar uma base de dados que sigam características específicas, possibilitando analisar o comportamento dos classificadores em vários cenários. Diferentes estratégias têm sido propostas para a geração de bancos de dados sintéticos para problemas de classificação. Uma abordagem comum é criar exemplos seguindo uma distribuição específica (AMANCIO et al., 2014), ou por exemplo, uma determinada correlação de *Pearson* (FILHO; JUNIOR, 2009). Suponhamos que queremos criar um banco de dados sintético a partir de uma distribuição normal, com médias e variâncias diferentes para as diferentes classes. Modificando os valores das médias e variâncias, é possível construir diferentes bases de dados, com taxas distintas de classes sobrepostas.

Outra abordagem propõe a geração de bancos de dados sintéticos modificando a estrutura geométrica dos dados (Macià et al., 2009), o que pode ser feito por meio de medidas de complexidade (Macià et al., 2009), (Melo; Lorena, 2018). Essas medidas de complexidade foram introduzidas por (HO; BASU, 2002), e têm sido adotadas em diferentes tipos de análises nos últimos anos como em (MACIÀ et al., 2008-a) e (Macià et al., 2010).

As medidas de complexidade extraem características estatísticas e geométricas dos conjuntos de dados, para um determinado problema de classificação, a fim de calcular sua complexidade.

Em tais cenários, alguns estudos têm proposto a otimização de diferentes medidas a fim de gerar bancos de dados sintéticos com diferentes níveis de complexidade. Isso geralmente é obtido pela aplicação de algoritmos de otimização (mono- ou multi-objetivo) para a geração dos conjuntos de dados sintéticos. Existem abordagens que criam um novo banco de dados sintético (Macià et al., 2009), (MACIÀ et al., 2008-b) ou que fazem uma amostragem de conjuntos de dados pré-existentes (Macià et al., 2010), (Melo; Lorena, 2018), para chegar a um determinado valor de uma ou mais medidas de complexidade.

1.1 Problema de Pesquisa

A geração de dados sintéticos é motivada, principalmente, pela falta de dados previamente anotados em repositórios públicos e pela falta de disponibilidade de dados reais. Além disso, os dados sintéticos, especificamente, os gerados nesse trabalho, tem uma característica muito valorizada em experimentos e testes. Eles são parametrizáveis com relação ao seu grau de dificuldade, parâmetros esses que levam em consideração algumas métricas de complexidade que são diretamente responsáveis pelo desempenho de algoritmos de classificação, podendo o usuário gerar dados sintéticos exatamente com as características necessárias para um determinado experimento.

Um dos desafios da pesquisa é que, até o momento, os trabalhos relacionados consideraram no máximo três medidas de complexidade a serem otimizadas no processo de geração. Além disso, o problema vem se mostrando complexo e demanda muito recurso computacional e tempo. Muitas áreas de conhecimento distintas se utilizam de algoritmos de classificação e alguns algoritmos se saem melhor em um determinado tipo de competência.

Com isso, surge a seguinte questão: É possível gerar dados baseados considerando múltiplas medidas de complexidade usando a otimização de muitos objetivos?

1.2 Proposta

Este trabalho propõe um método de otimização para a geração de novas bases de dados sintéticas levando em consideração quatro diferentes medidas de complexidade que podem ser escolhidas entre as medidas de sobreposição de atributos (F1), medidas de linearidade (L2), medidas de vizinhança (N1) e (N2), medidas de rede (ClsCoef), medidas de dimensionalidade (T2) e medidas de balanceamento de classes (C2). Vale ressaltar que os trabalhos relacionados, até o momento, consideraram no máximo três medidas de complexidade a serem otimizadas no processo de geração. Como principais contribuições deste trabalho, destacamos: i) uso de um algoritmo de otimização de muitos objetivos (MaOA) para geração de dados sintéticos; ii) proposição de um setup experimental para o uso de MaOA para o problema de geração de dados baseado na complexidade e iii) proposição de uma ferramenta pública de geração de dados sintéticos baseados em complexidade.

O método proposto foi avaliado na geração de conjuntos de dados sintéticos compostos por 100 e 500 instâncias com 2 e 10 atributos sendo, os pontos, dispostos usando distribuições gaussianas diversas e variação de ruído que aumentam e diminuem a proximidade dos atributos da base. Além disso, como meio de confirmação do desempenho do algoritmo genético para muitos objetivos, utilizamos o mesmo experimento realizado com o MaOA em uma geração aleatória onde a população e todos os indivíduos são gerados de forma aleatória, sem o auxílio do MaOA. Os resultados demonstram que o problema é desafiador, e a proposta é capaz de otimizar objetivos conflitantes, gerando conjuntos de dados de complexidades específicas. As figuras comprovam que os dados sintéticos gerados trazem desafios diversos para os classificadores e podem agregar conhecimento sobre eles.

1.3 Objetivos

1.3.1 Objetivo Geral

O principal objetivo deste trabalho é a geração de conjuntos de dados sintéticos baseados em quatro medidas de complexidade usando algoritmos de otimização de muitos objetivos para buscar valores e atributos específicos, gerando assim bases de dados relevantes para o processo de avaliação de classificadores.

1.3.2 Objetivo Específico

Para atingir o objetivo geral foram definidos os seguintes objetivos específicos:

- Realizar revisão da literatura verificando técnicas de geração de dados sintéticos que usam abordagens de otimização;
- Análise das medidas de complexidade existentes;
- Análise dos parâmetros que serão utilizados pelo algoritmo de otimização;
- Implementação e adaptação do algoritmo de otimização para a geração de conjunto de dados sintéticos baseados na complexidade;
- Teste dos dados gerados em classificadores para análise dos seus resultados;

1.4 Estrutura do Trabalho

Esta dissertação é comporta pelo capítulo de introdução e mais 7 capítulos organizados da seguinte forma:

Capítulo 2: São apresentados os conceitos básicos necessários para o entendimento deste trabalho, tais como, algoritmos genéticos, otimização, e conceitos de medidas de complexidade dos dados.

Capítulo 3: Neste capítulo tratamos dos trabalhos relacionados, mostrando o que já foi estudado sobre geração de conjuntos de dados usando abordagens evolucionárias.

Capítulo 4: Neste capítulo é apresentada a proposta da dissertação, sua motivação e a formalização do trabalho desenvolvido.

Capítulo 5: Neste capítulo são apresentados os experimentos desenvolvidos e como os mesmos foram realizados.

Capítulo 6: Neste capítulo são apresentados os resultados obtidos nos experimentos, bem como a visualização e análise dos dados.

Capítulo 7: Neste capítulo são apresentadas nossas conclusões sobre todo o trabalho desenvolvido.

Capítulo 8: Neste capítulo são apresentadas as contribuições, limitações e trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta conceitos básicos necessários para um melhor entendimento sobre o trabalho desenvolvido.

2.1 Algoritmos Genéticos

A Computação Evolutiva é uma subárea de estudo da Inteligência Artificial que busca, nos conceitos de evolução, inspiração para construir sistemas de otimização. Dentre os métodos pertencentes a essa subárea, temos os algoritmos genéticos (GAs) que foram introduzidos por (HOLLAND et al., 1992). Segundo (LINDEN, 2008), GAs são um ramo dos algoritmos evolucionários que podem ser definidos como uma técnica de busca baseada em uma metáfora do processo biológico de evolução natural, Linden (2008)/ ainda destaca que algoritmos genéticos são técnicas heurísticas de otimização global.

Os algoritmos genéticos usam um processo análogo ao biológico para encontrar soluções para problemas de otimização, tratando possíveis soluções como indivíduos e organizando suas informações como em um cromossomo. A coleção de indivíduos é chamada de população, e os operadores genéticos (mutação, cruzamento, seleção) são aplicados sobre os indivíduos promovendo a evolução (DAVIS, 1991). O espaço composto por todas as possíveis soluções candidatas à resolução do problema é chamado de espaço de busca. As soluções do espaço de busca são avaliadas através de uma função de aptidão (*fitness*), que quantifica a qualidade da solução. A função de *fitness* é particular, e varia de acordo com o que deseja ser resolvido. Considerando como exemplo um problema para geração de textos com um português correto, uma possível função de *fitness* seria a quantidade de número de erros. (KOZA, 2004). O *fitness*, na maioria das vezes, é usado na etapa de seleção onde podemos fazer com que os indivíduos mais bem avaliados sejam selecionados para a reprodução e, em alguns casos, indivíduos com classificações baixas podem ser selecionados com o objetivo de manter a diversidade na população.

2.2 Otimização

Quando um problema de otimização é representado por uma única função de *fitness* ele é definido como "mono-objetivo"ou SOOP (*single-objective otimization problem*). Nessa classe de problemas, os métodos de otimização tentam encontrar uma solução "viável"que represente o mínimo (ou máximo) da função objetivo. Existem casos que se procura obter a "melhor resposta"em mais de um aspecto do sistema e, consequentemente, atender a vários objetivos simultaneamente. Esse tipo de problema é definido como "multiobjetivo" ou MOOP (*multi-objective otimization problem*).

Problemas de otimização com dois a três objetivos conflitantes são conhecidos como problemas multiobjetivos. Esses problemas tentam encontrar soluções que satisfaçam todos os objetivos conflitantes ao mesmo tempo. Problemas multiobjetivos podem ser caraterizados da seguinte forma: dado um vetor de funções ($\vec{f}$) cujos componentes representam funções objetivo a serem otimizadas, o objetivo é encontrar um vetor de soluções que atendam às restrições definidas e também otimize $\vec{f}$ ao mesmo tempo.

Mais especificamente, considere o espaço de soluções definido como S, e um conjunto de funções objetivos $\vec{f}(\vec{x}) := [f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x})]$, onde $\vec{x} = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$ é o vetor de parâmetros em S. A tarefa de otimização é encontrar um conjunto de soluções $S^{opt} \subset S$, com base no vetor de funções objetivo $\vec{f}(\vec{x})$. Um problema geral de otimização multiobjetivo pode ser definido como:

$$\text{minimize } \vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x})] \tag{2.1}$$

Sujeito a:

$$g_l(\vec{x}) \leq 0 \quad l = 1, 2, ..., p, \tag{2.2}$$

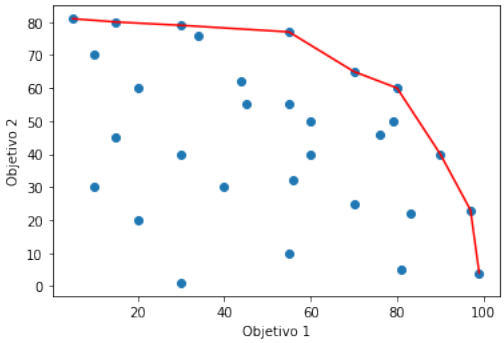
$$h_j(\vec{x}) = 0 \quad j = 1, 2, ..., q, \tag{2.3}$$

onde $\vec{x} = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$ é um vetor no espaço de busca; k é o numero de objetivos e $g_l(\vec{x})$ e $h_j(\vec{x})$ são as funções objetivos e a soma de p e q é o numero de objetivos do problema.

Em um problema de otimização multiobjetivo não-trivial, não existe um único esquema que maximize simultaneamente todos os critérios. Nesse caso, diz-se que o

problema possui objetivos conflitantes e existe um conjunto (possivelmente infinito) de soluções ótimas. Esse conjunto é comumente chamado de conjunto (ou frente) de Pareto. Para dois ou três objetivos, um conjunto de Pareto pode ser visualizado em um gráfico de dispersão (*scatter plot*) que tenha em seus eixos a resposta para cada objetivo como mostrado na Figura 1.

Figura 1 – Exemplo de um conjunto de soluções com objetivos conflitantes e sua frente de Pareto traçada em vermelho.



Fonte: O Autor.

Os algoritmos multiobjetivos geralmente funcionam muito bem em problemas com dois e três objetivos. No entanto, segundo (HISAO ISHIBUCHI et al., 2008), a capacidade de busca de algoritmos multiobjetivos como o *Strength Pareto Evolutionary Algorithm* (SPEA) (ZITZLER; THIELE, 1999) e o *Non-Dominated Sorting Genetic Algorithm II* (NSGA-II) (DEB et al., 2002) é deteriorada conforme o número de objetivos aumenta. Conforme isso acontece, quase todas as soluções em cada população se tornam não dominantes. Isso enfraquece o modelo de seleção baseada na dominância de Pareto e a propriedade de convergência dos algoritmos multiobjetivos é severamente impactada, visto que a frente de Pareto é uma hiper-superfície, e sua aproximação aumenta exponencialmente com o aumento do número de objetivos.

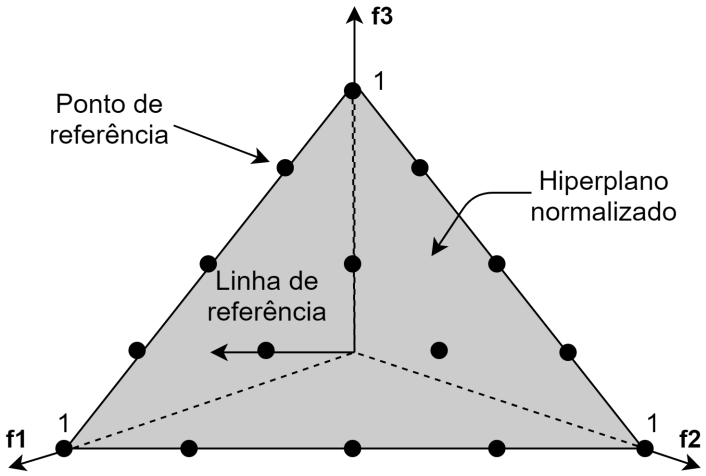
Problemas multiobjetivos com quatro ou mais objetivos são frequentemente chamados de problemas com muitos objetivos ou *Many-objective Optimization*. Alguns estudos como (ANDRA; Fleming, 2011), (HADKA; REED, 2013), (BADER; Zitzler, 2011) entre outros, sugerem alterações em algoritmos multiobjetivos para melhorar seu funcionamento em problemas de muitos objetivos, trazendo resultados promissores. No entanto, algoritmos exclusivos para o cenário de muitos objetivos passaram a ser desenvolvido, como o NSGA-III (DEB; JAIN, 2014).

2.2.1 NSGA-III

Segundo (DEB; JAIN, 2014), o NSGA-III é uma evolução do NSGA-II (DEB et al., 2002) e funciona de forma muito semelhante. O NSGA-III usa um conjunto de pontos de referência para garantir a diversidade nas soluções obtidas, esse procedimento é descrito da seguinte forma:

Primeiro, o usuário define um conjunto de pontos de referência H ou uma abordagem estruturada é utilizada para criá-los, o conjunto H contém p divisões para cada eixo objetivo, em um hiperplano M-dimensional que passa por cada eixo objetivo no ponto $1, 0$.

Figura 2 – Distribuição de 15 pontos de referência em um hiperplano tridimensional com 4 divisões para casa eixo de objetivo.



Fonte: Adaptado de (Deb; Jain, 2014).

Todo ponto de referência tem como origem o ponto ideal da população no espaço de busca e dá origem a uma direção de referência, então, outro hiperplano M-dimensional é encontrado identificando os pontos externos em cada eixo objetivo, os objetivos são normalizados de modo que o novo hiperplano coincida com o hiperplano marcado com os pontos de referência, em seguida, os indivíduos são associados à direção de referência mais próxima no espaço objetivo normalizado e finalmente, uma vizinhança é formada pelos indivíduos mais próximos de uma determinada direção. Vizinhanças pequenas são melhores para manter a diversidade.

O resultado ideal no fim da execução seria um membro para cada ponto de referência. A figura 2 trás um exemplo de $H = 15$ pontos de referência estruturados e distribuídos em um hiperplano normalizado, com $M = 3$ objetivos e $p = 4$ divisões. Também são descritos

o ponto ideal e uma das direções de referência. Os pontos de referência H foram criados usando a abordagem sistemática de (DAS; DENNIS, 1998) que é dada por:

$$H = \binom{M + p - 1}{p} \tag{2.4}$$

Existem outros algoritmos que podem ser usados para problemas de muitos objetivos, o *NSGA-III* tem alcançado resultados promissores nesse tipo de problema como visto em (FRANÇA et al., 2020), por isso o mesmo foi adotado nesse trabalho.

2.3 Complexidade dos dados

Tabela 1 – Categorias das principais medidas de complexidade

Classificação	**Breve descrição**
Medidas de sobreposição de atributos	Mostram o quão informativos são os atributos disponíveis para separar as classes
Medidas de linearidade	Tentam demostrar em números a possibilidade das classes serem separadas linearmente
Medidas de vizinhança	Caracterizam a presença e densidade de classes iguais ou diferentes nos vizinhos locais
Medidas de rede	Extraem informações estruturais do conjunto de dados o modelando como em um gráfico
Medidas de dimensionalidade	Avalia a dispersão dos dados com base no número de amostras em relação à dimensionalidade
Medidas de balanceamento de classes	Consideram a proporção do número de exemplos entre as diferentes classes

Fonte: O autor

2.3.1 Medidas de Sobreposição de Atributos

Essa categoria possui medidas que avaliam o poder discriminativo dos atributos. Se houver pelo menos uma característica muito discriminativa no conjunto de dados, o problema será considerado mais simples do que se não houvesse nenhum atributo com

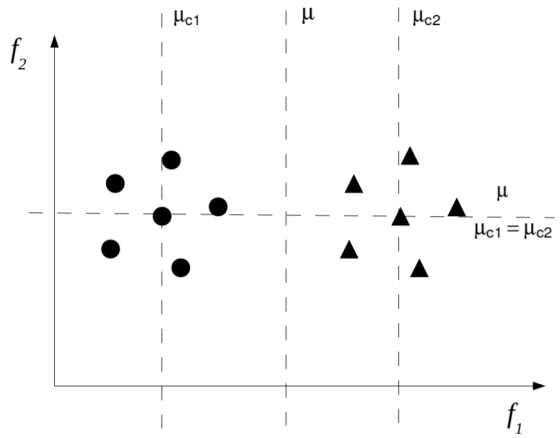
essas características. As medidas dessa categoria requerem que os atributos tenham valores numéricos e a maioria delas são definidas apenas para problemas de classificação binária.

Dentre as medidas dessa categoria, destacamos o *Maximum Fisher's Discriminant Ratio* (F1), que mede a sobreposição entre os valores dos atributos em diferentes classes, e é dada pela equação 2.5.

$$F1 = \max_{i=1}^{m} rf_i, \tag{2.5}$$

onde rf_i é a razão discriminante para cada atributo f_i, ou seja, F1 assume o valor da maior razão discriminante entre todos os atributos disponíveis. Tomando como exemplo o conjunto de dados mostrado na Figura 3, o atributo mais discriminativo seria o f1. A métrica F1 indica que as classes podem ser facilmente separáveis usando esse atributo, já f2 não é discriminativo, pois seus valores para as duas classes se sobrepõem.

Figura 3 – Exemplo da métrica F1 para um conjunto de dados de duas classes.



Fonte: (LORENA et al., 2019).

2.3.2 Medidas de Linearidade

Essas medidas procuram quantificar o quão linearmente separáveis são os exemplos de classes diferentes, ou seja, se é possível separá-las em um hiperplano. É motivada pela suposição de que um problema separável linearmente pode ser considerado mais simples do que um problema que exija uma decisão não linear.

Dentre as métricas deste grupo, destacamos *Error Rate of Linear Classifier (L2)* que, como proposto em (ORRIOLS-PUIG et al., 2010), calcula a taxa de erro do classificador

linear SVM, seja $h(x)$ o valor obtido pelo classificador linear. L2, então, é dada pela equação 2.6.

$$L2 = \frac{\sum_{i=1}^{n} I(h(x_i) \neq y_i)}{n}. \tag{2.6}$$

Valores maiores de L2 representam problemas mais complexos em relação ao fato de que os dados não podem ser separados linearmente.

2.3.3 Medidas de Vizinhança

Essas medidas tentam capturar a forma do limite de decisão e caracterizar a sobreposição de classes analisando as vizinhanças locais dos pontos. Algumas das métricas desse grupo também capturam a estrutura interna das classes. Para isso, estas métricas usam uma matriz de distâncias entre todos os pares de pontos no conjunto de dados. Entre as medidas de vizinhança destacamos a *Fraction of Borderline Points (N1)*. Nessa medida inicialmente, é gerada uma *Minumum Spanning Tree* (MST) a partir dos dados onde cada vértice corresponde a um exemplo e as arestas são ponderadas de acordo com a distância entre eles. A medida de N1 é obtida calculando a porcentagem de vértices conectando exemplos de classes opostas no MST gerado. Portanto, N1 estima a complexidade do limite de decisão por meio do limite de pontos críticos no conjunto de dados. Quanto maior o valor de N1, mais complexo é considerado o conjunto de dados. N1 pode ser expresso pela equação 2.7.

$$N1 = \frac{1}{n} \sum_{i=1}^{n} I((X_i, X_j) \in MST \wedge y_i \neq y_j). \tag{2.7}$$

N1 é sensível ao tipo de ruído onde os vizinhos mais próximos têm uma classe diferente da sua. Conjunto de dados com esse tipo de ruído são considerados mais difíceis do que conjuntos de dados equivalentes limpos conforme observa (LORENA et al., 2012).

Uma outra métrica interessante entre as de vizinhança é a *Ratio of Intra/Extra Class Nearest Neighbor Distance* (N2). Essa medida calcula a razão da média entre as distâncias do vizinho mais próximo de mesmas classe e de classes diferentes como mostrado na equação 2.8.

$$N2 = \frac{\sum_{i=1}^{n} d(x_i, NN(x_i) \in y_i)}{\sum_{i=1}^{n} d(x_i, NN(x_i) \in y_i) \neq y_i}, \tag{2.8}$$

onde $d(x_{i,}NN(x_i) \in y_i)$ corresponde a distância do exemplo x_i para seu vizinho mais próximo (NN) que tenha a classe igual a sua e $\sum_{i=1}^{n} d(x_i, NN(x_i) \in y_i) \neq y_i$ representa a distância entre x_i e seu vizinho mais próximo que tenha a classe diferente da sua.

2.3.4 Medidas de Rede

Essas medidas modelam o conjunto de dados como um grafo e usam extratores de características de redes complexas. Foi observado em (GARĆIA et al., 2013) que medidas de rede, apesar de semelhantes as medidas de vizinhança presentes em (HO; BASU, 2002) e no tópico anterior, trazem informações diferentes umas das outras, o que sustenta a relevância de explorar também essas medidas. Uma das medidas de rede é a *Clustering coefficient* (ClsCoef) que calcula o coeficiente de agrupamento de um vértice v_i (cada vértice é uma instância da base). Este coeficiente é calculado pela razão entre o número de arestas entre seus vizinhos e o número máximo de arestas possíveis entre eles. Bases de dados mais simples tenderão a ter conexões densas entre exemplos da mesma classe. Esta medida é calculada através da equação 2.9.

$$ClsCoef = 1 - \frac{1}{n}\sum_{i=1}^{n}\frac{2|e_{jk} : v_j, v_k \ \epsilon \ N_i|}{k_i\,(k_i - 1)}, \tag{2.9}$$

onde $N_i = \{v_j : e_{ij} \in E\}$ denota o conjunto de vizinhos de um vértice v_i e k_i é o tamanho de N_i. A medida de ClsCoef será maior para conjuntos de dados mais simples que terão conexões mais densas entre exemplos da mesma classe.

2.3.5 Medidas de Dimensionalidade

Segundo (LORENA et al., 2019), medidas desse grupo fornecem indicativo de dispersão dos dados. A ideia é que pode ser mais difícil extrair bons modelos de conjunto de dados esparsos devido à provável presença de regiões com baixa densidade que serão classificadas arbitrariamente. Dentre as medidas de dimensionalidade, destacamos a *Average number of points per dimension* (T2) que divide o número de exemplo do conjunto de dados pela sua dimensionalidade como na equação 2.10.

$$T2 = \frac{n}{m} \tag{2.10}$$

2.3.6 Medidas de Balanceamento de Classes

Segundo (LORENA et al., 2019), essas medidas tentam capturar um aspecto que pode influenciar muito o desempenho preditivo das técnicas de aprendizagem de máquina: o equilíbrio das classes. Quando essas diferenças são altas, a maioria das técnicas de aprendizagem de máquina favorece a classe majoritária e apresenta problemas de generalização. Quanto maior for o desequilíbrio entre as classes, maior a complexidade do conjunto de dados. Entre as medidas deste grupo, destacamos a *Imbalance ratio* (C2), proposta em (TANWANI; FAROOQ, 2009), que calcula o equilíbrio das classes através da equação 2.11:

$$C2 = \frac{n_c - 1}{n_c} \sum_{i=1}^{n_c} \frac{n_i}{n - n_i}, \tag{2.11}$$

onde n_i é o número de instâncias na i-ésima classe, o valor mínimo de C2 é alcançado para problemas balanceados. Esta medida é adequada também para cálculos de C2 em problemas multi-classes.

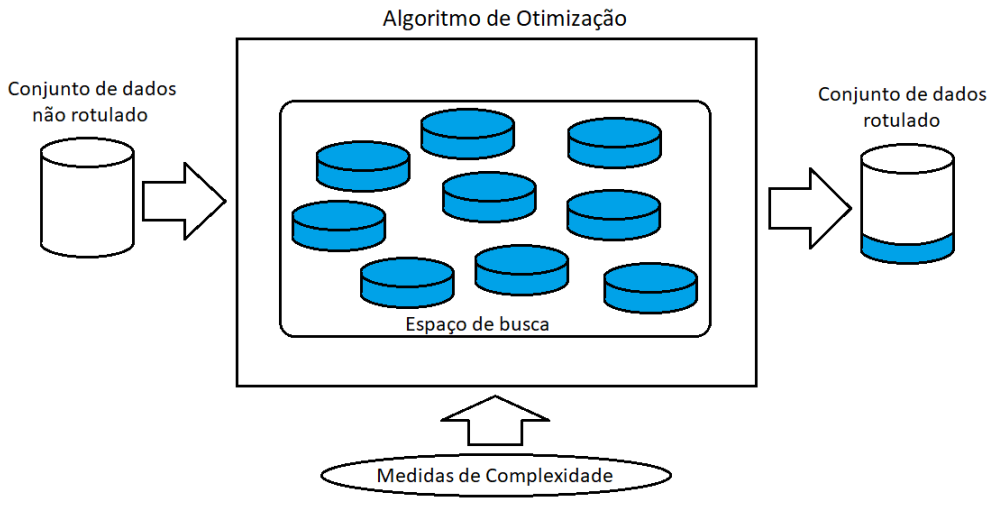
Mais detalhes sobre as medidas de complexidade podem ser encontrados em (LORENA et al., 2019), (HO; BASU, 2002) e (GARĆIA et al., 2013).

3 Revisão da Literatura

Encontrar dados com características específicas e em grande volume que possam ser usados no processo de avaliação de classificadores não é trivial (MUñOZ; SMITH-MILES, 2017). A geração de dados com complexidades conhecidas possibilita a criação de conjuntos de dados com características específicas (e.g., número de exemplos, ruído, distribuição e outros) ou que podem se assemelhar a dados reais. Segundo (POPIć et al., 2019b), a síntese dos dados é a simulação usada como meio para geração dos dados como descritos pelo usuário e existem diversos tipos de sínteses como sequências, números aleatórios, ruídos e funções matemáticas. Elas podem ser usadas individualmente ou em conjunto.

A geração de dados sintéticos é um assunto que vem ganhando atenção na comunidade de aprendizagem de máquina, e nesse capítulo iremos citar trabalhos alguns que se relacionam, de alguma forma, com esta pesquisa. Dentre as técnicas de geração de dados já conhecidas como as técnicas de aprendizagem de máquina, algoritmos de compressão e codificação entre outras presentes em (POPIć et al., 2019a), destacamos a geração de dados sintéticos baseados em complexidade. Essa técnica é exemplificada na Figura 4, onde:

Figura 4 – Pipeline da geração de dados sintéticos baseados em complexidade.



Fonte: O autor.

- Inicialmente, um conjunto de dados não rotulado é passado para o algoritmo de otimização;

- O algoritmo de otimização navega por um espaço de busca com uma variedade de possíveis rótulos. Os rótulos possíveis serão atribuídos ao conjunto de dados, e a medida de complexidade de cada par (conjunto de dados não rotulado + rótulo do espaço de busca) será calculada e armazenada;
- Esse processo se repete com novos itens no espaço de busca gerados pelo processo de mutação e cruzamento detalhados no Capítulo 5;
- Valores de referência para medidas de complexidades que o usuário deseja alcançar são passadas para o algoritmo;
- Por fim, após todas as execuções, o algoritmo retorna o conjunto de dados junto com o rótulo que teve suas medidas de complexidades mais próximas das medidas passadas pelo usuário.

A síntese da revisão da literatura relacionada à geração de dados sintéticos baseada em complexidades resultou nos trabalhos presentes na Tabela 2.

O destaque vai para a pesquisadora Nuria Marciá, especialista em aprendizagem de máquina e complexidade dos dados, foi pioneira na geração de dados sintéticos baseados em medidas de complexidade usando algoritmos de otimização e publicou a maioria dos trabalhos citados em nossa revisão da literatura.

O trabalho de MACIÀ et al., 2008-a propõe a geração de conjuntos de dados sintéticos com base na complexidade dos dados. Nele os autores se concentram na métrica de complexidade *Fraction of borderline points* (N1). Inicialmente foi restringido no número de classes para duas classes. Um problema tradicional de classificação binária, o conjunto de dados foi caracterizado como: número de recursos *features*, número de instâncias e a complexidade desejada. Depois que esses parâmetros foram definidos, foram gerados n pontos com o valor dos atributos distribuídos uniformemente. Em seguida, as classes de cada ponto foram rotuladas de acordo com o valor e (N1) especificado. Para isso foi construída uma MST *(minimum spanning tree)* e com a ajuda dele, é possível rotular a classe dos pontos para atingir a complexidade desejada.

O trabalho mostra que para um dado MST e valor de N1 = b, teremos $\left(\frac{n-1}{(n-1)-p)}\right)$ diferentes rótulos onde p é o número de arestas conectando classes diferentes, podemos observar que a métrica é um fator dominante na complexidade dos dados e que influencia diretamente no desempenho de classificadores. Os conjuntos de dados sintéticos gerados no experimento correspondem a complexidade requerida e tem o desempenho esperado. Os

autores acreditam que isso pode fornecer um conjunto prático de referências para avaliar a qualidade dos classificadores e identificar os tipos de problemas em que cada classificador é mais adequado.

Em MACIÀ et al., 2008-b é utilizada a mesma métrica de medição de complexidade de dados (N1) como referência para geração dos dados sintéticos. No entanto, a diferença está no modo como os dados são gerados. Nesse trabalho foi utilizado um algoritmo genético mono-objetivo para otimizar o valor de complexidade de cada conjunto de dados. A base de dados não rotulada foi gerada usando uma distribuição uniforme e os testes foram realizados com bases de 101 e 501 instâncias, todas as bases de dados geradas tinham 2 atributos. O tamanho do espaço de solução deste problema de otimização aumenta exponencialmente com o número de instâncias de entrada, tornando os mecanismos tradicionais de busca exaustiva inadequados para gerar conjuntos de dados de médio ou grande porte com uma dada complexidade de contorno. No trabalho foi projetado um algoritmo genético com uma população de tamanho fixo para resolver o problema de geração de conjuntos de dados com uma complexidade definida pelo usuário. O trabalho mostrou que é possível usar algoritmos genéticos para gerar conjuntos de dados sintéticos baseados na métrica de complexidade N1.

O trabalho de Macià et al., 2009 foi um dos primeiros a trazer a possibilidade de otimização de até 3 medidas de complexidades. Nesse trabalho os autores usam um algoritmo multi-objetivo, o NSGA-II (do inglês *Non-dominated Sorting Genetic Algorithm II*), para otimizar os valores de complexidade desejados. As medidas otimizadas foram N1 (*Fraction of borderline points*), N2 (*Ratio of average intra/internearest neighbor distance*) e F1 (*Ratio of the maximum Fisher's discriminant*). Os resultados foram promissores para 3 objetivos. O trabalho em questão usou em seu experimento uma distribuição uniforme e uma distribuição baseada no problema Iris do repositório UCI (ASUNCION; NEWMAN, 2007). Além da limitação relacionada ao número máximo de medidas de complexidade que serão otimizadas em simultâneo, o trabalho está limitado ao estudo das medidas de complexidade disponíveis na época.

O mesmo foi feito em Macià et al., 2010 onde o mesmo algoritmo de otimização, NSGA-II, foi utilizado. Porém, outras medidas foram otimizadas, a F2 (*Volume of overlapping region*), N4 (*Non linearity of NN classifier*) e T1 (*Fraction of maximum coveringspheres*). Além disso foi proposto uma seleção de amostragem de problemas do

mundo real, onde o conceito é descrito pelas amostras coletadas nos experimentos ou de conjuntos de dados sintéticos seguindo distribuições específicas e conhecidas. Assim, o problema ainda é como um problema de otimização multiobjetivo, mas aqui baseado na seleção de instâncias onde os conjuntos de dados têm que obedecer à minimização ou maximização dos valores das medidas de complexidade especificadas e das restrições internas. Como restrições para o problema de otimização, foram considerados: o número mínimo de instâncias, o desequilíbrio de classes, o cálculo das medidas de complexidade e a duplicidade de instâncias. Esse trabalho utilizou como base para a geração dos seus dados um conjunto de instâncias rotuladas que podem ser extraídas de dados do mundo real ou de dados sintéticos. Ao longo do trabalho, podemos observar resultados interessantes que apontaram a importância da estrutura do problema para se ter sucesso na geração de conjuntos de dados com complexidade específica. O sistema também serviu para confirmar que problemas com uma certa complexidade que não encontramos nos repositórios públicos podem ser gerados.

O trabalho de Melo; Lorena, 2018 utilizou o algoritmo de busca local *hill-climbing* para otimizar as medidas de complexidade F1 (*Ratio of the maximum Fisher's discriminant*), N1 (*Fraction of borderline points*) e N3 (*Error rate of NN classifier*) sendo uma por vez. Nesse trabalho os pesquisadores levaram em consideração número de instancias $n = 100$ e 1000, número de atributos $n_f = 2$ e 1000 e o número de classes $n_c = 2$ e 5. Foi observado que a proposta foi capaz de gerar os dados sintéticos com as complexidades desejadas em um tempo aceitável. O tempo de processamento da otimização não é afetado pelo número de classes e, dependendo da medida de complexidade adotada, o número de recursos também não influencia no tempo de processamento. Há, entretanto, um tempo de inicialização para a criação de algumas estruturas de dados iniciais.

O presente trabalho tem como diferencial a otimização de 4 medidas de complexidade de uma única vez usando um algoritmo genético para muitos objetivos. Essa abordagem se mostra desafiadora pois muitas das medidas de complexidade existentes são conflitantes. Além disso, traz como diferencial a possibilidade de parametrizar a distribuição dos pontos podendo escolher entre uma simples distribuição uniforme ou um dos três tipos de distribuições gaussianas disponíveis, nesse trabalho também há a possibilidade de escolher ruído das distribuições, o número de classes, número de instâncias e o número de atributos.

Tabela 2 – Trabalhos que usam otimização para geração de dados sintéticos levando em consideração a complexidade dos dados.

Referência	**Abordagem**	**obj**	**Métricas Otimizadas**	**Outras Informações**
MACIÀ et al. (2008-a)	Não informado	1	Fraction of borderline points (N1)	Distribuição dos dados: Uniforme, instâncias n = 100, número de atributos = 10
MACIÀ et al. (2008-b)	GA	1	Fraction of borderline points (N1)	Distribuição dos dados: Uniforme, 101 e 501 instâncias e 2 atributos.
Macià et al. (2009)	NSGA II	3	Fraction of borderline points (N1), Ratio of average intra/inter nearest neighbor distance (N2), Ratio of the maximum Fisher's discriminant (F1)	Distribuição dos dados: Uniforme e Baseada em dados reais do repositório UCI.
Macià et al. (2010)	NSGA II	3	Volume of overlapping region (F2), Non linearity of NN classifier (N4), Fraction of maximum covering spheres (T1)	Distribuição dos dados usam dados do repositório UCI.
Melo; Lorena (2018)	Hill-climbing greedy	1	F1, N1 e Error rate of NN classifier (N3) (sendo uma métrica otimizada por vez).	Distribuição dos dados: distribuição uniforme, número de classes 2 e 5, número de exemplos 100 e 1000, número de atributos 2 e 1000.
FRANÇA et al. (2020)	NSGA III	4	Linearidade (L2), Fraction of borderline points (N1), Ratio of average intra/inter nearest neighbor distance (N2) e Balanceamento entre as classes C2.	Distribuição dos dados parametrizável podendo escolher uma distribuição uniforme ou um dos tipos de distribuição gaussiana disponíveis, valor de ruído parametrizável, há a possibilidade de escolher número de instâncias, classes e de atributos.

Fonte: O autor.

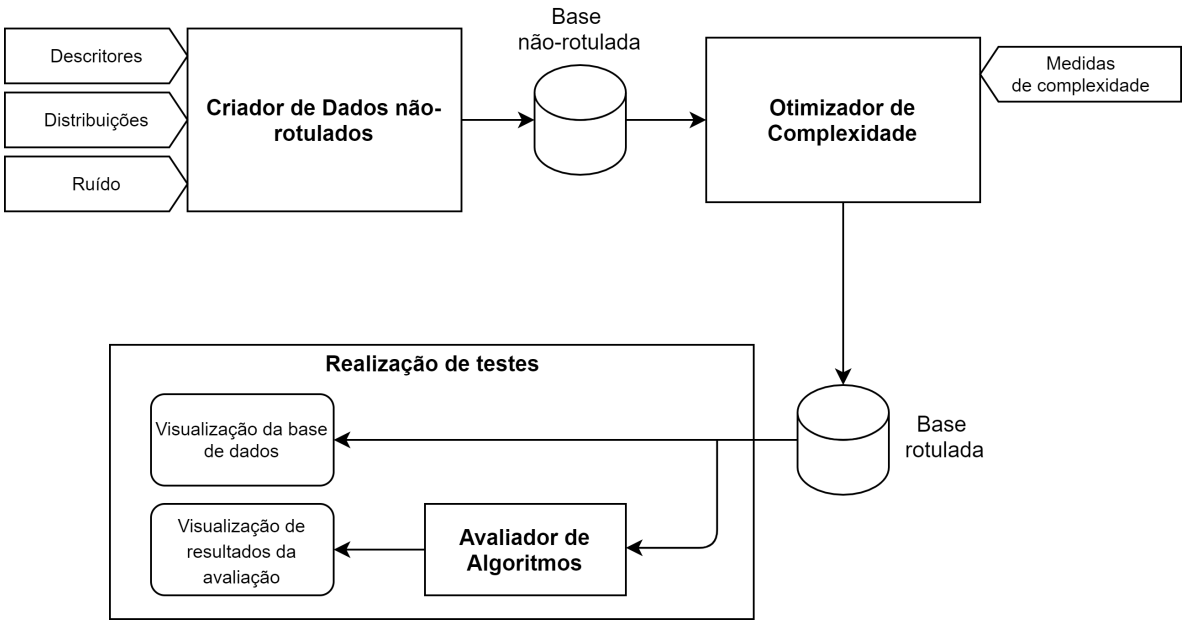
4 Proposta

Esse trabalho propõe a geração de conjunto de dados sintéticos a partir de medidas de complexidade e parametrizações determinadas pelo especialista. Desta forma, o problema de geração foi modelado como um problema de otimização de muitos objetivos (medidas de complexidade) a serem satisfeitos. A principal contribuição deste trabalho é a utilização de um algoritmo para otimizar quatro objetivos (medidas de complexidade) para a geração de dados sintéticos, e com isso, a criação de um ambiente experimental controlado para testes de classificadores.

4.1 Arquitetura

A geração de conjunto de dados sintéticos baseado em complexidade, proposto nesse trabalho segue como descrito na arquitetura da Figura 5, onde foi separado o processo em três partes destacadas a seguir.

Figura 5 – Arquitetura da geração de dados sintética baseada em complexidade.



Fonte: O autor.

4.1.1 Módulo de Criação de base não rotulada

A primeira etapa a ser realizada na arquitetura proposta é a criação de uma base de dados composta por instâncias não-rotuladas. Para a criação desta base é necessário que o usuário forneça os seguintes dados de entrada: dados descritores da base, nível de ruído e tipo de distribuição desejada.

Descritores: Aqui devem ser definidas informações que descrevem a base de dados, tais como: número de atributos, e o tipo de cada atributo; número de instâncias (ou registros) e classes.

Distribuições: Deve-se aqui definir o tipo de distribuição a ser adotada na base de dados sintéticos. As distribuições disponíveis na ferramenta são: bolhas com centroide(s), através de funções Gaussianas; luas; círculos concêntricos; ou nenhuma distribuição específica (distribuição uniforme).

Ruído: O usuário pode incluir ruído na base de dados sintéticos. O nível de ruído pode ser configurado com valores entre 0 (nenhum ruído) e 0.5 (nível alto de ruído).

4.1.2 Módulo de Otimização de Complexidade

Este módulo é o núcleo do processo de geração de dados sintéticos baseado em medidas de complexidade. Nesse módulo, são escolhidas quais medidas de complexidade se deseja otimizar e também são definidos valores alvo para cada uma delas. O módulo rotula as instâncias da base não-rotulada tentando satisfazer tais medidas de complexidade, ou seja, alcançar os valores alvo. Inicialmente, é fornecida uma base de dados sem rótulos (da etapa anterior) para um algoritmo de otimização. O algoritmo de otimização tem o objetivo de encontrar a melhor combinação de instâncias-rótulos (solução) dentre todas as possíveis combinações (espaço de soluções), que satisfaça as medidas de complexidade anteriormente definidas. Esta tarefa se trata de um problema tradicional de otimização combinatória de natureza exponencial (FRANÇA et al., 2020).

A saída deste módulo é uma base de dados rotulada com a máxima aderência encontrada considerando-se os valores alvo de complexidade definidos pelo especialista. Neste módulo, para otimização de quatro objetivos conflitantes, foi utilizado o NSGA-III como algoritmo de otimização para muitos objetivos.

4.1.3 Módulo de Testes

A base de dados rotulada na etapa anterior é fornecida como entrada para o módulo de testes. O módulo de testes dá opções de visualização gráfica da base de dados sintéticos criada, bem como dos resultados de desempenho dos classificadores na respectiva base. No momento existem quatro algoritmos classificadores pré-configurados: *Decision Tree*, *Naive Bayes*, SVM e *Random Forest*. Estes algoritmos são implementações provenientes da biblioteca *Scikit-Learn*, e seus parâmetros foram definidos com valores *default*.

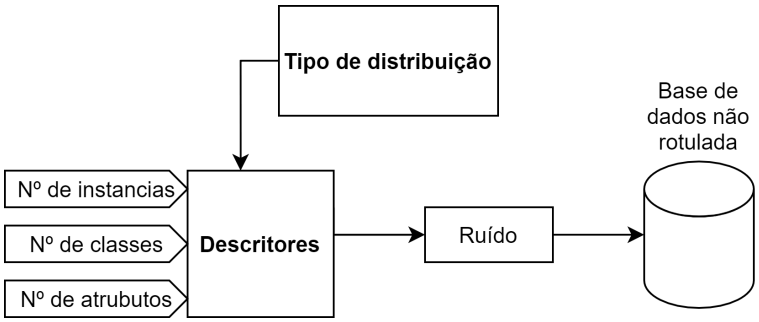
4.2 Implementação da Arquitetura

Esta seção destaca, de forma detalhada, como cada módulo do processo de geração de bases de dados sintéticos foi implementada.

4.2.1 Criação da base não-rotulada

Antes do processo de otimização, acontece a geração da base de dados sintética não rotulada que ocorre como apresentado na figura 6:

Figura 6 – Pipeline da geração da base não rotulada.



Fonte: O autor.

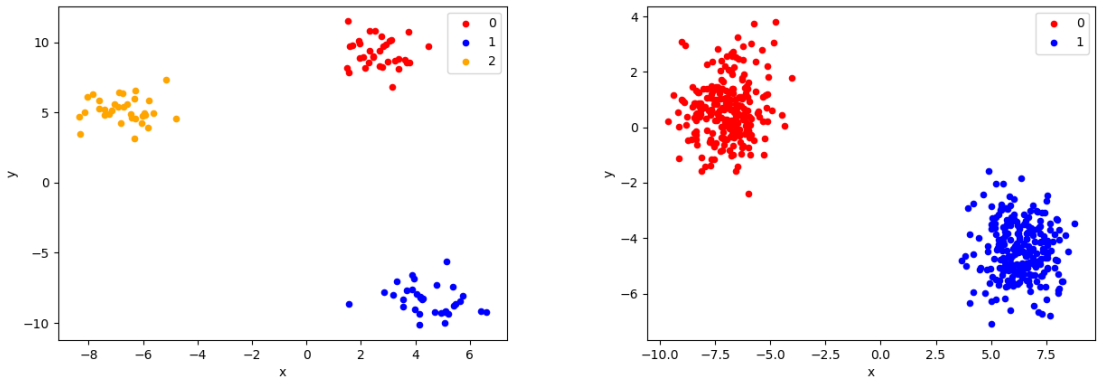
Inicialmente se escolhe o tipo de distribuição. Ela pode seguir padrões como bolhas com centroide(s), através de funções Gaussianas; luas; círculos concêntricos; ou uma distribuição uniforme. Para casos onde seguimos padrões de distribuição dos pontos, ela será feita com a ajuda da biblioteca *sklearn*[1] que traz a função *datasets* e

[1] https://scikit-learn.org/stable/

dela usamos 3 funções para distribuição dos pontos na base de dados não rotulada. São elas:

- **make_blobs**: Essa função serve para gerar bolhas de pontos com uma distribuição Gaussiana, Nela podemos controlar quantas bolhas queremos gerar, número de atributos entre outros parâmetros. Ela é adequada para gerar problemas de classificação linear dada a natureza linearmente separável das bolhas. A Figura 7 mostra dois exemplos de distribuição de pontos usando essa função. No exemplo (a) é apresentado uma distribuição de pontos usando os seguintes parâmetros: número de centros = 3, número de atributos = 2 e número de instâncias = 100. O exemplo (b) mostra a mesma distribuição com atributos diferentes, o número de centros = 2, número de atributos = 10 e número de instâncias = 500. Esses valores podem ser modificados de acordo com a necessidade de cada estudo.

Figura 7 – Distribuição dos pontos usando a função make_blobs.



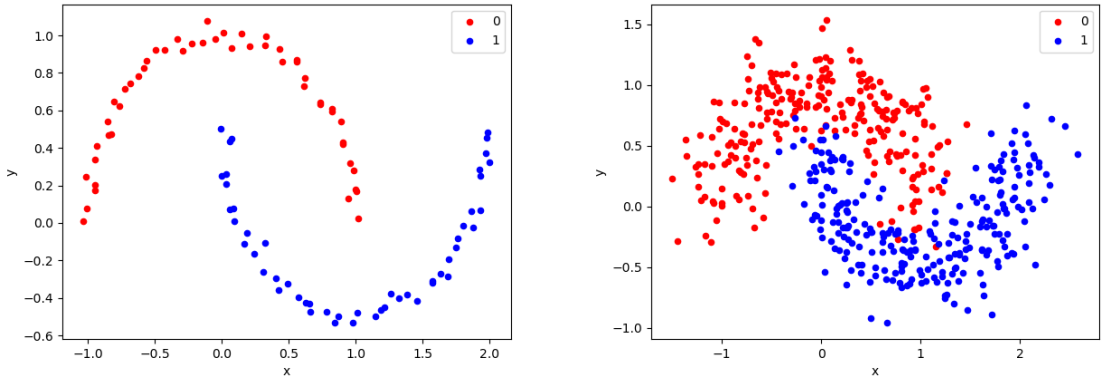
(a) 100 instâncias e 3 centros

(b) 500 instâncias e 2 centros

Fonte: O autor.

- **make_moons**: Essa função vai gerar um padrão de espiral ou duas luas. É possível controlar o ruído das formas da lua e o número de instâncias que serão geradas. Ela é adequada para algoritmos que sejam pouco sensíveis à linearidade de classes. A Figura 8 mostra dois exemplos de distribuição de pontos usando parâmetros diferentes. No exemplo (a) há como parâmetros número de instâncias = 100 e valor do ruído = 0.03. Já no exemplo (b) temos número de instâncias 500 e valor do ruído 0.2. Esses valores podem ser modificados de acordo com a necessidade de cada estudo.

Figura 8 – Distribuição dos pontos usando a função make_moons.

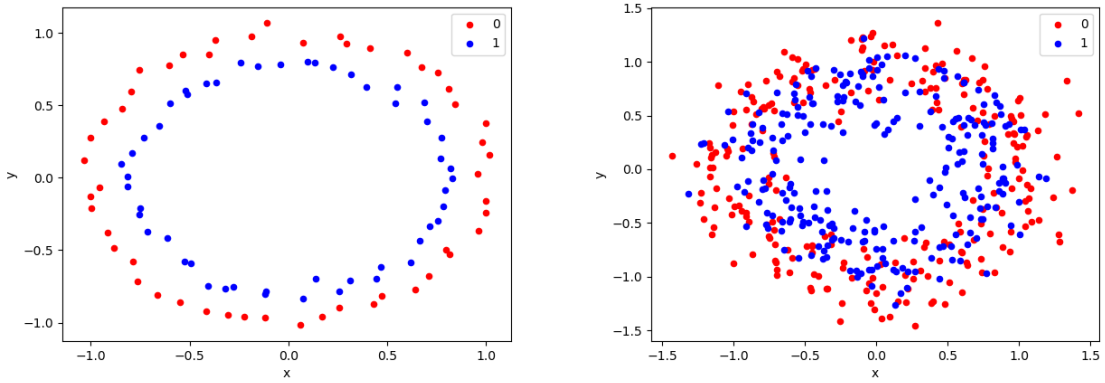


(a) 100 instâncias e 0,03 de ruído (b) 500 instâncias e 0,2 de ruído

Fonte: O autor.

- **make_circles**: Essa função vai gerar um padrão de círculos concêntricos. Aqui podemos controlar o ruído das formas dos círculos e o número de instâncias que serão geradas. Ela é adequada para algoritmos que possam aprender variedades não lineares complexas. A Figura 9 mostra um exemplo de distribuição de pontos gerados usando a função **make_circle** configurada com número de instâncias = 100 e quantidade de ruído = 0.03. Apesar da figura separar os pontos em 2 rótulos ou classes (0 e 1), esses valores serão desconsiderados na fase de otimização e novos valores serão adicionados aos pontos, valores esses que satisfaçam os critérios de complexidade definidos pelos especialistas.

Figura 9 – Distribuição dos pontos usando a função make_circles.



(a) 100 instâncias e 0,03 de ruído (b) 500 instâncias e 0,2 de ruído

Fonte: O autor.

Para a geração de dados sintéticos presente nesse trabalho, foi utilizada a distribuição de círculos com ruído de 0,2.

Logo após a escolha da distribuição dos pontos, são definidos os descritores: número de exemplos ou instâncias n, que nas execuções de teste foram definidos como 100 e 500, os valores foram escolhidos com o intuito de verificar o desempenho do algoritmo genético para bases de dados um pouco maiores. O número de atributos (*features*) m, aqui criamos problemas com 2 e 10 atributos também com o intuito de verificar o comportamento do algorítmo para problemas com poucos atributos e com uma quantidade um pouco maior deles, e o número de classes n_c que foi definido como 2 gerando assim problemas de classificação binários. Seguimos então para a escolha do ruído que pode ser um valor de 0 a 0,5 onde valores menores significam um ruído menor. Por outro lado, valores maiores geram distribuições mais ruidosas. Exemplos da aplicação do ruído podem ser vistos nas Figuras 7, 8 e 9. Após a escolha do ruído, será gerada a base de dados não rotulada e ela estará pronta para a fase de otimização.

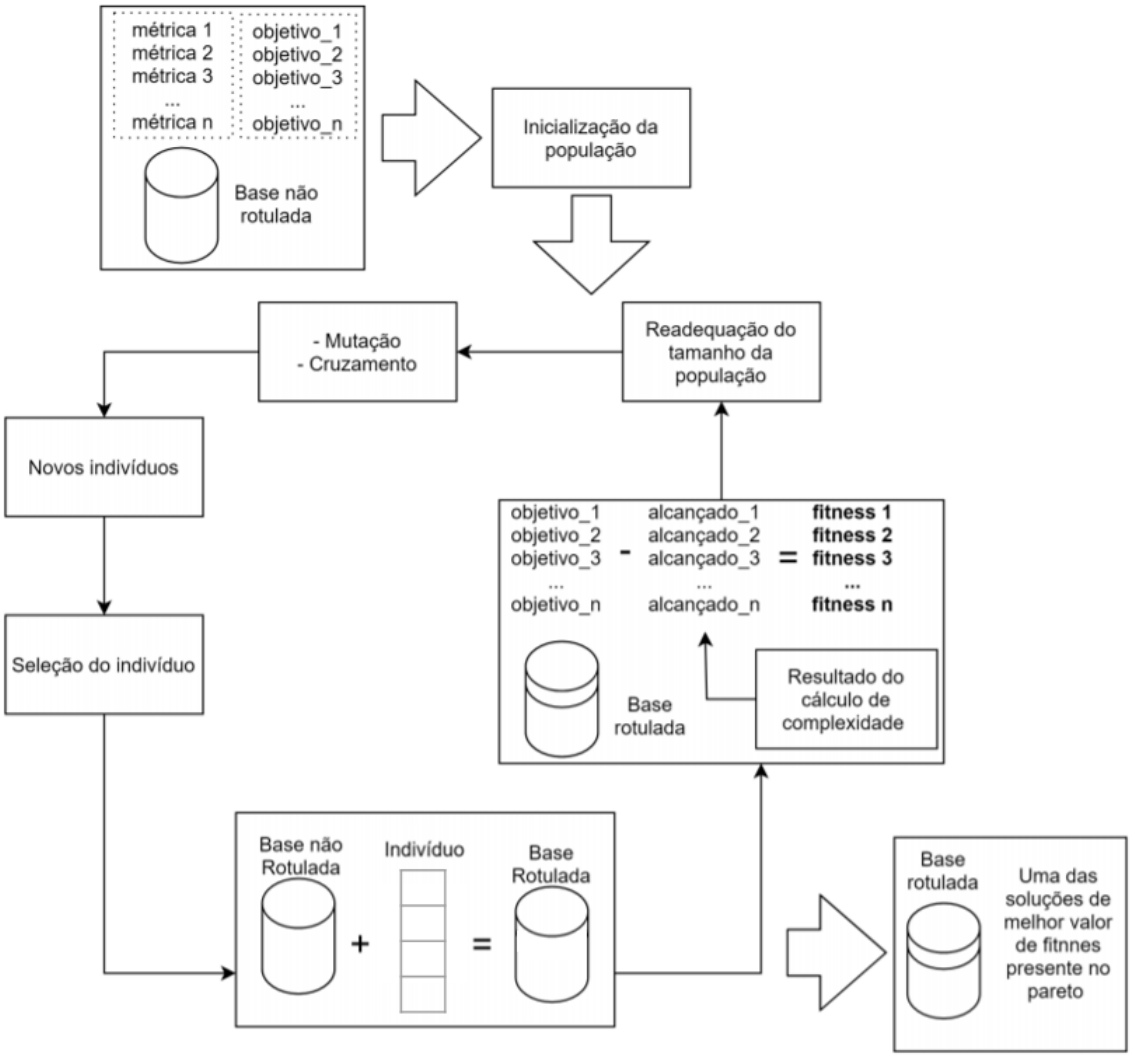
4.2.2 Módulo de Otimização

A figura 10 ilustra o processo de otimização para geração da base de dados rotulada com as características passadas pelo usuário. O processo de geração é detalhado a seguir.

Inicialmente, o módulo recebe a base não rotulada gerada no módulo anterior. Aqui também será realizada a escolha do conjunto de métricas que serão otimizadas neste módulo. Essas serão as métricas de complexidade de dados que irão ser otimizadas pelo algoritmo genético para muitos objetivos NSGA-III, já discutido na seção 2.2. Cabe ao especialista escolher para otimizar quatro métricas entre L2, N1, N2, C2, T2 e F1 e quais valores de complexidade ele deseja alcançar para cada métrica em sua base de dados sintética.

Em seguida, uma população inicial é gerada. O tamanho da população inicial foi de 30 indivíduos. Esses indivíduos são possíveis rótulos para a base de dados recebida. Eles terão como tamanho o valor de n (o número de instâncias do conjunto de dados não rotulado gerado no módulo anterior). O número de classes do problema é um valor parametrizável. Para problemas de três classes o indivíduo será um vetor de tamanho n que pode conter valores 0, 1 ou 2 em cada uma de suas posições. Essas classes estarão

Figura 10 – Processo de otimização.



Fonte: O autor.

presentes em cada posição do vetor onde em um indivíduo $\vec{I}$ cada posição assume uma classe como mostrado na figura 11.

Figura 11 – Representação do indivíduo para problemas de três classes.

0	0	1	0	2	2	2	1	0	0	1	1	1	0	2	2	0	1	0	1

Fonte: O autor.

Seja D um conjunto de dados com n instâncias. Cada instância n_i em D tem um rótulo $y_i \in 1...,n_c$, em que n_c é o número de classes. Neste trabalho os rótulos das classes são atribuídos pelo algoritmo genético de muitos objetivos. Desta forma, um indivíduo $\vec{I}$ é

representado por um vetor de comprimento n, em que cada posição assume um valor de classe y_i:

$$\vec{I} = (y_i, y_2, ..., y_n). \tag{4.1}$$

Portanto, cada indivíduo representa uma possível combinação de rótulos de classe para o conjunto de dados. Se tivermos D com $n = 100$ instâncias e $n_c = 3$ rótulos de classe possíveis, o espaço de pesquisa resultante terá 3^{100} soluções possíveis.

Cada indivíduo $\vec{I}$ está associado a um vetor de valores de *fitness* derivados das funções objetivo em $\vec{f}$. No contexto da geração de dados sintéticos, $\vec{f}$ é o conjunto escolhido de medidas de complexidade. A escolha das métricas a serem consideradas no processo de otimização depende da demanda do especialista em termos de complexidade. Neste trabalho, especificamente, quatro medidas de complexidade foram escolhidas para execução e testes: uma medida de linearidade, duas medidas baseadas em vizinhança e uma medida de equilíbrio de classe. Nesse caso essas medidas formam o vetor $\vec{f}$ apresentado na imagem como retorno.

O próximo passo é a geração de novos indivíduos através da mutação, usando o método *shufle indexes*, e cruzamento onde foi utilizado o método *two points*, de indivíduos selecionados com base nos seus valores de *fitness* através de torneio, o algoritmo foi configurado com o valor de probabilidade de reprodução de 0.6, taxa de mutação de 0.2 e taxa de cruzamento de 0.9, como descrito na Tabela 3.

Cada uma das medidas de complexidade produz um valor no intervalo [0, 1]. Quanto mais simples o problema, mais próximo de 0 ele é, caso contrário, mais próximo de 1. Por exemplo, se a medida L2 = 0, significa que os dados são totalmente linearmente separáveis, ou seja, um simples problema de classificação. Por outro lado, se L2 = 1, significa que os dados não são linearmente separáveis ao extremo.

Para calcular o valor de *fitness* de uma solução candidata (combinação de rótulos), cada uma das medidas de complexidade é calculada, levando em consideração o banco de dados de soluções rotuladas "base não rotulada"+ "indivíduo"que também podemos chamar de "distribuição dos pontos"+ "combinação de rótulos".

As funções de *fitness* definidas aqui calculam a distância entre o valor obtido por cada medida de complexidade, e o valor de complexidade desejado pelo especialista. Assim,

existem seis caminhos dentro da função objetivo (f_{L2}, f_{N1}, f_{N2}, $f_{ClsCoef}$, f_{T2} e f_{C2}) para minimizar a distância entre o valor calculado e o valor esperado. Os possíveis caminhos disponíveis dentro da função de *fitness* são definidos nas seguintes equações:

$$f_{L2} = |objetivo_L2 - L2|, \tag{4.2}$$

$$f_{N1} = |objetivo_N1 - N1|, \tag{4.3}$$

$$f_{N2} = |objetivo_N2 - N2|, \tag{4.4}$$

$$f_{C2} = |objetivo_C2 - C2|, \tag{4.5}$$

$$f_{T2} = |objetivo_T2 - T2|e \tag{4.6}$$

$$f_{F1} = |objetivo_F1 - F1|, \tag{4.7}$$

onde, $objetivo_L2$, $objetivo_N1$, $objetivo_N2$, $objetivo_C2$, $objetivo_T2$ e $objetivo_F1$ são valores de referência definidos pelo especialista.

As funções de *fitness* acima citadas serão utilizadas pelo algoritmo de acordo com as métricas de complexidade escolhidas pelo especialista.

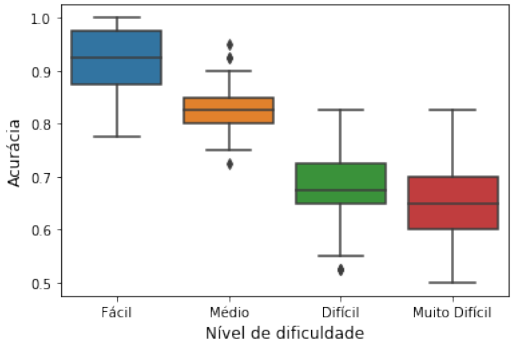
Após o cálculo da complexidade de cada conjunto de base não rotulada + indivíduo, ele recebe um valor de *fitness* o qual será posicionado dentro da população. Depois de todo o processo, os indivíduos que ultrapassam o tamanho da população serão descartados e inicia-se novamente o processo em uma nova geração. Foi estipulado um valor máximo de 5000 gerações. Após a execução de todas as gerações ou o valor ideal ser encontrado, temos como resultado um conjunto de rótulos onde o melhor. Em termos de *fitness*, será escolhido para rotular a distribuição de pontos resultando assim em uma base de dados rotulada e com características de complexidades conhecidas.

4.2.3 Módulo de Testes

O módulo de testes tem como objetivo analisar o comportamento de algoritmos de classificação usando os conjuntos de dados sintéticos gerados. Esses dados estão sendo divididos em base de treino e base de teste e utilizamos a técnica de *Cross-validation* para verificar o comportamento dos classificadores. São gerados diagramas de caixa (*boxplots*) com os valores obtidos como resultado das classificações. Com relação aos resultados, analisar uma só métrica de avaliação (acurácia por exemplo) pode nos dar uma leitura enganosa, por isso, para extrair informações dessas execuções, utilizamos quatro métricas de avaliação: Acurácia, Medida-F, Precisão e Revocação.

Foram utilizados os algoritmos de classificação floresta aleatória, árvore de decisão, *naive bayes* e SVM e os dados gerados com níveis de dificuldade FÁCIL, MÉDIO, DIFÍCIL e MUITO DIFÍCIL. Cada algoritmo é executado 100 vezes e todos os resultados são apresentados em *box plots* como apresentado na Figura 12, onde cada *box plot* representa o desempenho médio do classificador dataset gerado. Mais detalhes experimentais são fornecidos na seção 5.3.

Figura 12 – Exemplo de figura resultante do módulo de testes.



Fonte: (O autor).

5 Metodologia

Este capítulo apresenta as ferramentas e técnicas utilizadas no desenvolvimento do trabalho e a metodologia experimental adotada para avaliar a proposta.

5.1 Ambiente Experimental

5.1.1 Linguagem de Desenvolvimento

Python™[1] foi a linguagem principal adotada para o desenvolvimento. Trata-se de uma linguagem de programação funcional de código aberto de alto nível utilizada pelo software de mesmo nome. A linguagem conta com um grande número de bibliotecas maduras e que facilitam cálculos matemáticos, funções estatísticas, geração de gráficos entre outras funções.

Para cálculos das medidas de complexidade, utilizamos uma biblioteca da linguagem de programação R[2] chamada ECoL[3]. Essa biblioteca traz consigo cálculos robustos de complexidade e a mesma é bastante utilizada em projetos que fazem uso dessas métricas.

5.1.2 Bibliotecas Utilizadas

Foi utilizada a biblioteca de computação evolutiva DEAP[4] para o uso dos algoritmos evolucionários de muitos objetivos e a biblioteca *Scikit Learn*[5] para o uso de classificadores e funções de aprendizagem de máquina. Essas bibliotecas estão disponíveis na linguagem de programação Python™ e podem ser encontradas no repositório PyPI[6]

5.1.3 Ambiente de Desenvolvimento

Para os experimentos aqui descritos, foi utilizado um computador com o Processador Intel(R) Core(TM) i7-8750H com *clock* de 2.20GHz, 4M de cache e 6 Núcleo(s), 12

[1] https://wiki.python.org.br/DocumentacaoPython
[2] https://www.r-project.org/
[3] https://github.com/lpfgarcia/ECoL
[4] https://deap.readthedocs.io/en/master/
[5] https://scikit-learn.org/stable/
[6] https://pypi.org/

Processadores Lógicos, 16 GB de memória RAM DDR4 2666 MHz, placa gráfica Nvidia GTX 1050Ti. Como sistema operativo, foi utilizada a distribuição GNU/Linux MINT[7].

5.2 Execução

Analisando trabalhos anteriores, decidimos usar o NSGA-III como algoritmo para muitos objetivos. As execuções foram realizadas usando os parâmetros listados na Tabela 3. Várias configurações foram executadas como teste para encontrar a melhor delas. Dentre todas as combinações, configurações e parâmetros testados, esse é o que nos trouxe resultados mais promissores em um tempo mais curto.

Tabela 3 – Configuração do algoritmo e valores iniciais do conjunto de dados sintéticos.

Parâmetro	**Valor**
Quantidade de exemplos n	100; 500
Número de atributos m	2; 10
Número de classes	2
Método de inicialização	Uniforme
Tamanho da população	30
Gerações	5000
Método de seleção	Torneio
Taxa de reprodução	0.6
Taxa de mutação	0.2
Taxa de cruzamento	0.9
Método de cruzamento	Two-point
Método de mutação	Shuffle Indexes

Fonte: O autor.

[7] https://linuxmint.com/

5.3 Metodologia de Avaliação

O experimento foi dividido em duas partes: a visualização dos dados e a análise quantitativa. Consideramos conjuntos de dados de 2 atributos com 100 e 500 amostras. Logo depois o algoritmo de otimização foi utilizado para gerar conjuntos de dados sintéticos em vários cenários (de quatro objetivos) como descrito na Tabela 4.

Tabela 4 – Valores de complexidade para cada nível de problema.

Nível do problema	**L2**	**N1**	**N2**	**C2**
Fácil	0.07	0.07	0.07	0.07
Médio	0.07	0.07	0.07	0.48
Difícil	0.22	0.22	0.22	0.22
Muito difícil	0.35	0.35	0.35	0.48

O cenário **Fácil** tem como característica um conjunto de dados balanceado, linearmente separável e com vizinhos próximos sendo da mesma classe. Já o cenário **Médio** tem caraterísticas parecidas com o fácil, porém, com as classes desbalanceadas. O cenário **Difícil** nos traz um conjunto de dados com um alto grau de vizinhos mais próximos de classes diferentes. Nesse nível de problema não temos a base balanceada nem linearmente separável. O problema **Muito difícil** nos apresenta um cenário caótico onde as classes são extremamente desbalanceadas, não linearmente separável e a maioria dos vizinhos mais próximos são de classes diferentes. O número de instâncias sobrepostas de classes diferentes também e alto nesse problema.

O objetivo dessa parte do experimento é mostrar graficamente através de *scatter plots* os conjuntos de dados rotulados com uma solução promissora obtida pelo algoritmo de otimização em cada um dos níveis de dificuldade do problema. Como a saída do NSGA-III é um Pareto composto por um conjunto de soluções particulares, utilizamos um algoritmo de contagem de borda (EMERSON, 2013), (BRAMS; FISHBURN, 2002) para selecionar uma solução do Pareto. Este é um método de classificação de vencedor onde são classificados por cada critério e, em seguida, uma classificação média é retornada e daí retiramos o vencedor. Usamos essa solução vencedora para a visualização dos dados.

A segunda parte do experimento é uma análise quantitativa onde são utilizados conjuntos de dados com 100 instâncias considerando 2 e 10 atributos e verificamos o

desempenho do otimizador nesses cenários. Nesse experimento verificamos os valores de *fitness*, o tempo de execução e o número de interações usados pelo otimizador para alcançar os valores desejados em cada cenário.

Como complemento, realizamos a geração de dados sintéticos de forma aleatória apenas como meio de comparação e avaliação da abordagem. Os valores encontrados pela abordagem evolucionária serão comparados com os melhores valores encontrados pela geração aleatória. Além disso, foram também executamos classificadores nos diferentes conjuntos de dados gerados para verificar a performance com relação às métricas de acurácia, medida-F e precisão. Nessa análise verificamos a dificuldade que os algoritmos enfrentam nos diferentes cenários.

6 Resultados

Nesse capítulo, discutiremos os resultados obtidos nos experimentos realizados na pesquisa.

6.1 Análise da Otimização

Aqui serão apresentados e discutidos valor alcançado pelo algoritmo otimização de muitos objetivos na geração de dados sintéticos.

6.1.1 Conjunto de dados com 100 instâncias

A Tabela 5 mostra os resultados de otimização para 2 e 10 atributos em um conjunto de dados com 100 instâncias.

Tabela 5 – Valores de L2, N1, N2 e C2 dos conjuntos de dados gerados com 100 instâncias considerando 2 e 10 atributos.

Problema	**n**	**m**	**Target (L2, N1, N2, C2)**	**Resultado (L2, N1, N2, C2)**	**Tempo (s)**
Fácil	100	2	(0.07, 0.07, 0.07, 0.07)	(0.07, 0.07, 0.28, 0.07)	2.460
		10	(0.07, 0.07, 0.07, 0.07)	(0.09, 0.07, 0.28, 0.07)	2.475
Médio	100	2	(0.07, 0.07, 0.07, 0.48)	(0.11, 0.07, 0.27, 0.48)	1.470
		10	(0.07, 0.07, 0.07, 0.48)	(0.11, 0.08, 0.27, 0.50)	1.488
Difícil	100	2	(0.22, 0.22, 0.22, 0.22)	(0.22, 0.22, 0.23, 0.22)	1.950
		10	(0.22, 0.22, 0.22, 0.22)	(0.22, 0.22, 0.25, 0.22)	2.010
Muito difícil	100	2	(0.35, 0.35, 0.35, 0.48)	(0.17, 0.35, 0.27, 0.48)	2.190
		10	(0.35, 0.35, 0.35, 0.48)	(0.18, 0.35, 0.27, 0.43)	2.215

Fonte: O autor

- **Fácil**

A tabela 5 mostra os resultados encontrados pelo algoritmo para os diferentes níveis de complexidade, o problema fácil tem como valor alvo 0,07 para todas as medidas de

complexidade. Foram realizados testes com bases de dados de 2 e 10 atributos, para as bases de 2 atributos ($m = 2$) o algoritmo foi capaz de encontrar os valores de 0,07 para L2, N1 e C2. Aqui é interessante observar o valor obtido pela métrica N2. O problema fácil tem como característica ser linearmente separável e essa métrica é conflitante com métricas de vizinhança, principalmente a N2 que calcula a diferença entre vizinhos próximos de classes iguais e de classes diferentes.

Observando os resultados para ($m = 10$), pela maior quantidade de atributos a dificuldade em tornar a base linearmente separável aumenta já que agora estamos levando em consideração vários atributos, ainda assim o algoritmo foi capaz de encontrar valores muito próximos ao alvo.

- **Médio**

Para o problema médio o algoritmo tem como objetivo encontrar uma base de dados linearmente separável, com vizinhos próximos pertencentes a mesma classe, o ponto que deixará essa base mais desafiadora que a base de dados fácil é o desbalanceamento. As classes estarão extremamente desbalanceadas, para L2, N1 e N2 o valor de meta será de 0,07. Já para a métrica C2 o valor alvo será de 0,48.

Os valores alcançados para N2 confirmam a dificuldade em gerar dados que sejam ao mesmo tempo linearmente separáveis e com vizinhos próximos de classes diferentes. O valor encontrado para N2 foi 0,28 e o valor de L2 "linearidade"foi 0,11 onde o valor alvo seria 0,07. Os valores encontrados para as outras métricas foram muito próximos aos esperados.

- **Difícil**

Para o problema difícil o desafio foi gerar uma base com um valor alvo mediano para cada uma das medidas. Os valores alvo escolhidos foram 0,22 para cada. O algoritmo se mostrou muito eficiente ao alcançar essas métricas com exceção da N2 onde o algoritmo foi capaz de encontrar 0,23 para as bases onde $m = 2$ e 0,25 para os casos onde $m = 10$.

- **Muito Difícil**

O objetivo aqui é gerar uma base caótica, com vizinhos próximos de classes diferentes não linearmente separável e desbalanceada. Para isso, foram colocados valores altos para os alvos de cada medida de complexidade. Os valores encontrados para essa base quando $m = 2$ foram $L2 = 0.17, N1 = 0.35, N2 = 0.27$ e $C2 = 0.48$. Para a base de dados onde $m = 10$ foram encontrados os valores $L2 = 0.18$, $N1 = 0.35$, $N2 = 0.27$ e $C2 = 0.43$.

6.1.2 Conjunto de dados com 500 instâncias

Um outro teste realizado, foi a geração de conjuntos de dados com 2 e 10 atributos e 500 instâncias como exibido na figura 14. O intuito foi verificar se há impacto na otimização das medidas de complexidade quando o número de instâncias aumenta. A Tabela 6 mostra os resultados da otimização para 2 e 10 atributos e 500 instâncias.

Tabela 6 – Valores de L2, N1, N2 e C2 dos conjuntos de dados gerados com 500 instâncias considerando 2 e 10 atributos.

Problema	**N**	**M**	**Target (L2, N1, N2, C2)**	**Resultado (L2, N1, N2, C2)**	**Tempo (s)**
Fácil	500	2	(0.07, 0.07, 0.07, 0.07)	(0.09, 0.15, 0.26, 0.07)	19.188
		10	(0.07, 0.07, 0.07, 0.07)	(0.12, 0.24, 0.25, 0.07)	19.370
Médio	500	2	(0.07, 0.07, 0.07, 0.48)	(0.09, 0.09, 0.26, 0.59)	11.466
		10	(0.07, 0.07, 0.07, 0.48)	(0.10, 0.11, 0.26, 0.48)	11.895
Difícil	500	2	(0.22, 0.22, 0.22, 0.22)	(0.22, 0.22, 0.23, 0.22)	15.210
		10	(0.22, 0.22, 0.22, 0.22)	(0.22, 0.22, 0.24, 0.22)	15.372
Muito difícil	500	2	(0.35, 0.35, 0.35, 0.48)	(0.16, 0.35, 0.27, 0.48)	17.082
		10	(0.35, 0.35, 0.35, 0.48)	(0.19, 0.38, 0.27, 0.38)	17.513

Fonte: O autor.

- **Fácil**

Como no teste anterior, o valor alvo para o problema fácil é de 0.07 para todas as medidas de complexidade, o algoritmo sentiu o maior volume de instancias e teve mais dificuldade para chegar a valores baixos para cada métrica, ainda assim foi possível encontrar bons valores para o conjunto de dados fácil. Como mostrado na tabela 6, os valores encontrados para essa base quando $m = 2$ foram $L2 = 0.09, N1 = 0.24, N2 = 0.25$ e $C2 = 0.07$. Para a base de dados onde $m = 10$ foram encontrados os valores $L2 = 0.12$, $N1 = 0.24$, $N2 = 0.26$ e $C2 = 0.07$.

A distribuição dos pontos e o valor alto de m (maior quantidade de instâncias) deu origem a uma base de dados com muito pontos sobrepostos e muito volume de pontos extremamente próximos uns dos outros, o que dificultou a geração de bases linearmente

separáveis (L2) e com vizinhos próximos de classes diferentes (N2) ao mesmo tempo.

A base de dados fácil é linearmente separável e balanceada, tem um valor baixo de vizinhos próximos de classes diferentes.

- **Médio**

Aqui no problema médio o algoritmo tem como objetivo encontrar uma base de dados linearmente separável e com vizinhos próximos pertencentes a mesma classe, o ponto que deixará essa base mais desafiadora que a base de dados fácil é o desbalanceamento. As classes estarão extremamente desbalanceadas, para L2, N1 e N2 o valor de meta será de 0,07. Já para a métrica C2 o valor alvo será de 0,48.

Os valores alcançados para N2 confirmam a dificuldade em gerar dados que sejam ao mesmo tempo linearmente separáveis e com vizinhos próximos de classes diferentes. O algoritmo foi capaz de gerar a base de dados de dificuldade média, a base de dados gerada é linearmente separável, os pontos próximos são de classes iguais e como desafio, temos a base desbalanceada com valor de $C2 = 59$ para $m = 2$ e $C2 = 48$ para $m = 10$.

- **Difícil**

Para o problema difícil, assim como nas bases onde $n = 100$ o desafio foi gerar uma base com um valor alvo mediano para cada uma das medidas. Os valores alvo escolhidos foram 0,22 para cada. O algoritmo se mostrou muito eficiente ao alcançar essas métricas achando valores exatamente iguais aos valores alvo, com exceção da N2 onde o algoritmo foi capaz de encontrar 0,23 para as bases onde $m = 2$ e 0,24 para os casos onde $m = 10$.

- **Muito Difícil**

O objetivo aqui é gerar uma base caótica, com vizinhos próximos de classes diferentes não linearmente separável e desbalanceada. Para isso, foram colocados valores altos para os alvos de cada medida de complexidade. Os valores encontrados para essa base quando $m = 2$ foram $L2 = 0.16, N1 = 0.35, N2 = 0.27$ e $C2 = 0.48$. Para a base de dados onde $m = 10$ foram encontrados os valores $L2 = 0.19$, $N1 = 0.38$, $N2 = 0.27$ e $C2 = 0.38$.

Em todos os testes, as bases onde $m = 10$ trouxeram maior desafio para o algoritmo quando o objetivo era o balanceamento, ainda assim ele foi capaz de encontrar bons valores, em alguns casos, valores tão bons quanto nas bases onde $m = 2$.

6.2 Proposta X Abordagem de geração aleatória

Com o intuito de verificar a efetividade do algoritmo genético para encontrar valores de complexidade especificados pelo usuário, realizamos o experimento inicializando uma população aleatória a cada geração, não fazendo mutação, cruzamento ou realizando nenhuma outra operação. Novos indivíduos são gerados aleatoriamente e nos casos onde eles sejam melhores que os já presentes na população, os mesmos serão mantidos para a próxima geração. No final da execução o melhor resultado será retornado. A Tabela 7 mostra a diferença dos resultados encontrados usando algoritmos genéticos e a abordagem aleatória.

Tabela 7 – Comparação da geração de base de dados usando o a abordagem evolucionária e uma abordagem aleatória.

Problema	**Valores Alvo (L2, N1, N2, C2)**	**Abordagem evolucionária (L2, N1, N2, C2)**	**Abordagem aleatória (L2, N1, N2, C2)**
Fácil	(0.07, 0.07, 0.07, 0.07)	(0.07, 0.07, 0.28, 0.07)	(0.06, 0.24, 0.64, 0.29)
Médio	(0.07, 0.07, 0.07, 0,48)	(0.11, 0.07, 0.27, 0.48)	(0.09, 0.23, 0.65, 0.29)
Difícil	(0.22, 0.22, 0.22, 0.22)	(0.22, 0.22, 0.23, 0.22)	(0.09, 0.23, 0.65, 0.28)
Muito Difícil	(0.35, 0.35, 0.35, 0.48)	(0.17, 0.35, 0.27, 0.48)	(0.09, 0.23, 0.69, 0.31)

Fonte: O autor

- **Fácil**

Para o problema fácil, como já visto nos testes anteriores, o objetivo é gerar bases de dados linearmente separáveis, com vizinhos próximos de classes iguais, e com número de classes balanceado. Para o problema fácil, o valor alvo para todas as métricas de complexidade foi 0.07 para cada uma das métricas, o algoritmo foi capaz de gerar a base de dados com os valores desejados, exceto para o N2 onde o valor alvo seria 0.07 e o valor encontrado foi de 0.28.

Na abordagem aleatória a base com valores mais próximos dos valores alvo teve como valores: $L2 = 0.06$, $N1 = 0.24$, $N2 = 0.64$ e $C2 = 0.29$.

- **Médio**

Para o problema médio, os valores alvos para cada uma das métricas são 0.07,

exceto para o balanceamento C2 onde o valor alvo é 0.48. A base com valores alvos mais próximos gerada pela abordagem aleatória teve como valores: $L2 = 0.09$, $N1 = 0.23$, $N2 = 0.65$ e $C2 = 0.29$.

- **Difícil**

Para o problema difícil, os valores alvos para cada uma das métricas é de 0.22. A base com valores alvos mais próximos gerada pela abordagem aleatória teve como valores: $L2 = 0.09$, $N1 = 0.23$, $N2 = 0.65$ e $C2 = 0.28$.

- **Muito Difícil**

Para o problema muito difícil, o objetivo era gerar bases de dados caóticas e os valores alvos para cada uma das métricas é de 0.35 exceto para o balanceamento onde o valor alvo foi 0.48. A base com valores alvos mais próximos gerada pela abordagem aleatória teve como valores: $L2 = 0.09$, $N1 = 0.23$, $N2 = 0.69$ e $C2 = 0.31$.

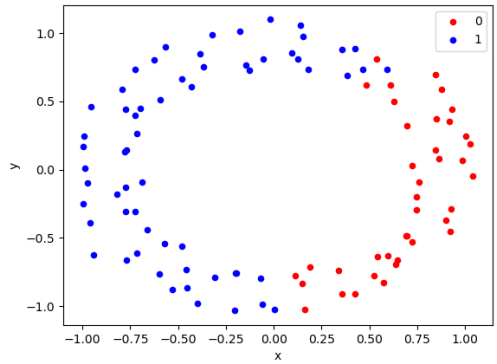
6.3 Visualização dos dados

O módulo de visualização dos dados mostra, de forma gráfica, como as diferentes classes estão distribuídas. Cada nível de complexidade tem características bem marcantes que podem ser observadas nas Figuras 13, 14, 15 e 16.

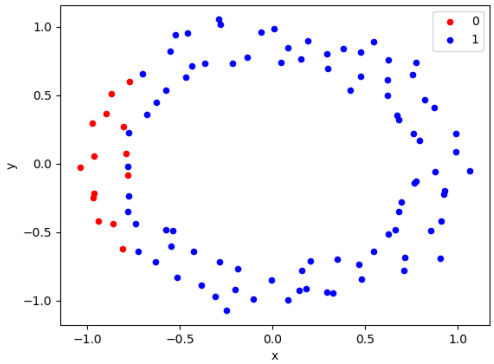
6.3.1 Abordagem evolucionária

A Figura 13 mostra a posição de cada instância do conjunto de dados sintéticos gerado com seus respectivos rótulos, otimizados pelo NSGA-III. Os eixos x e y são os valores dos atributos do conjunto de dados. As Figuras 13-a, -b, -c e -d representam uma das soluções pertencentes ao Pareto resultante, encontrado pelo NSGA-III, para cada um dos níveis de problema (ver Tabela 4). Este experimento considera a geração de conjuntos de dados com 100 instâncias, 2 atributos e ruído 0,1.

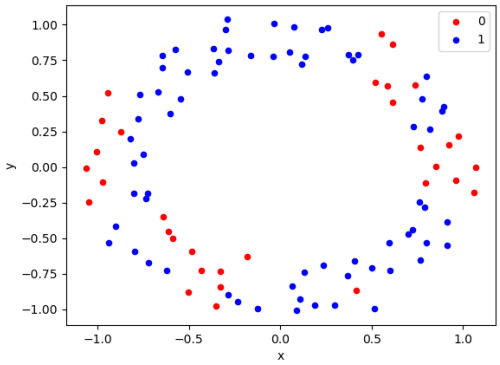
A Figura 13-a mostra o conjunto de dados com seus rótulos otimizados para atender aos objetivos de um conjunto de dados considerado fácil. O problema simples é extremo, onde todos os objetivos precisam ser otimizados para os menores valores de complexidade. Como as medidas de complexidade L2, N1 e N2 são conflitantes, minimizá-las ao mesmo tempo é uma tarefa difícil. Os valores alcançados para cada um dos objetivos foram, L2 =



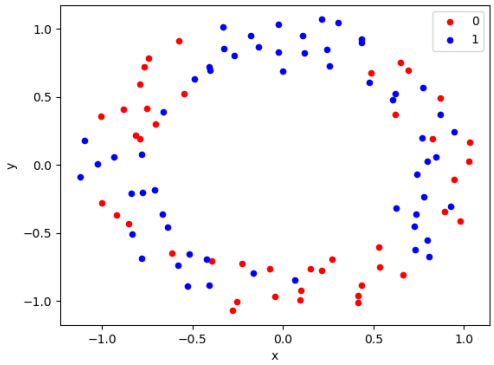
(a) Problema fácil com 100 instâncias



(b) Problema médio com 100 instâncias



(c) Problema difícil com 100 instâncias



(d) Problema muito difícil com 100 instâncias

Figura 13 – Posição de cada instância em bases de dados geradas com diferentes níveis de complexidade.

0.07, N1 = 0.07, N2 = 0,28 e C2 = 0,07. O algoritmo atingiu um valor próximo ao ótimo em termos de C2, tornando a base balanceada. Embora o conjunto de dados gerado tenha um valor baixo de L2, ele não tem um comportamento separável completamente linear. Isso se deve ao equilíbrio das classes, dificultando a tarefa de redução da sobreposição de N1 e principalmente N2.

A Figura 13-b mostra o conjunto de dados com seus rótulos otimizados para atender aos objetivos do problema médio como exibido na tabela 4. Neste caso, tenta-se otimizar o valor de C2 = 0.48, aumentando o desequilíbrio de base. Os valores alcançados para cada objetivo do problema médio foram, L2 = 0.11, N1 = 0.07, N2 = 0.27 e C2 = 0.48.

O algoritmo de otimização foi capaz de otimizar o valor de C2 ao valor desejado, desequilibrando a base. O desequilíbrio favorece a redução da sobreposição, pois há menos instâncias de uma determinada classe. Por esse motivo, os valores alcançados para cada um dos outros objetivos foram menores quando comparados ao problema simples, ainda

assim o fato da base gerada ter um comportamento linear separável traz dificuldade em alcançar valores baixos de sobreposição, principalmente da métrica N2.

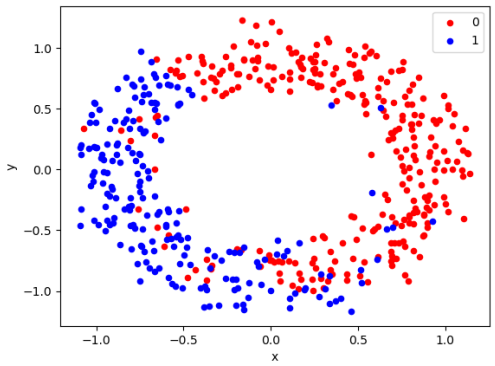
A Figura 13-c mostra o conjunto de dados com seus rótulos otimizados para atender aos objetivos do problema difícil. Neste caso, tenta-se aumentar os valores de medição, aumentando a complexidade do conjunto de dados como exibido na tabela 4. Os valores alcançados pelo algoritmo para cada objetivo do problema difícil foram, L2 = 0.22, N1 = 0.22, N2 = 0.32 e C2 = 0.22.

A Figura 13-d mostra o conjunto de dados com seus rótulos otimizados para atender aos objetivos do problema muito difícil. Como o problema simples, o problema muito difícil é extremo, tendo em vista que todos os objetivos precisam ser otimizados para os mais altos valores de complexidade. Como as medidas de complexidade L2 e N1 e N2 são conflitantes, otimizar seus valores ao mesmo tempo não é trivial. Nesse problema, o objetivo é encontrar uma solução que se aproxime de L2 = 0,35, N1 = 0,35, N 2 = 0,35 e C2 = 0,48. Porém, devido à natureza conflitante dos objetivos, os valores alcançados para cada um deles foram, L2 = 0,20, N1 = 0,35, N2 = 0,27 e C2 = 0,35. O resultado alcançado mostra um conjunto de dados com comportamento caótico, com alto índice de desequilíbrio e sobreposição.

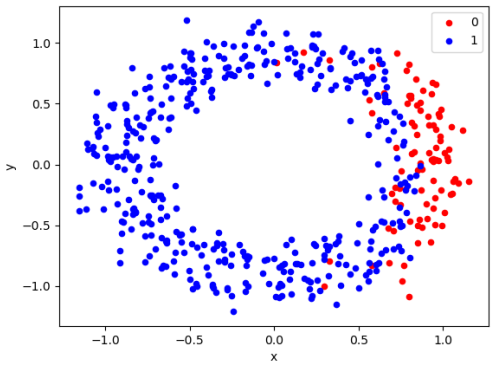
O algoritmo de otimização também foi usado para gerar conjuntos de dados com 10 atributos. O objetivo é verificar se há impacto na otimização das medidas de complexidade quando o número de atributos aumenta.

O mesmo experimento realizado na Figura 13 para conjuntos de dados com 100 instâncias, foi utilizado para o conjunto de dados com 2 atributos e 500 instâncias aqui na figura 14, nesse conjunto de dados com mais instâncias, é possível visualizar melhor o fenômeno da linearidade como pode ser visto nas figuras 14-a e -b.

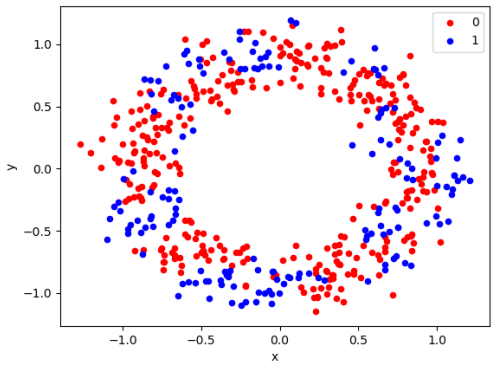
Figura 14 – Posição de cada instância em bases de dados geradas com diferentes níveis de complexidade, disposição dos dados em círculos com ruído em 0,1.



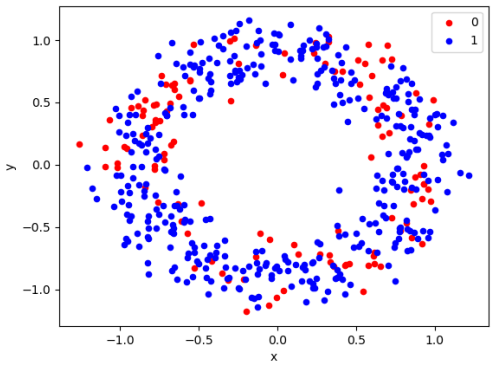
(a) Problema fácil com 500 instâncias



(b) Problema médio com 500 instâncias



(c) Problema difícil com 500 instâncias

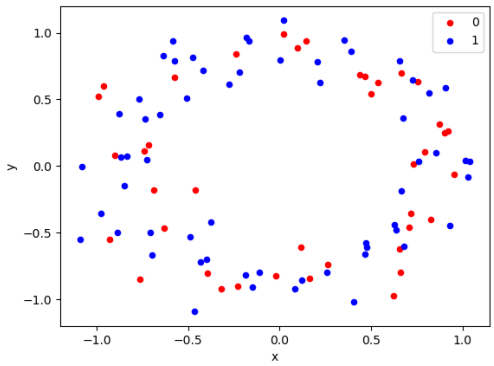


(d) Problema muito difícil com 500 instâncias

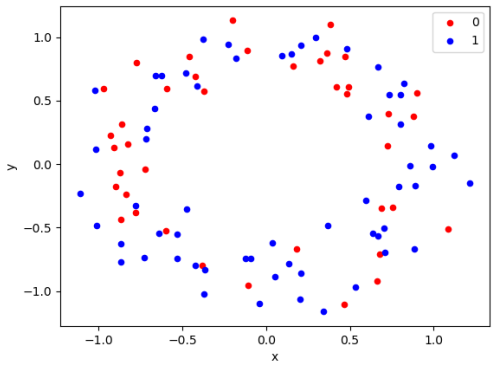
Fonte: O autor.

6.3.2 Abordagem aleatória

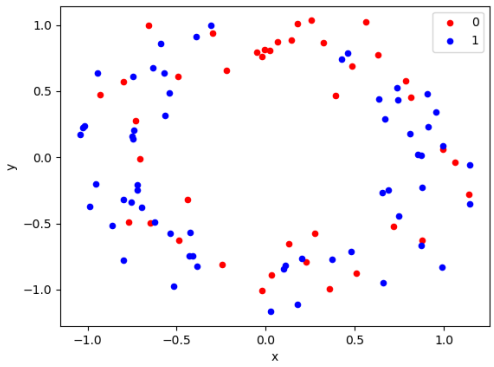
A Figura 15 mostra a posição de cada instância do conjunto de dados sintéticos gerado com seus respectivos rótulos, usando a abordagem aleatória. Os eixos x e y são os valores dos atributos do conjunto de dados. As Figuras 15-a, -b, -c e -d representam uma das soluções pertencentes ao Pareto resultante, da geração aleatória de indivíduos que teve como objetivo demonstrar a diferença de desempenho dessa abordagem comparada a evolucionária (ver Tabela 7). Este experimento considera a geração de conjuntos de dados com 100 instâncias, 2 atributos e ruído 0,1.



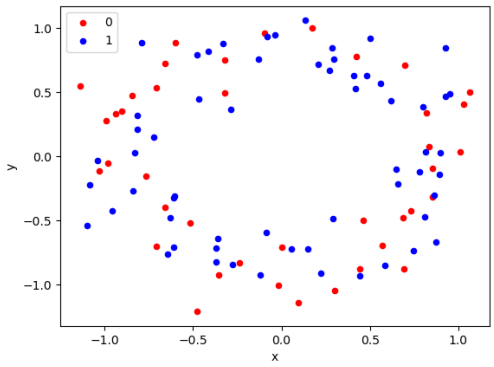
(a) Problema fácil com 100 instâncias



(b) Problema médio com 100 instâncias



(c) Problema difícil com 100 instâncias

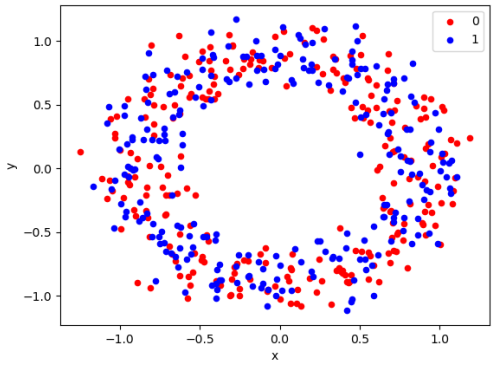


(d) Problema muito difícil com 100 instâncias

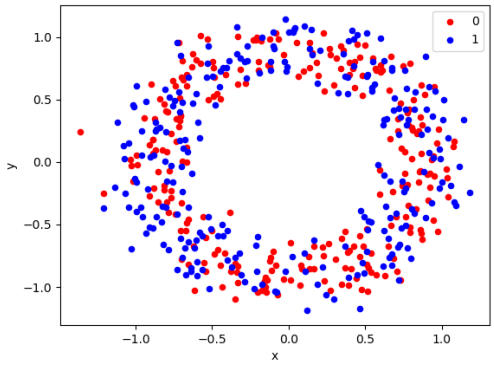
Figura 15 – Posição de cada instância em bases de dados geradas de forma aleatória.

A Figura 16 retrata o mesmo experimento da figura anterior, mudando o número de instancias de 100 para 500. As Figuras 16-a, -b, -c e -d representam uma das soluções pertencentes ao Pareto resultante, da geração aleatória de indivíduos que teve como objetivo demonstrar a diferença de desempenho dessa abordagem comparada a evolucionária (ver

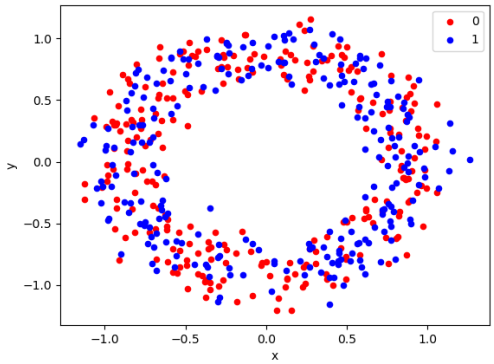
Tabela 7). Este experimento considera a geração de conjuntos de dados com 500 instâncias, 2 atributos e ruído 0,1.



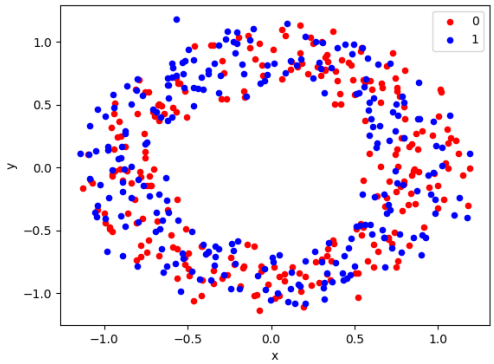
(a) Problema fácil com 500 instâncias



(b) Problema médio com 500 instâncias



(c) Problema difícil com 500 instâncias



(d) Problema muito difícil com 500 instâncias

Figura 16 – Posição de cada instância em bases de dados geradas de forma aleatória.

Podemos observar nessas figuras o quão caótico é cada uma das bases de dados geradas de modo aleatório independentemente de qual seja o valor a ser alcançado.

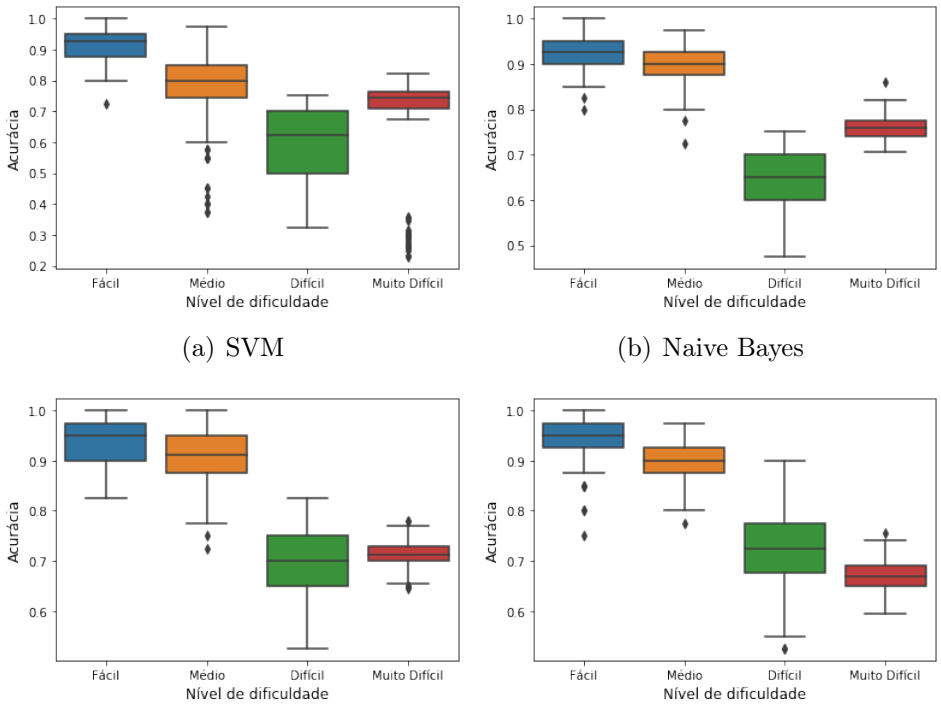
6.4 Análise do Desempenho dos Classificadores

Para complementar a análise anterior, classificadores foram executados, com seus parâmetros padrão, nos conjuntos de dados gerados de forma aleatória e nos conjuntos de dados gerados usando a abordagem evolucionária.

Na Figura 17 foram usados os conjuntos de dados com 100 instâncias, 2 atributos e 2 classes, a figura mostra a acurácia média alcançada (eixo x) para cada classificador em diferentes níveis de problemas (eixo y). A acurácia dos classificadores foi calculada

por meio do experimento de validação cruzada (cv = 10), sendo realizada 100 vezes para cálculo da média e desvio padrão. O comportamento dos classificadores *Decision Tree* e *Random Forest* (figuras 17-c e 17-d) apresentou um padrão. Conforme a complexidade do problema aumentava, a acurácia alcançada por esses classificadores também diminuía. Isso mostra a influência que as medidas de complexidade escolhidas têm sobre o desempenho desses classificadores e pode ajudar na estimativa de sua acurácia.

Figura 17 – Comparação da acurácia obtida usando os classificadores SVM, *Naive Bayes*, *Random Forest*, e *Decision Tree* usando as bases de dados geradas com 100 instâncias e 2 atributos para cada nível de problema.



(a) SVM (b) Naive Bayes

(c) Random Forest (d) Decision Tree

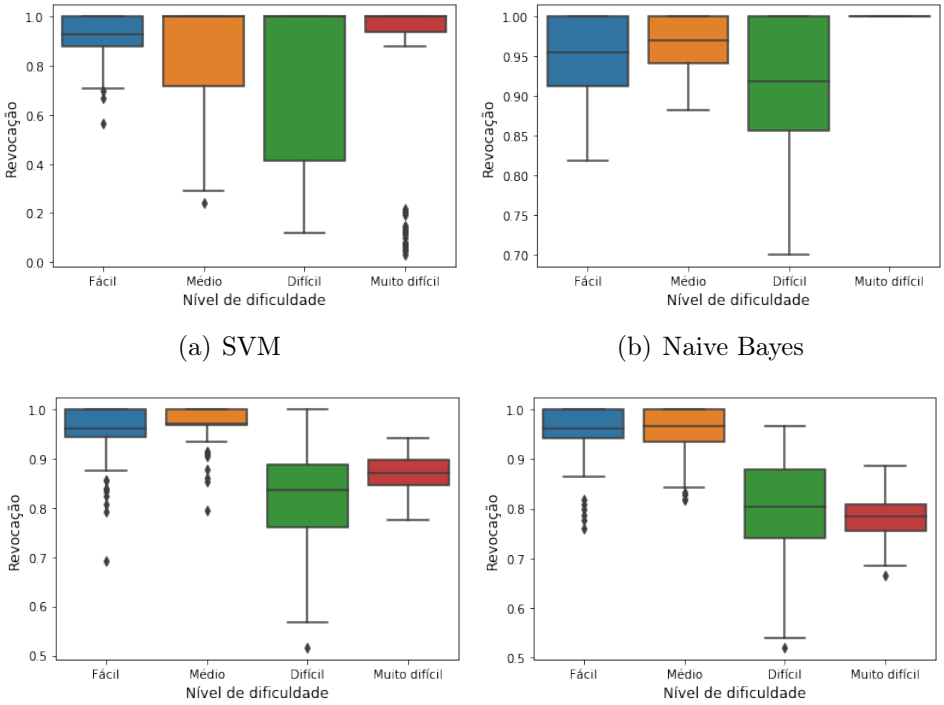
Fonte: O autor.

Os algoritmos SVM e *Naive Bayes* (figuras 17-a e 17-b) se comportaram de maneira diferente dos outros dois algoritmos para o problema classificado como difícil. Esses algoritmos são muito sensíveis à linearidade, os valores alcançados para linearidade para o problema difícil foi L2 = 0.22 e para o problema muito difícil foi L2 = 0.17, então, com relação a linearidade (L2) o problema muito difícil é mais simples que o problema difícil e isso explica o comportamento do gráfico na figura.

Na Figura 18 foram usados os conjuntos de dados com 100 instâncias, 2 atributos e 2 classes, ela mostra a revocação média alcançada (eixo x) para cada classificador em

diferentes níveis de problemas (eixo y). Um valor alto para a métrica de revocação indica que classes que deveriam ser marcadas como verdadeiras foram classificadas corretamente, a revocação dos classificadores foi calculada por meio do experimento de validação cruzada (cv = 10), sendo realizada 100 vezes para cálculo da média e desvio padrão. Os valores diferentes para cada cenário mostram a influência que as medidas de complexidade escolhidas têm sobre o desempenho desses classificadores e pode ajudar na estimativa de seu desempenho.

Figura 18 – Comparação da revocação obtida usando os classificadores SVM, *Naive Bayes*, *Random Forest*, e *Decision Tree* usando as bases de dados geradas com 100 instâncias e 2 atributos para cada nível de problema.



(a) SVM (b) Naive Bayes

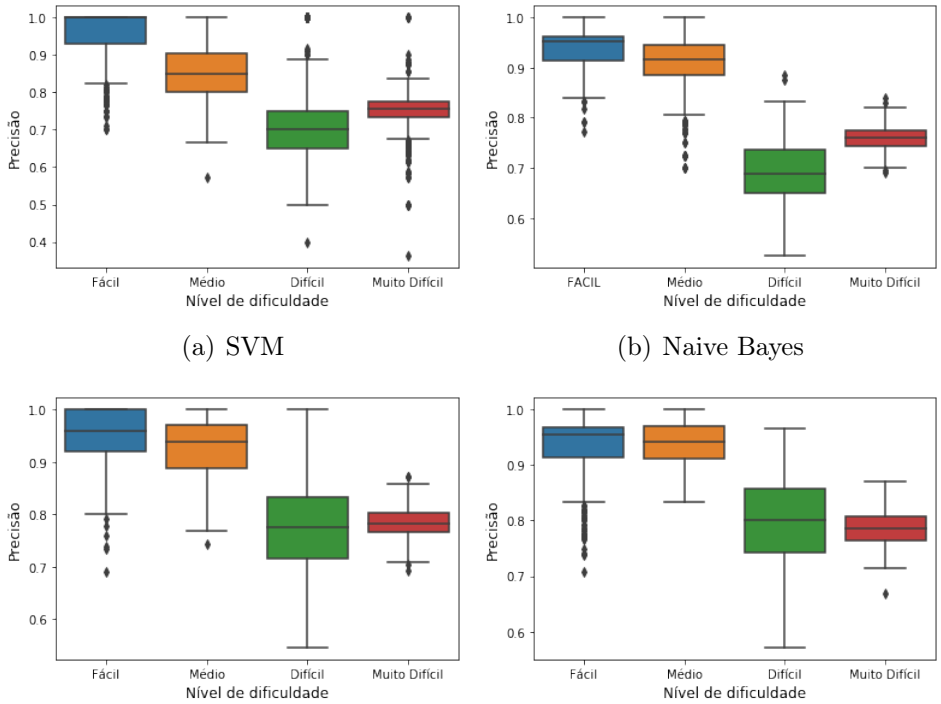
(c) Random Forest (d) Decision Tree

Fonte: O autor.

Na Figura 19 também foram usados os conjuntos de dados com 100 instâncias, 2 atributos e 2 classes, ela mostra a precisão média alcançada (eixo x) para cada classificador em diferentes níveis de problemas (eixo y). Essa métrica verifica, dentre as classificações positivas, quantas estão corretas. A precisão dos classificadores foi calculada por meio do experimento de validação cruzada (cv = 10), sendo realizada 100 vezes para cálculo da média e desvio padrão. Um comportamento interessante pode ser observado no classificador *Random Forest*, que apensar de ter como resultado um valor alto de revocação, teve um

valor relativamente baixo de precisão mostrando que muitas das métricas classificadas como positivo foram classificadas de modo errado, uma característica de problemas desbalanceados.

Figura 19 – Comparação da precisão obtida usando os classificadores SVM, *Naive Bayes*, *Random Forest*, e *Decision Tree* usando as bases de dados geradas com 100 instâncias e 2 atributos para cada nível de problema



(a) SVM (b) Naive Bayes

(c) Random Forest (d) Decision Tree

Fonte: O autor.

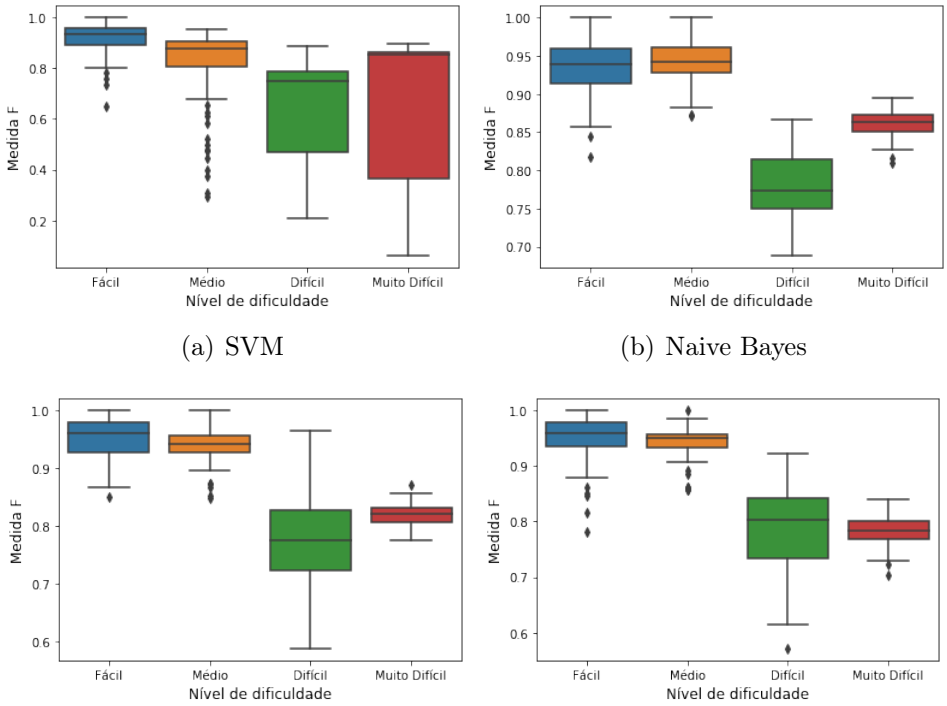
Na Figura 20 foram usados os conjuntos de dados com 100 instâncias, 2 atributos e 2 classes, ela mostra a medida F1 alcançada (eixo x) para cada classificador em diferentes níveis de problemas (eixo y). A medida F1 dos classificadores foi calculada por meio do experimento de validação cruzada (cv = 10), sendo realizada 100 vezes para cálculo da média e desvio padrão.

A medida F1 mede a média entre precisão e revocação, a Figura 20-b mostra o resultado do algoritmo *Naive Bayes*, um fato interessante é que para o problema desbalanceado "Muito Difícil"houve um *overtrainig* de resultados positivos, observamos um padrão no comportamento dos classificadores *Decision Tree* e SVM (figuras 20-a e 20-d).

A base de dados "Muito difícil"teve um desempenho melhor da métrica medida F1

que a base de dados "Difícil"para os algoritmos *Random Forest* e *Naive Bayes*, o resultado mostra que esses algoritmos são menos sensíveis a métricas de vizinhança.

Figura 20 – Comparação da medida F1 obtida usando os classificadores SVM, *Naive Bayes*, *Random Forest*, e *Decision Tree* usando as bases de dados geradas com 100 instâncias e 2 atributos para cada nível de problema.



(a) SVM (b) Naive Bayes

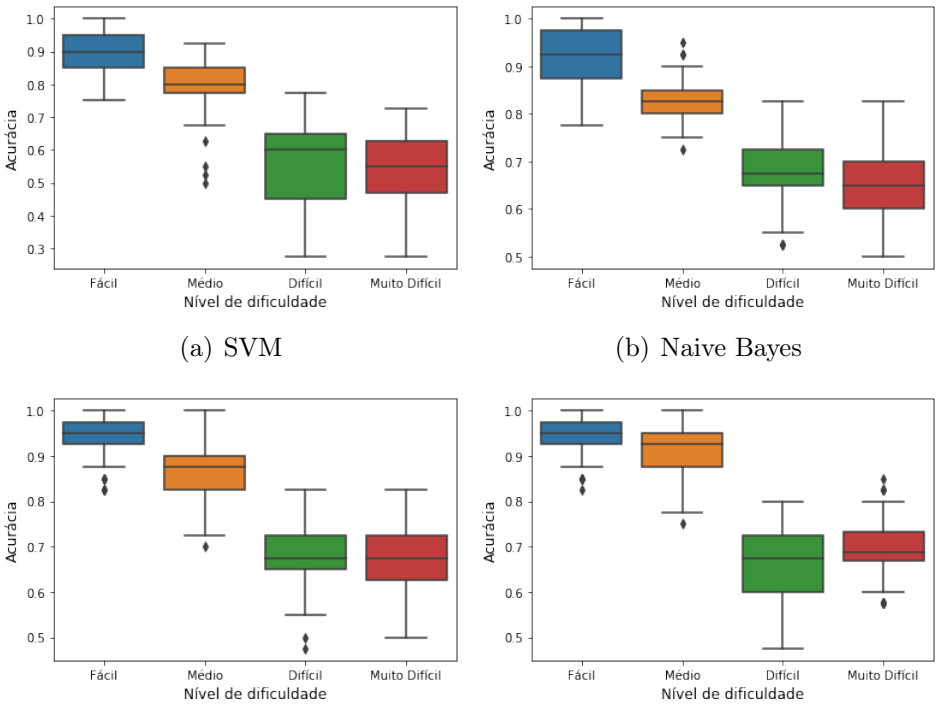
(c) Random Forest (d) Decision Tree

Fonte: O autor.

Na Figura 21 foram usados os conjuntos de dados com 100 instancias, 10 atributos e 2 classes. Ela mostra a acurácia medida alcançada (eixo x) para cada classificador em diferentes níveis de problemas (eixo y). A acurácia dos classificadores foi calculada por meio do experimento de validação cruzada (cv = 10), sendo realizada 500 vezes para cálculo da medida e desvio padrão.

Nesse exemplo, diferente do resultado encontrado nas Figuras 17-a e 17-b, temos um padrão para todos os classificadores. Conforme a complexidade do problema aumentava, a acurácia alcançada por esses classificadores também diminuía. Isso mostra que o número de atributos influencia diretamente em métricas de complexidade que levam em consideração a posição de pontos vizinhos como N1 e N2. Nessas métricas, como citado no Capítulo 2, são verificados vizinhos próximos marcados com classes diferentes. Quanto mais atributos têm uma base de dados, mais pontos teremos e consequentemente, mais relevante serão as

Figura 21 – Comparação da acurácia obtida usando os classificadores SVM, *Naive Bayes*, *Random Forest*, e *Decision Tree* usando as bases de dados geradas com 100 instâncias e 10 atributos para cada nível de problema.



(a) SVM (b) Naive Bayes

(c) Random Forest (d) Decision Tree

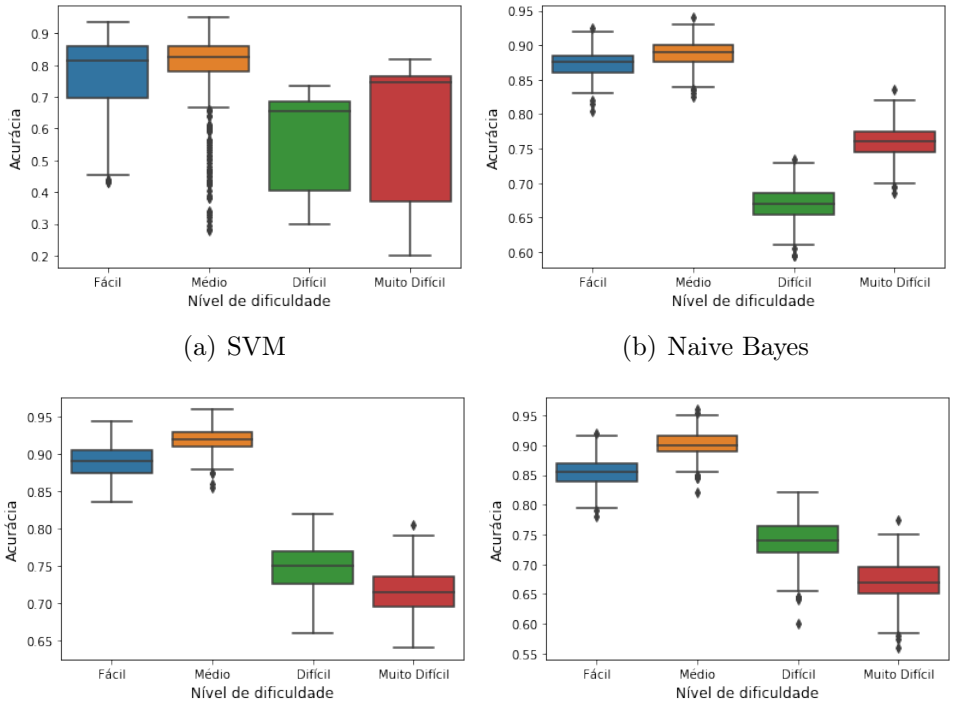
Fonte: O autor.

métricas de vizinhança.

Na Figura 22 foram usados os conjuntos de dados com 500 instâncias, 2 atributos e 2 classes, ela mostra a acurácia média alcançada (eixo x) para cada classificador em diferentes níveis de problemas (eixo y). A acurácia dos classificadores foi calculada por meio do experimento de validação cruzada (cv = 10), sendo realizada 500 vezes para cálculo da média e desvio padrão.

Nesse exemplo, diferente do resultado encontrado nas figuras 17-a e 17-b, temos um padrão para todos os classificadores. Conforme a complexidade do problema aumentava, a acurácia alcançada por esses classificadores também diminuía. Isso mostra que o número de atributos influencia diretamente em métricas de complexidade que levam em consideração a posição de pontos vizinhos como N1 e N2, nessas métricas como citado no capítulo 2, são verificados vizinhos próximos marcados com classes diferentes e quanto mais atributos têm uma base de dados, mais pontos teremos e consequentemente, mais relevante será essa métrica de complexidade.

Figura 22 – Comparação da acurácia obtida usando os classificadores SVM, *Naive Bayes*, *Random Forest*, e *Decision Tree* usando as bases de dados geradas com 500 instâncias e 10 atributos para cada nível de problema.



(a) SVM

(b) Naive Bayes

(c) Random Forest

(d) Decision Tree

Fonte: O autor.

7 Conclusão

Os algoritmos de aprendizado de máquina são comumente avaliados por meio de avaliação empírica em dados do mundo real. Porém, às vezes não há dados previamente rotulados acessíveis. Os conjuntos de dados sintéticos têm ganhado reconhecimento, como uma alternativa para avaliação eficiente do classificador, porque estão disponíveis e são altamente parametrizáveis para uma determinada tarefa de aprendizagem.

A caracterização de tais conjuntos de dados pode ser feita empregando descritores no objeto de aprendizagem em questão. Por exemplo, medidas de complexidade, que extraem características estatísticas e geométricas dos conjuntos de dados, para um determinado problema de classificação, para avaliar sua complexidade. Essas medidas de complexidade podem ser usadas para orientar a criação de conjuntos de dados sintéticos. O presente trabalho propôs o uso de um algoritmo de múltiplos objetivos para a geração de dados sintéticos considerando quatro medidas de complexidade que foram otimizadas ao mesmo tempo. Os resultados alcançados foram muito promissores, o algoritmo foi capaz de encontrar valores próximos dos ótimos para a maioria dos conjuntos de dados gerados.

Além disso, alguns algoritmos de classificação foram utilizados para testar as bases de dados sintéticas geradas. Os resultados foram analisados confirmando assim a eficácia do algoritmo genético de muitos objetivos na geração dessas bases de dados baseadas em múltiplas complexidades.

Os resultados demonstram que o problema é desafiador e a proposta é capaz de otimizar objetivos conflitantes, gerando conjuntos de dados de complexidades específicas. Os experimentos realizados mostraram o potencial da proposta.

8 Considerações Finais

8.1 Contribuições

As principais contribuições deste trabalho são: i) uso de um algoritmo de otimização de muitos objetivos (MaOA) para gerar dados sintéticos; ii) proposição de um setup experimental para o uso de tais algoritmos para o problema de geração de dados baseado na complexidade; iii) a proposta de uma ferramenta de geração de conjunto de dados baseados em complexidade descrita na seção abaixo:

8.2 CbDGEN

Como contribuição desse trabalho, em FRANÇA et al., 2020 foi proposta uma ferramenta para geração de bases de dados sintéticas totalmente parametrizável. A ferramenta CbDGEN (*Complexity Based Dataset Generator*) é composta por um sistema de algoritmos evolucionários *Many-objetive*, *Multi-objetive* e algoritmos mono objetivos de modo que o usuário pode escolher quais e quantas medidas de complexidade ele pretende otimizar e de acordo com sua escolha, o sistema escolhe entre os algoritmos presentes, qual o mais apropriado.

A Figura 23 mostra as telas de configuração da ferramenta.

Figura 23 – Telas para fornecimento de informações para a geração de bases sintéticas.

Fonte: FRANÇA et al., 2020.

A tela da esquerda recebe dados descritivos, quantidade de instancias, quantidade de classes, quantidade de atributos e o tipo dos atributos. Após preencher esses dados o

usuário irá avançar para a próxima tela (centro) onde o usuário escolhe a distribuição e o valor de ruído. Ele pode escolher entre uma das distribuições ou não escolher nenhuma e deixar a distribuição de pontos aleatória. Na tela da direita, o usuário especifica quais complexidades deseja considerar e seus respectivos níveis. Ele pode escolher mais de uma medida de complexidade a depender da sua necessidade.

8.3 Artigos Aceitos

Abaixo são mostrados alguns artigos aceitos para publicações em conferências e periódicos, frutos deste trabalho de dissertação.

- França, T. R.; Miranda, P. B. C.; Prudêncio, R. B. C.; Lorenaz, A. C.; Nascimento, A.C. A. A many-objective optimization approach for complexity-based data set generation. In:2020 IEEE Congress on Evolutionary Computation (CEC). [S.l.: s.n.], 2020.p. 1–8.
- FRANÇA, T.; MIRANDA, P.; PRUDÊNCIO, R.; NASCIMENTO, A. Cbdgen: Uma ferramenta de geração de dados sintéticos baseada na complexidade. In: SBC. Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional. [S.l.],2020. p. 128–139.

8.4 Limitações e Trabalhos Futuros

Alguns aspectos do trabalho podem ser melhorados no futuro.

- Há uma limitação de desempenho onde o aumento de exemplos deixa o problema muito complexo de modo a exigir computadores de alto desempenho. Pretende-se criar mecanismos que favoreçam o paralelismo da abordagem.
- Existem desafios com relação as métricas que podem ser otimizadas juntas. Como trabalho futuro pretende-se fazer um estudo de quais métricas são interessantes de serem otimizadas ao mesmo tempo, e quais combinações são mais influentes no desempenho dos classificadores.
- Pretende-se avaliar diferentes algoritmos de otimização de muitos objetivos para a geração de conjuntos de dados; otimizar outras medidas de complexidade diferentes e investigar seu impacto na busca de soluções; avaliar a possibilidade de gerar dados

sintéticos baseados em medidas de complexidades extraídas de dados reais; e gerar conjuntos de dados sintéticos com um maior número de classes e exemplos.

- Por fim, pretende-se construir uma meta-base para apoiar a recomendação de classificadores adequados para problemas de classificação com características específicas de complexidade.

Referências

AMANCIO, D. R.; COMIN, C. H.; CASANOVA, D.; TRAVIESO, G.; BRUNO, O. M.; RODRIGUES, F. A.; COSTA, L. d. F. A systematic comparison of supervised classifiers. **PLOS ONE**, Public Library of Science - PLOS, 2014.

ANDRA, S. F.; Fleming, P. J. Diversity management in evolutionary many-objective optimization. **IEEE Transactions on Evolutionary Computation**, v. 15, n. 2, p. 183–195, 2011.

ASUNCION, A.; NEWMAN, D. **UCI machine learning repository**. [S.l.]: Irvine, CA, USA, 2007.

BADER, J.; Zitzler, E. Hype: An algorithm for fast hypervolume-based many-objective optimization. **Evolutionary Computation**, v. 19, n. 1, p. 45–76, 2011.

BRAMS, S. J.; FISHBURN, P. C. Voting procedures. **Handbook of social choice and welfare**, Elsevier, v. 1, p. 173–236, 2002.

DAS, I.; DENNIS, J. E. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. **SIAM journal on optimization**, SIAM, v. 8, n. 3, p. 631–657, 1998.

DAVIS, L. Handbook of genetic algorithms. CumInCAD, 1991.

DEB, K.; JAIN, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. **IEEE Transactions on Evolutionary Computation**, v. 18, n. 4, p. 577–601, 2014.

DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. **IEEE Transactions on Evolutionary Computation**, v. 6, n. 2, p. 182–197, 2002.

EMERSON, P. The original borda count and partial voting. **Social Choice and Welfare**, Springer, v. 40, n. 2, p. 353–358, 2013.

FILHO, D. B. F.; JUNIOR, J. A. S. Desvendando os mistérios do coeficiente de correlação de pearson (r). **Revista Política Hoje**, v. 18, n. 1, p. 115–146, 2009.

FRANÇA, T.; MIRANDA, P.; PRUDÊNCIO, R.; NASCIMENTO, A. Cbdgen: Uma ferramenta de geração de dados sintéticos baseada na complexidade. In: SBC. **Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional**. [S.l.], 2020. p. 128–139.

FRANÇA, T. R.; MIRANDA, P. B. C.; PRUDÊNCIO, R. B. C.; LORENAZ, A. C.; NASCIMENTO, A. C. A. A many-objective optimization approach for complexity-based data set generation. In: **2020 IEEE Congress on Evolutionary Computation (CEC)**. [S.l.: s.n.], 2020. p. 1–8.

GARĆIA, L. P. F.; CARVALHO, A. C. de; LORENA, A. C. Noisy data set identification. In: SPRINGER. **International Conference on Hybrid Artificial Intelligence Systems**. [S.l.], 2013. p. 629–638.

HADKA, D.; REED, P. Borg: An auto-adaptive many-objective evolutionary computing framework. **Evolutionary Computation**, v. 21, n. 2, p. 231–259, 2013.

HARPER, P. R. A review and comparison of classification algorithms for medical decision making. **Health Policy**, Elsevier, v. 71, n. 3, p. 315–331, 2005.

HISAO ISHIBUCHI; NORITAKA TSUKAMOTO; YUSUKE NOJIMA. Evolutionary many-objective optimization: A short review. In: **2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)**. [S.l.: s.n.], 2008. p. 2419–2426.

HO, T. K.; BASU, M. Complexity measures of supervised classification problems. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 24, n. 3, p. 289–300, 2002.

HOLLAND, J. H. et al. **Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence**. [S.l.]: MIT press, 1992.

KAUR, P.; SINGH, M.; JOSAN, G. S. Classification and prediction based data mining algorithms to predict slow learners in education sector. **Procedia Computer Science**, Elsevier, v. 57, p. 500–508, 2015.

KOZA, J. R. **Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems**. [S.l.: s.n.], 2004. v. 34.

LINDEN, R. **Algoritmos genéticos (2a ediçao)**. [S.l.]: Brasport, 2008.

LORENA, A. C.; COSTA, I. G.; SPOLAÔR, N.; SOUTO, M. C. D. Analysis of complexity indices for classification problems: Cancer gene expression data. **Neurocomputing**, Elsevier, v. 75, n. 1, p. 33–42, 2012.

LORENA, A. C.; GARCIA, L. P. F.; LEHMANN, J.; SOUTO, M. C. P.; HO, T. K. How complex is your classification problem? a survey on measuring classification complexity. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 52, n. 5, set. 2019. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3347711>.

MACIÀ, N.; BERNADO-MANSILLA, E.; ORRIOLS-PUIG, A. Preliminary approach on synthetic data sets generation based on class separability measure. In: **2008 19th International Conference on Pattern Recognition**. [S.l.: s.n.], 2008–a. p. 1–4.

Macià, N.; ORRIOLS-PUIG, A.; BERNADó-MANSILLA, E. Beyond homemade artificial data sets. In: **Proceedings of the 4th International Conference on Hybrid Artificial Intelligence Systems**. Berlin, Heidelberg: Springer-Verlag, 2009. (HAIS '09), p. 605–612. ISBN 9783642023187. Disponível em: <https://doi.org/10.1007/978-3-642-02319-4_73>.

_____. In search of targeted-complexity problems. In: **Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation**. New York, NY, USA: Association for Computing Machinery, 2010. (GECCO '10), p. 1055–1062. ISBN 9781450300728. Disponível em: <https://doi.org/10.1145/1830483.1830674>.

MACIÀ, N.; ORRIOLS-PUIG, A.; BERNARDÓ-MANSILLA, E. Genetic-based synthetic data sets for the analysis of classifiers behavior. In: IEEE. **2008 Eighth International Conference on Hybrid Intelligent Systems**. [S.l.], 2008–b. p. 507–512.

Melo, V. V. de; Lorena, A. C. Using complexity measures to evolve synthetic classification datasets. In: IEEE. **2018 International Joint Conference on Neural Networks (IJCNN)**. [S.l.], 2018. p. 1–8.

MUñOZ, M. A.; SMITH-MILES, K. Generating custom classification datasets by targeting the instance space. In: **Proceedings of the Genetic and Evolutionary Computation Conference Companion**. New York, NY, USA: Association for Computing Machinery, 2017. (GECCO '17), p. 1582–1588. ISBN 9781450349390. Disponível em: <https://doi.org/10.1145/3067695.3082532>.

ORRIOLS-PUIG, A.; MACIA, N.; HO, T. K. Documentation for the data complexity library in c++. **Universitat Ramon Llull, La Salle**, v. 196, p. 1–40, 2010.

POPIć, S.; PAVKOVIć, B.; VELIKIć, I.; TESLIć, N. Data generators: a short survey of techniques and use cases with focus on testing. In: **2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)**. [S.l.: s.n.], 2019. p. 189–194.

POPIć, S.; VELIKIC, I.; TESLIC, N.; PAVKOVIC, B. Data generators: a short survey of techniques and use cases with focus on testing. In: . [S.l.: s.n.], 2019.

SOARES, C. Uci++: Improved support for algorithm selection using datasetoids. In: SPRINGER. **Pacific-Asia Conference on Knowledge Discovery and Data Mining**. [S.l.], 2009. p. 499–506.

TANWANI, A. K.; FAROOQ, M. Classification potential vs. classification accuracy: a comprehensive study of evolutionary algorithms with biomedical datasets. In: **Learning Classifier Systems**. [S.l.]: Springer, 2009. p. 127–144.

XIA, R.; ZONG, C.; LI, S. Ensemble of feature sets and classification algorithms for sentiment classification. **Information sciences**, Elsevier, v. 181, n. 6, p. 1138–1152, 2011.

ZITZLER, E.; THIELE, L. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. **IEEE Transactions on Evolutionary Computation**, v. 3, n. 4, p. 257–271, 1999.