Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

Algoritmos de Seleção de Protótipos podem obter desempenho similares aos algoritmos de Geração de Protótipos?

Rodolfo José de Oliveira Soares

Trabalho de Graduação

Recife
15 de dezembro de 2017

Universidade Federal de Pernambuco
Centro de Informática

Rodolfo José de Oliveira Soares

Algoritmos de Seleção de Protótipos podem obter desempenho similares aos algoritmos de Geração de Protótipos?

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: *George Darmiton da Cunha Cavalcanti*

Recife
15 de dezembro de 2017

Dedico este trabalho aos meus avós paternos, "In Memorian", maternos e aos meus pais, pois sem eles este trabalho e muitos dos meus sonhos não se realizariam.

Agradecimentos

Em primeiro lugar agradeço a Deus por todo o seu amor e por todas as oportunidades que ele já me proporcionou. Agradeço aos meus pais, Mércia e Reinaldo, aos meus irmãos, Ronald e Ruan, e a minha namorada, Vitória, que com muito amor e carinho, não mediram esforços para que eu chegasse até esta etapa da minha vida.

Agradeço a todos os meus amigos do LIVE e do MIC ETEPAM, por sempre me ajudarem suprindo dúvidas, por todas as brincadeiras e momentos de diversão. Agradeço, também, a Peterson e Jonata, meus amigos desde a infância, que sempre estiveram ao meu lado.

Ao meu orientador George, agradeço por todo o apoio e ajuda fornecida para a realização deste trabalho.

A todos que direta e indiretamente fizeram parte da minha formação, o meu muito obrigado.

A mente que se abre a uma nova ideia jamais volta ao seu tamanho original.

—ALBERT EINSTEIN

Resumo

A utilização de aprendizagem de máquina para a realização de tarefas como análise de créditos, localização *indoor*, detecção de malware, entre outras, é algo que vem sendo aplicado em várias soluções do nosso cotidiano. Em conjunto com o crescimento exponencial dos dados, esses problemas têm elevado o custo de armazenamento e aumentado o tempo para criações de modelos preditivos. Para reduzir esses custos, são propostas na literatura duas linha de pesquisas que procuram diminuir o tamanho do conjunto de dados e, além disso, melhorar as taxas de acertos dos classificadores utilizados. Esses dois estudos são nomeadas de **Geração** e **Seleção** de Protótipos. A Geração de Protótipos busca, a partir de uma base de dados, criar novas instâncias que tenha o mesmo poder de representatividade do conjunto inicial. A Seleção de Protótipos, por outro lado, seleciona um subconjunto de protótipos, escolhendo instâncias presentes no conjunto de dados inicial, selecionando exemplos relevantes, sem a necessidade de gerar novos dados. É conhecido que as técnicas de Geração alcançam melhores taxas de acertos, quando comparadas com as técnicas de Seleção. Por outro lado, para obter os protótipos gerados é necessário um tempo computacional maior, tornando as técnicas de seleção mais rápidas quando comparadas às técnicas de geração. Este trabalho propõe uma metodologia para verificar se técnicas de Seleção de Protótipos podem alcançar taxas de erros semelhantes as técnicas de Geração de Protótipos. O resultado encontrado mostra que, em 92,92% das bases de dados avaliadas, as técnicas de seleção alcançaram taxas de erros semelhantes às técnicas de geração.

Palavras-chave: Seleção de Protótipos; Geração de Protótipos; Metodologia; Avaliação.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

1.1 Motivação

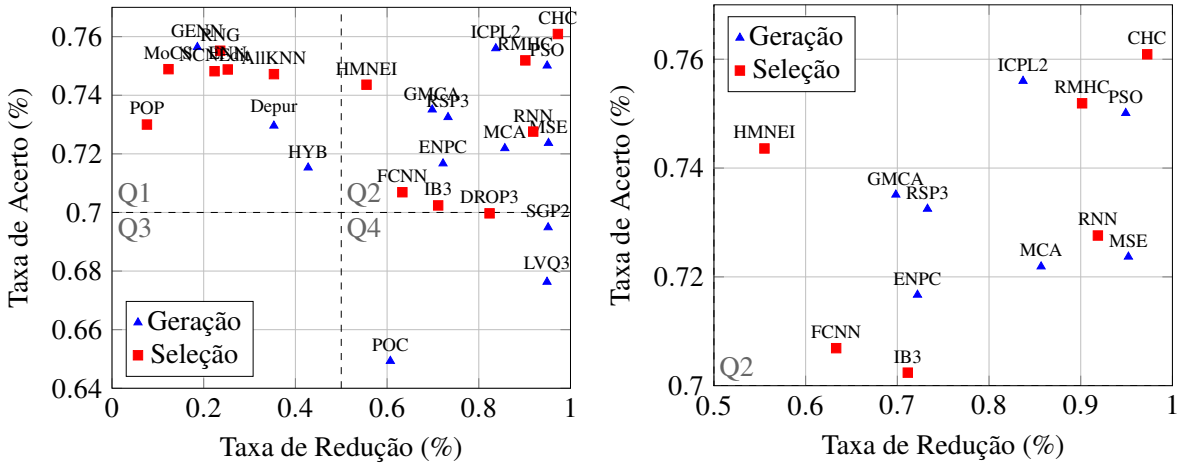
Em aprendizagem de máquina, classificação de padrões tem como objetivo tomar decisões futuras baseado em informações previamentes conhecidas. Podemos conceituar padrão como um conjunto de características (atributos) que definem um objeto ou um grupo de objetos [10]. Seja x um padrão e $\hat{w}$ um rótulo (classe) pré-determinado, o objetivo de uma máquina de aprendizagem é definir uma função $f : x \rightarrow \hat{w}$, no qual mapeia cada padrão de entrada x em uma classe $\hat{w}$, transformando a máquina de aprendizagem em um modelo preditivo que aprende com as informações presentes nesses padrões.

A quantidade de informações produzidas na *web* cresce de forma exponencial. O investimento de redes móveis, combinado com o uso diário de *smartphones*, são fatores diretamente ligados à produção dessas informações. Contudo, esse crescimento exponencial no volume de dados gera um aumento do custo computacional [10], prejudicando o processo de treinamento de máquina de aprendizado.

Na literatura existem duas linhas de pesquisas que tratam da redução do tamanho desses conjuntos de dados, com o objetivo de melhorar as taxas de acertos dos classificadores utilizados. Esses estudos são conhecidos como: **Geração** e **Seleção** de Protótipos. Na Geração de Protótipos, as técnicas procuram criar novas instâncias a partir do conjunto original de dados, permitindo preencher regiões no domínio do problema que não possuem representatividade nos dados [9]. Pro outro lado, técnicas de Seleção de Protótipos selecionam padrões já existentes no conjunto de dados. A principal vantagem dessas técnicas é a capacidade de escolher exemplos relevantes, sem a necessidade de gerar novos dados [13].

De acordo com a literatura, [9, 13, 14], as técnicas de Geração de Protótipos alcançam melhores resultados quando comparadas às técnicas de Seleção de Protótipos, utilizando como métrica de avaliação a **Taxa de Redução**, a **Taxa de Acerto** e o **Tempo de Execução**. A Taxa de Redução informa a capacidade de uma técnica de redução de dados em diminuir o tamanho do conjunto original. A Taxa de Acerto informa o poder de generalização de um classificador quando, o mesmo, utiliza o conjunto de protótipos no seu processo de treinamento.

Em [9] e [13], os autores realizam um estudo comparativo para técnicas de Geração de Protótipos e Seleção de Protótipos, separadamente, com o objetivo de avaliar o desempenho dessas técnicas sobre diversas bases de dados. A partir desses estudos, foram selecionadas 26 técnicas

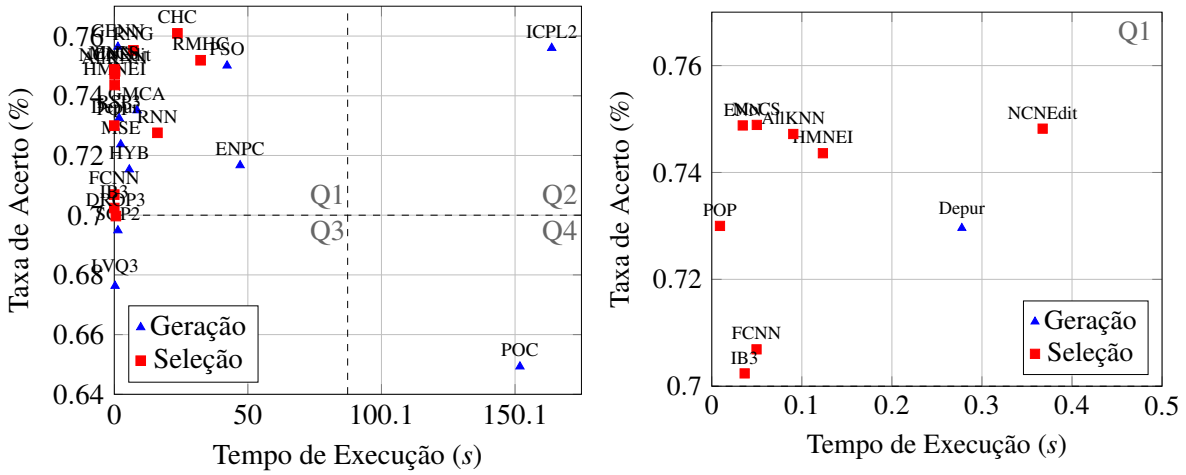


(a) Técnicas de redução de dados.

(b) Técnicas com melhores desempenho.

Figura 1.1: Comparação da Taxa de Redução (%) *versus* Taxa de Acerto (%) para técnicas de Geração e Seleção de Protótipos, a partir dos resultados encontrados em [9] e [13].

de redução de dados que apresentaram melhores taxa de acerto e taxa de redução. A Figura 1.1a apresenta o resultado encontrado para 26 técnicas de redução de dados, igualmente divididas em 13 técnicas de Geração de Protótipos e 13 técnicas de Seleção de Protótipos. O espaço é dividido em 4 quadrantes, e os melhores resultados encontram-se no **Quadrante 2 (Q2)**. A Figura 1.1b exibe um foco dado ao Quadrante 2, exibindo as 13 técnicas que apresentam melhores resultados, sendo 7 técnicas de Geração de Protótipos e 6 técnicas de Seleção de Protótipos.



(a) Técnicas de redução de dados.

(b) Técnicas com melhores desempenho.

Figura 1.2: Comparação do Tempo de Execução (*s*) *versus* Taxa de Acerto (%) para técnicas de Geração e Seleção de Protótipos a partir dos resultados encontrados em [9] e [13].

Por outro lado, as técnicas de Geração de Protótipos apresentam um tempo de execução

maior quando comparadas às técnicas de Seleção de Protótipos. A Figura 1.2a apresenta o resultado do tempo de execução para as mesmas 26 técnicas exibidas na Figura 1.1. Diferente do resultado exibido na Figura 1.1, as melhores técnicas encontram-se no **Quadrante 1 (Q1)**, no qual apresenta maior taxa de acerto e menor tempo de execução. A Figura 1.2b exibe um foco dado ao Quadrante 1. Nota-se que as técnicas de Seleção de Protótipos possuem um tempo de execução menor.

1.2 Objetivo

O objetivo deste trabalho é propor uma metodologia para verificar se técnicas de Seleção de Protótipos podem encontrar resultados iguais ou melhores comparadas às técnicas de Geração de Protótipos. Comprovando-se essa hipótese, técnicas de seleção podem ser propostas selecionando as melhores instâncias do conjunto de treinamento, obtendo as mesmas taxas de acerto com maior velocidade que as técnicas de geração.

1.3 Estrutura do trabalho

A fim de atingir o objetivo apresentado, este trabalho está organizado da seguinte forma: o Capítulo 2 será abordada a fundamentação teórica do projeto, apresentando assuntos de relevância para o entendimento do trabalho. O Capítulo 3 apresenta a metodologia ao qual o projeto foi desenvolvido. No Capítulo 4 os resultados encontrados são exibidos. Por último, o Capítulo 5 apresenta as considerações finais e possíveis trabalhos futuros.

CAPÍTULO 2

Fundamentação Teótica

Neste capítulo serão revisados de forma breve alguns assuntos de importância para o entendimento deste trabalho.

2.1 Geração de Protótipos

Técnicas de geração de protótipos criam novas instâncias a partir de um conjunto de dados inicial. Formalmente, podemos definir geração de protótipos a seguir: seja T um conjunto de treinamento formado por t instâncias. Métodos de geração de protótipos procuram obter um conjunto de protótipos CP, composto de c instâncias, tal que $c < t$, no qual cada instância de CP é gerada, a partir de T. Os protótipos do conjunto gerado são determinados para representar eficientemente as distribuições das classes e para discriminar bem quando usados para classificar os objetos treinados [9]. Por possuir cardinalidade menor que o conjunto de dados original, a utilização de um conjunto de protótipos reduz o tempo de avaliação de um classificador e, consequentemente, a quantidade de armazenamento.

Triguero *et. al.* [9] caracteriza as técnicas de geração de protótipos baseado em 4 relevantes propriedades: (1) Tipo de redução; (2) Conjunto de geração resultante; (3) Procedimento de geração; (4) Avaliação da estratégia. Cada uma dessas propriedades ajuda a definir cada técnicas de geração de protótipos encontradas na literatura. A seguir iremos detalhar cada uma delas.

2.1.1 Tipo de redução

Diretamente ligada à cardinalidade do conjunto de protótipos resultante, as técnicas de geração de protótipos utilizam diferentes esquemas de redução de dados para construir tais conjuntos. Melhorar a acurácia de classificação, utilização de baixo armazenamento, velocidade, são umas das principais vantagens dos diferentes esquemas de redução de dados. Por outro lado, conjuntos com um número elevado de protótipos ou um alto custo computacional são algumas das desvantagens citadas.

2.1.2 Conjunto de geração resultante

Outro fator importante referente aos conjuntos de protótipos gerados é conhecer como suas instâncias estão distribuídas ao longo do espaço de características. Algumas heurísticas focam em preservar instâncias localizadas nas regiões de decisão mantendo a acurácia sobre o conjunto

de treinamento, podendo ter seu poder de generalização afetado. Outras abordagens, por outro lado, buscam por remover ruídos obtendo uma baixa taxa de redução.

2.1.3 Mecanismos de geração

Essa propriedade define como as técnicas de geração de protótipos utilizam distintas operações para gerar as novas instâncias que constituem o conjunto de protótipos resultante. Técnicas baseadas na alteração do rótulo das instâncias aumentam o poder de generalização do classificador utilizado. Técnicas que utilizam o cálculo de centróides obtém uma alta taxa de redução. Por outro lado, métodos baseados no reposicionamento de instâncias, ajustam os protótipos gerados para preencher regiões do espaço de características que não possuem exemplos relevantes.

2.1.4 Avaliação da estratégia

Um dos pontos chaves para construir os conjuntos de protótipos é avaliar o poder de representatividade dos dados sob um classificador. Em geral, as técnicas de geração de protótipos utilizam o classificar 1-NN, obtendo melhores acurácias sobre os exemplos de teste. Por outro lado, essas técnicas podem ter um alto custo computacional, devido a computação sobre o conjunto de protótipos [9]. Outras técnicas optam por escolher um classificador diferente do 1-NN, reduzindo seu tempo de avaliação e degradando sua acurácia.

2.2 Seleção de Protótipos

Podemos definir, formalmente, seleção de protótipos a seguir: seja H um conjunto de treinamento. Técnicas de seleção de protótipos buscam obter um subconjunto S, tal que $S \subseteq H$, selecionando instâncias já existentes em H. A principal vantagem dessas técnicas é a capacidade de escolher exemplos relevantes, sem a necessidade de gerar novos dados, removendo ruídos ou instâncias redundantes [13].

Nos últimos anos, várias técnicas de seleção de protótipos foram proposta na literatura. Baseado neste fato, Salvador García *et. al.* [13], propõe uma categorização dos métodos de seleção de protótipos definidas em 3 pilares: (1) Direção da busca; (2) Tipo de seleção; (3) Avaliação da seleção. Nas subseções seguintes iremos detalhar cada uma das propriedades mencionadas.

2.2.1 Direção da busca

Esta propriedade ajuda compreender como as técnicas de seleção de protótipos construem o subconjunto de protótipos, a partir do conjunto de treinamento. A ordem de apresentação das instâncias do conjunto de treinamento é extremamente importante. Algumas heurísticas tem a capacidade de adaptação, o que propricia sua utilização em aplicação de *streams* ou aprendizagem *online* [13]. Outras abordagens possuem uma alta taxa de redução de dados, facilitando a obtenção de subconjuntos com boa precisão.

2.2.2 Tipo de seleção

Conhecer quais protótipos serão selecionados pelas técnicas de seleção de protótipos a partir de um conjunto de treinamento é um fator extremamente relevante. Algumas estratégias buscam por selecionar instâncias próximas da região de decisão, preservando a acurácia sobre o conjunto de treinamento, com uma alta taxa de redução de dados. Por outro lado, outras heurísticas selecionam instâncias localizadas internamentes, removendo ruídos e obtendo um aumento da precisão sobre o conjunto de teste.

2.2.3 Avaliação da seleção

Para encontrar um conjunto de protótipos definitivo, técnicas de seleção de protótipos avaliam conjunto candidatos, utilizado o algoritmo *k*-NN. Por ser um método simple, o *k*-NN ajuda à direcionar a busca das técnicas de seleção de protótipos [13]. Algumas técnicas utilizam o algoritmo *k*-NN sobre parte dos dados de treinamento, tornando-os mais eficiente computacionalmente. Já outras técnicas optam por utilizar o algoritmo *k*-NN sobre todo o conjunto de treinamento, em conjunto com a validação *leave-one-out*, obtendo uma melhor precisão sobre os dados de testes [13].

2.3 Aprendizagem de máquina

Aprendizagem de máquina é um campo da inteligência artificial, IA, cujo o objetivo é o desenvolvimento de técnicas computacionais que aprendem interativamente a partir de dados, encontrando *insights* ocultos, sem serem explicitamente programados. Formalmente, em [11], Tom Mitchell define aprendizagem de máquina como o estudo de algoritmos que melhoram sua performance P em alguma tarefa T com experiência E. Comumente interligada à outras áreas, tal como estatística, é utilizada em uma variedade de tarefas computacionais onde criar e programar algoritmos explícitos é impraticável. Seu uso abrange desde jogos, passando pela detecção de fraudes, análise estatísticas da bolsa de valores, reconhecimento facial até a construção de sistemas de recomendações, como o da *Netflix*.

Os métodos de aprendizagem de máquina são tipicamente divididos em 3 pilares: (1) Aprendizado supervisionado; (2) Aprendizado não supervisionado; (3) Aprendizado por reforço. Algoritmos de aprendizado supervisionado são treinados utilizando exemplos rotulados, indicando a classe a que eles pertencem. O objetivo é aprender uma regra geral que mapeia as entradas para as saídas. Por outro lado, aprendizado não supervisionado, os algoritmos não conhecem a "resposta correta", e seu propósito é explorar os dados e encontrar alguma estrutura neles. Por fim, algoritmos de aprendizado por reforço, a partir de tentativa e erro, encontram as melhores ações para cada estado. Normalmente é aplicado em problemas em que um agente tem de interagir com um ambiente (em geral, dinâmico).

2.3.1 Nearest Prototype Classifier (NPC)

Em aprendizagem de máquina, *Nearest Prototype Classifier* (*NPC*) é um método de classificação local que atribui à um padrão de teste o rótulo da classe dos exemplos de treinamento cujo protótipo é o mais próximo.

Formalmente, *NPC* consiste de um conjunto $T = (\theta_j, c_j)$ de protótipos, sendo θ_j vetor de atributos do espaço de características e $c_j, j = 1, \ldots, N$ suas correspondentes classes. Logo, dado um padrão de teste x sua classe é determinada selecionando o protótipo mais próximo de x, como definido na Equação 2.1 [15]:

$$q = \arg\min_j d(x, \theta_j), \tag{2.1}$$

sendo $d(x, \theta_j)$ uma medida de distância, e atribuindo a classe c_q do protótipo mais próximo ao padrão de teste x. Em geral, utiliza a distância euclidiana como medida de similaridade entre as amostras.

Sua execução é muito similar ao k-NN, porém ao utilizar apenas protótipos devidamente escolhidos, ao invés de todos os dados de treinamento, esse método torna-se mais efiente, uma vez que a quantidade de itens que necessitam ser armazenados e para qual um padrão de teste deve ser comparado é consideravelmente menor [15].

2.3.2 Árvore de Decisão

A árvore de decisão é um dos classificadores mais práticos e mais usados em aprendizagem de máquina. Com base em um conjunto de treinamento, uma árvore é montada e, a partir dessa árvore, pode-se classificar um padrão de teste sem necessariamente avaliar todos os seus atributos. A Figura 2.1 mostra uma representação de uma árvore de decisão.

Esse classificador utiliza a estratégia dividir para conquistar: o espaço de característica é divido em sub-espaços e a cada sub-espaços uma classe é associada. Além disso, regras do tipo **se-então** podem ser facilmente extraídas. Como os valores de um atributo definem partições do conjunto de treinamento, faz-se necessário escolher o melhor atributo que divide os exemplos de treinamento. Para tal, dois conceitos são utilizados: (1) Entropia e (2) Ganho de Informação. A Entropia mede a pureza de um conjunto de dados, ou seja, é uma medida da falta de homogeneidade dos dados de entrada em relação a sua classificação. Seja S um conjunto de treinamento, $p_\oplus$ a proporção de exemplos positivos em S e $p_\ominus$ a proporção de exemplos negativos, a entropia de S será dado por:

$$Entropia(S) = -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus \tag{2.2}$$

O Ganho de Informação mede a redução da entropia causada pela partição dos exemplos de acordo com os valores do atributo. Seja Ganho(S, A) a redução esperada na entropia de S, ordenado pelo atributo A, podemos definir Ganho de Informação como:

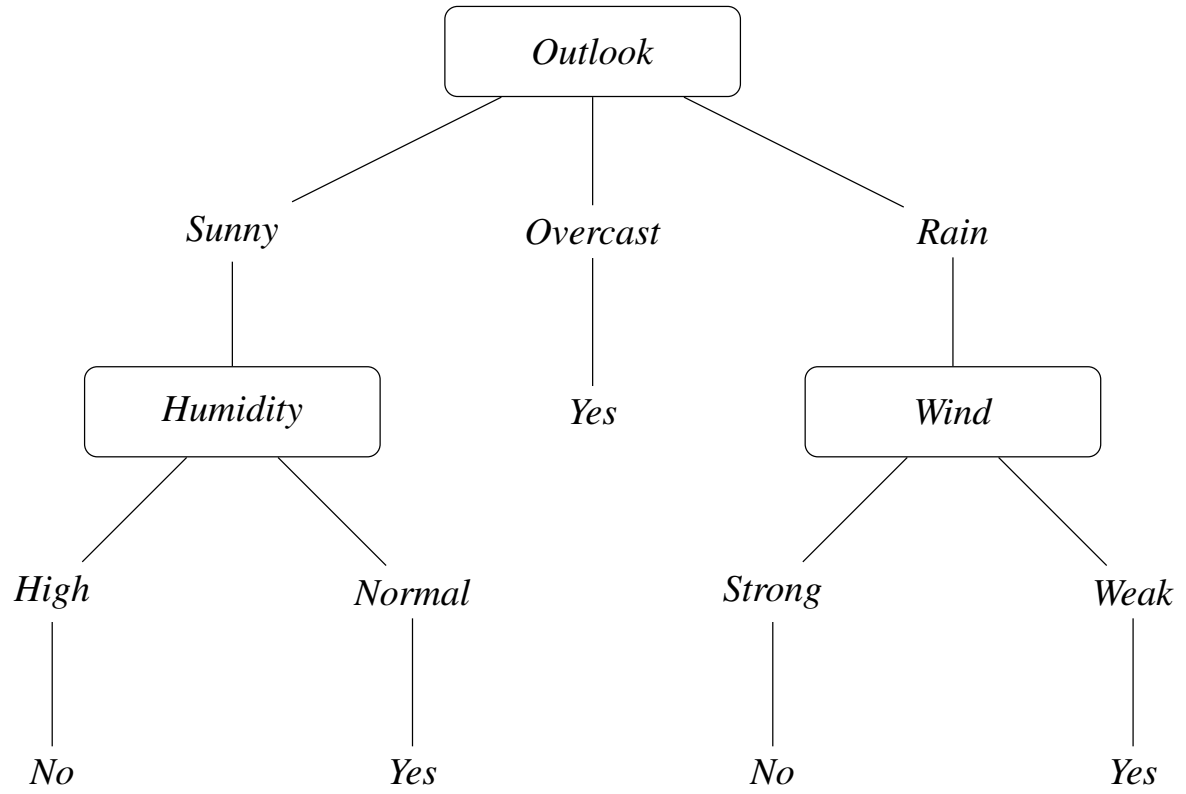


Figura 2.1: Representação de uma árvore de decisão. Fonte: "`https://goo.gl/EuWkkv`", Acessado em 06/12/2017.

$$Ganho(S,A) = Entropia(S) - \sum_{v \in valores(A)} \frac{|S_v|}{|S|} Entropia(S_v) \tag{2.3}$$

Existem vários algoritmos de classificação que utilizam a árvore de decisão. Dentre eles o algoritmo ID3 foi um dos primeiros para árvore de decisão. Proposto por Quinlan [6], este método constrói árvores de decisão a partir de um dado conjunto de exemplos, encontrando para cada nó da árvore o atributo que tem o maior ganho de informação. Ao término da construção uma etapa de poda é geralmente aplicada para melhorar a capacidade de generalização da árvore.

2.3.3 *k* Vizinhos mais próximos (*k*-NN)

O *k*-NN é um classificador simples, de fácil entendimento, e que obtém bons resultados em diversas aplicações. Essa técnica é baseada em instâncias [2], isto é, o processo de aprendizagem consiste em simplesmente armazenar os dados de treinamento.

Durante a execução do algoritmo, dado um padrão de teste, as k instâncias de treinamento que possuam o conjunto de atributos mais similar ao do padrão de teste são selecionadas. As-

sim, a classificação do padrão de teste é feita associando-o à classe que for mais frequente entre as k instâncias mais próximas. A Figura 2.2 mostra um exemplo da execução do algoritmo k-NN, sendo $k = 3$.

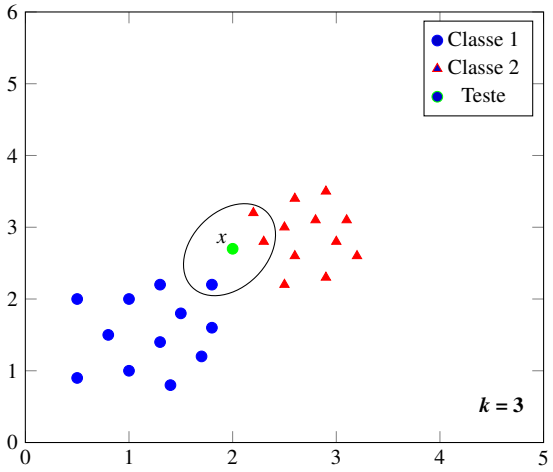


Figura 2.2: Exemplo de k-NN, com $k = 3$.

Diversas variações do algoritmo k-NN são encontradas na literatura, porém a sua forma mais geral utiliza distância euclidiana como medida de similaridade entre os padrões de treinamento e teste, e a média aritmética das saídas das k instâncias para rotular um padrão de teste.

Por mais simples seja sua implementação, o k-NN possui algumas desvantagens, entre elas ser sensível a ruídos de dados, ou seja, uma nova instância que seja inserida ao conjunto de treinamento pode facilmente impactar no resultado da classificação de um padrão de teste. Além disso, para rotular um padrão de teste o algoritmo necessita calcular e comparar a distância para todas as instâncias presentes no conjunto de treinamento, obtendo um alto custo computacional.

2.3.4 *Bagging*

Proposto por Leo Breiman, o método de *bagging* - ***b**ootstrap **agg**regat**ing*** - foi projetado para melhorar a estabilidade e acurácia, combinando classificações de conjuntos de treinamento gerados aleatoriamente. Dado um conjunto de treinamento T com n instâncias, *bagging* gera m novos conjuntos de treinamentos T_m, por amostragem com reposição, cada um consistindo de k instâncias, sendo $k = n$. Esses novos conjuntos gerados são conhecidos como *bootstrap sample*, e são combinados pela média da saída (para problemas de regressão) ou por votação (para problemas de classificação). *Bagging* leva a melhoria de procedimentos instáveis, mas por outro lado pode degradar ligeiramente o desempenho de procedimentos estáveis [8].

2.4 Teste de Hipótese

O teste de hipótese é um procedimento estatístico que permite tomar uma decisão sobre conjecturas ou suposições, utilizando os dados observados de um determinado experimento. Essas conjecturas ou suposições são chamadas de hipóteses estatísticas.

Um teste de hipótese examina duas hipóteses opostas sobre uma população: a hipótese nula e a hipótese alternativa. A hipótese nula, denotada por H_0, é a declaração que está sendo testada sobre o valor de um parâmetro (por exemplo a média) e tido como verdadeira até que as provas estatísticas indiquem o contrário. A hipótese alternativa deve ser contrária à hipótese nula, ou seja, é a afirmação que deve ser verdadeira caso a hipótese nula seja falsa. A hipótese alternativa é denotada por H_1.

O conjunto de valores assumidos pela estatística de teste para os quais a hipótese nula é rejeitada é conhecida como região crítica. Independente dos valores críticos utilizados para determinar a região crítica, as decisões tomadas estão sujeitas a erros. Dois tipos de erros podem ocorrer durante a execução de um teste de hipótese:

- Rejeitar a hipótese H_0, quando ela é verdadeira.
- Não rejeitar a hipótese H_0, quando ela é falsa.

A Tabela 2.1 resume as situações acima.

	Aceitar H_0	Rejeitar H_0
H_0 verdadeira	Decisão correta	Erro do tipo I
H_0 falsa	Erro tipo II	Decisão correta

Tabela 2.1: Tipos de erros para um teste de hipótese.

A probabilidade de cometer o Erro do tipo I é definido pelo nível de significância α, sendo $\alpha = 5\%$ o valor mais frequente utilizado. Ao diminuir a probabilidade de ocorrer um Erro do tipo I, ou seja, diminuir o valor de α, aumenta-se a probabilidade de se ocorrer um Erro do tipo II, definida como β ou poder de um teste.

Para determinar se a hipótese nula deve ser rejeitada utilizamos o p-valor. Essa medida é o menor nível de significância com que se rejeitaria a hipótese nula. No geral, um p-valor pequeno significa que a probabilidade de obter um valor da estatística de teste como o observado é muito improvável, levando assim à rejeição da hipótese nula.

CAPÍTULO 3

Metodologia Proposta e Sistema de Classificação

Neste capítulo são apresentadas a metodologia proposta (seção 3.1) e o sistema de classificação (seção 3.2) desenvolvidos. A metodologia proposta tem o objetivo de avaliar se técnicas de seleção de protótipos podem obter taxas de erro semelhantes às técnicas de geração de protótipos. Já o sistema de classificação foi desenvolvido com o objetivo de predizer qual das técnicas, geração ou seleção de protótipos, deve ser aplicada em cada uma das bases de dados utilizadas.

3.1 Metodologia Proposta

Nesta seção será apresentada a metodologia proposta. Para facilitar um melhor entendimento da metodologia proposta, segue a notação adotada:

- Γ - Conjunto de treinamento;
- Ψ - Conjunto de teste;
- C_P - Conjunto de protótipos obtido a partir de uma técnica de geração;
- p - Protótipo gerado;
- x_i - Instância mais próxima a p, tal que *classe(p)* = *classe(x_i)*;
- C_S - Conjunto de instâncias x_i selecionadas;
- e_P - Taxas de erro obtidas na geração de protótipos;
- s_P - Desvios padrões obtidas na geração de protótipos;
- μ_P - Média das taxas de erro obtidas na geração de protótipos;
- e_S - Taxas de erro obtidas na seleção de protótipos;
- s_S - Desvios padrões obtidos na seleção de protótipos;
- μ_S - Média das taxas de erro obtidas na seleção de protótipos;
- α - nível de significância do teste *T-Student*.

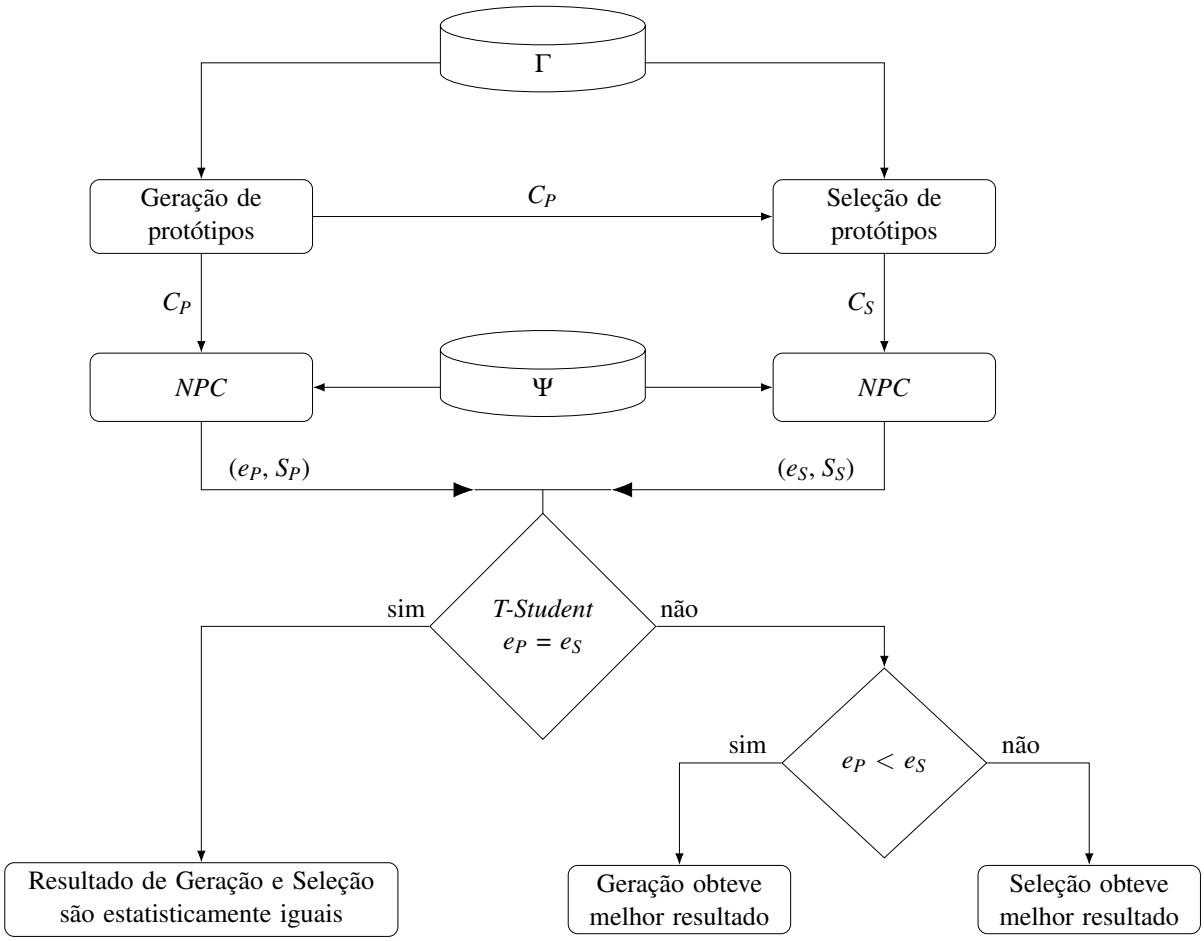


Figura 3.1: Fluxograma da metodologia proposta.

A Figura 3.1 apresenta as etapas da metodologia proposta. Essa metodologia é dividida em 4 fases: **Geração de protótipos**, **Seleção de protótipos**, **Teste** e **Avaliação**. A etapa de geração de protótipos visa a criação de um conjunto de protótipos a partir de um conjunto de treinamento Γ. A etapa de seleção de protótipos tem como objetivo selecionar instâncias do conjunto de treinamento mais próximas dos protótipos gerados. Na etapa de teste, o objetivo é avaliar o conjunto Ψ utilizando os conjuntos definidos nas etapas anteriores. E, por fim, a etapa de avaliação tem a finalidade de analisar os resultados obtidos na etapa de teste. Essa divisão ajuda a compreender o papel desempenhado por cada etapa do processo, e seus respectivos detalhamentos são apresentados nas subseções seguintes.

O pseudocódigo da metodologia proposta é apresentado em Algoritmo 2 (Anexo I).

3.1.1 Etapa de geração de protótipos

A Figura 3.2 apresenta a etapa de geração de protótipos. Inicialmente, o conjunto de treinamento Γ é fornecido como entrada para alguma técnica de geração de protótipo. Por sua vez,

o método de geração irá, a partir do conjunto de treinamento, gerar um conjunto de protótipos C_P de tamanho menor, criando protótipos artificialmente, visando melhorar a precisão do classificador.

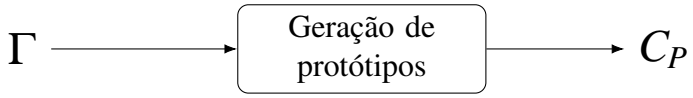


Figura 3.2: Fluxograma da etapa de geração de protótipos

3.1.2 Etapa de seleção de protótipos

Ao término da etapa de geração de protótipos, o conjunto de treinamento Γ e o conjunto de protótipos C_P são utilizados pelo algoritmo de seleção de instância VMPMC - Vizinho mais próximo de mesma classe. O VMPMC procura no conjunto de treinamento Γ as instâncias x_i mais próximas de cada protótipo gerado $p \in C_P$, de modo que a classe da instância x_i seja igual à classe do protótipo gerado p, como apresentado na Figura 3.4.

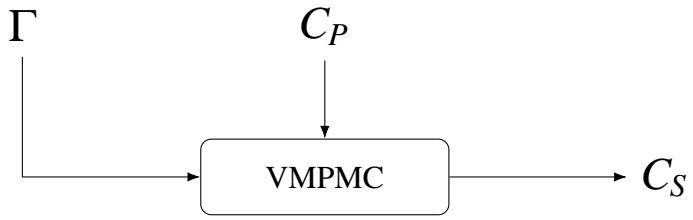


Figura 3.3: Fluxograma da etapa de seleção de protótipos

Na Figura 3.4, é exibido o exemplo de uma iteração do algoritmo VMPMC. Nela é apresentado o protótipo gerado p que pertence a classe 1 (●) e a instância x_i mais próxima do protótipo p, cuja classe é a mesma. Para calcular a distância d entre o protótipo gerado e as instâncias do conjunto de treinamento, é utilizado a distância euclidiana.

O pseudocódigo do algoritmo VMPMC é exibido no Algoritmo 1 (Anexo 1). O objetivo desse método é encontrar quais instâncias do conjunto de treinamento Γ são mais próximas dos protótipos gerados. Esse algoritmo recebe como entrada o conjunto de treinamento Γ e o conjunto de protótipos C_P. Inicialmente, ambos os conjuntos, Γ e C_P, são copiados, evitando alguma modificação após o término da execução do método VMPMC. Além disso, um conjunto vazio de instâncias selecionadas C_S é definido. A cada iteração do algoritmo, um protótipo $p \in C_P$ é obtido, e sua distância para todas as instâncias de Γ é calculada e inserida em um vetor de distâncias D. Ao fim do cálculo da distância para o protótipo p, ordenamos o conjunto Γ, em ordem crescente, de acordo com o vetor D. Em seguida, procuramos em Γ a instância x_i mais próxima de p, de mesma classe, e adicionamos ao conjunto C_S. Ao término de todas as iterações o conjunto C_S é retornado como resultado do algoritmo.

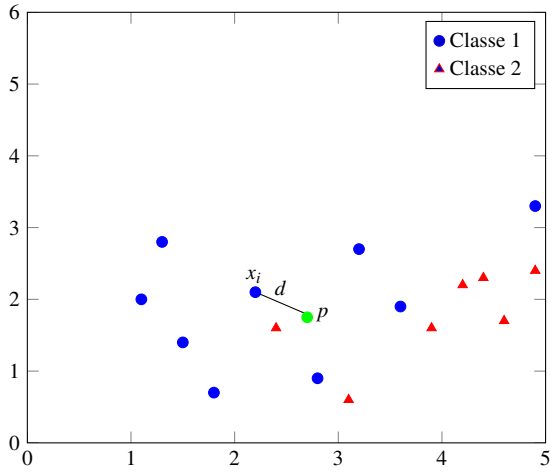


Figura 3.4: Seleção de instâncias x_i

Algoritmo 1: VMPMC

Entrada: Conjuntos Γ e C_P

```
1  início
2      T ← clone(Γ)
3      C ← clone(C_P)
4      C_S ← ∅
5      para cada c ∈ C faça
6          D ← ∅
7          para cada t ∈ T faça
8              d ← Distancia − euclidiana(c,t)
9              D ← D ∪ {d}
10         fim
11         Ordena(T,D)
12         para cada t ∈ T faça
13             se classe(c) == classe(t) então
14                 C_S ← C_S ∪ {t}
15                 goto linha 5
16             fim
17         fim
18     fim
19     retorna C_S
20 fim
```

3.1.3 Etapa de teste

A Figura 3.5 exibe o fluxograma da fase de teste. Nesta etapa o objetivo é utilizar os conjuntos C_P e C_S, separadamente, para avaliar o conjunto de teste Ψ. Para isso, utilizaremos o classifica-

dor *Nearest Prototype Classifier (NPC)* [15] e cada um dos conjuntos, C_P e C_S, são utilizados como conjunto referência do classificador. Ao término da classificação, é obtida as taxas de erros, e_G e e_S, e os desvios padrões, s_G e s_S, para os conjuntos C_P e C_S, respectivamente.

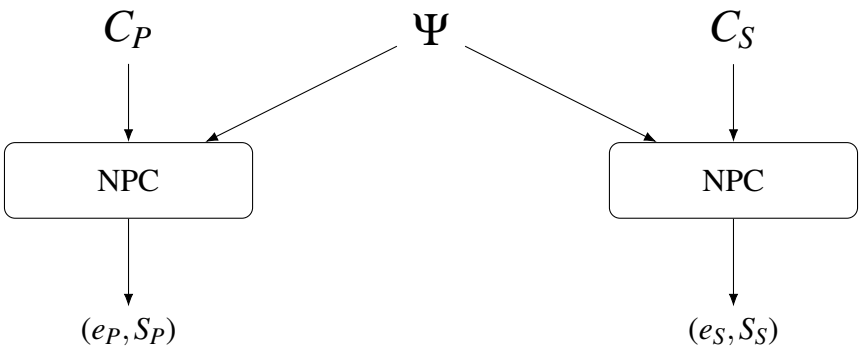


Figura 3.5: Fluxograma da etapa de Teste

3.1.4 Etapa de avaliação

Para validar que técnicas de seleção de protótipos podem obter resultados estatisticamente iguais ou semelhantes, quando comparadas as técnicas de geração de protótipos, utilizaremos nessa etapa o teste de hipótese *T-Student*. A Figura 3.6 apresenta o fluxo da etapa de avaliação.

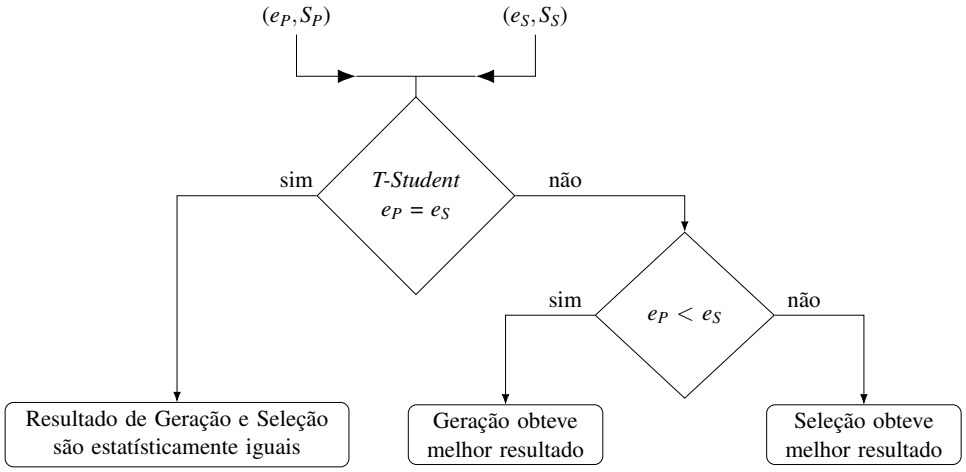


Figura 3.6: Fluxograma da etapa de avaliação dos resultados

Seja H_0: $\mu_G = \mu_S$ nossa hipótese nula e H_a: $\mu_G \neq \mu_S$ nossa hipótese alternativa, e $\alpha = 5\%$. Caso a hipótese nula seja rejeitada a partir do teste *T-Student*, podemos afirmar que as médias, μ_G e μ_S, são estatisticamente diferentes, ou seja, as técnicas de geração e seleção de protótipos obtiveram resultados distintos e, consequentemente, a técnica que tiver a menor taxa de erro, irá obter o melhor resultado. Por outro lado, caso a hipótese nula seja aceita, podemos concluir

que as taxas de erros são estatisticamente iguais, portanto, os resultados encontrados para técnicas de geração e seleção de protótipos são semelhantes.

3.2 Sistema de Classificação

Nesta seção, um sistema de classificação é proposto com o objetivo de predizer qual das técnicas de redução de dados, Geração ou Seleção de Protótipos, deve ser utilizada em uma determinada base de dados. A Figura 3.7 apresenta o fluxograma do sistema de classificação.

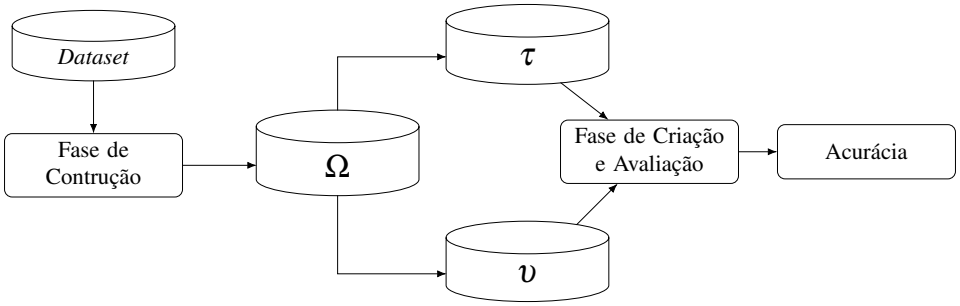


Figura 3.7: Fluxograma do Sistema de Classificação

Tal sistema é dividido em 2 partes: **Fase de Construção da Base de Dados**, **Fase de Criação e Avaliação**. A etapa de Construção tem o objetivo rotular as bases de dados utilizada, a partir da extração de características e aplicando a metodologia proposta. E a etapa de Criação foca em treinar e avaliar uma máquina de aprendizagem. Seja Ω a base de dados geradas a partir da Fase de Construção e τ e υ , os conjuntos de treinamento e teste, respectivamente. A seguir iremos detalhar cada etapa separadamente.

3.2.1 Fase de Construção da Base de Dados

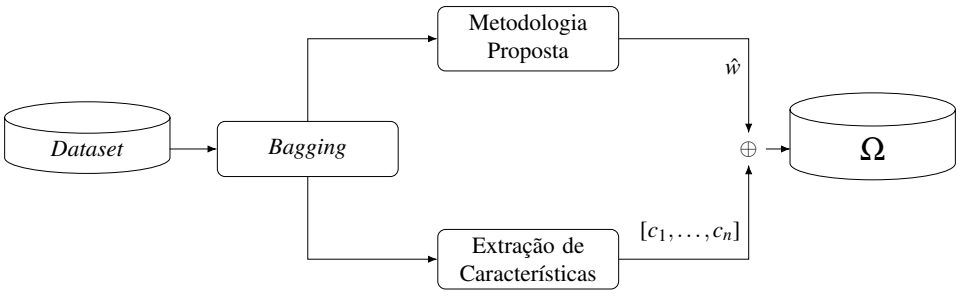


Figura 3.8: Fluxograma da fase de Construção da base de Dados

A Figura 3.8 apresenta o fluxograma da fase de construção da base de dados. Dado uma base de dados, utilizaremos a técnica de *Bagging* [8]. O objetivo desta técnica é melhorar a estabilidade e acurácia do classificador gerando dados adicionais ao treinamento, ou seja, usando

combinações com repetição, geram-se múltiplas bases de dados (de mesma cardinalidade) a partir do *dataset* original. Ao término desta fase, os novos conjuntos gerados são submetidos à metodologia proposta e a um método de extração de características.

Aplicando-se a metodologia proposta à uma base de dados, temos como saída qual técnica de redução de dados obteve melhor resultado. Seja $\hat{w}$ a saída da metodologia proposta. Ao fornecer uma base de dados d como entrada da metodologia, iremos utilizar $\hat{w}$ como o rótulo para a base de dados.

O método de extração de característica computa medidas sobre a distribuição dos dados, obtendo como saída um vetor $\vec{c} = (c_1, \ldots, c_n)$ de características para cada base de dados fornecidas como entrada, sendo n o número total de características extraídas.

Combinando o vetor $\vec{c}$ obtido a partir do método de extração de características com a saída $\hat{w}$ da metodologia proposta quando aplicada sobre o mesmo conjunto de dados, temos um vetor $\vec{i} = [c_1, \ldots, c_n, \hat{w}]$. Realizando essa processo para todas as bases de dados geradas, produzimos um conjunto de dados Ω no qual será utilizado no sistema de classificação.

3.2.2 Fase de Criação e Avalição

Nesta fase buscamos por criar e avaliar um módelo preditivo a partir do conjunto de dados gerado na etapa anterior. Inicialmente o conjunto Ω é dividido em 2 subconjuntos: τ, conjunto de treinamento e υ conjunto de teste. Em seguida, o conjunto de treinamento τ é utilizada para treinar um classificador c, no qual criará um modelo preditivo. E por último, utilizamos o conjunto de teste υ para avaliar o modelo gerado, estimando a sua acurária como forma de avalição.

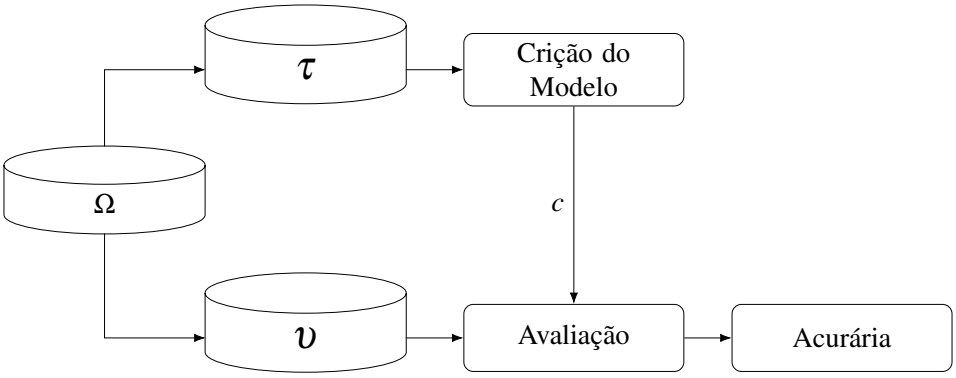


Figura 3.9: Fluxograma da fase de Criação do Modelo Preditivo

CAPÍTULO 4

Experimentos e Resultados

4.1 Experimentos

Foram realizados experimentos em 17 bases de dados: *bupa, contracptive, ionosphere, iris, led7digit, monk-2, newthyroid, page-blocks, pima, satimage, spambase, spectfheart, twonorm, vehicle, wbdc, wine, winequality-red*, disponíveis no repositório *KEEL Dataset Repository* [4]. A Tabela 4.1 resume as propriedades das bases mencionadas. Para cada *dataset*, o número de instâncias (#Instâncias), a quantidade de atributos (#Atributos) e o número de classes (#Classes) é informado.

Dataset	#Instâncias	#Atributos	#Classes
bupa	345	6	2
contraceptive	1473	9	3
ionosphere	351	33	2
iris	150	4	3
led7digit	500	7	10
monk-2	432	6	2
newthyroid	250	5	3
page-blocks	5472	10	5
pima	768	8	2
satimage	6435	36	7
spambase	4597	57	2
specfheart	267	44	2
twonorm	7400	20	2
vehicle	846	18	4
wbdc	569	30	2
wine	178	13	3
winequality-red	1599	11	11

Tabela 4.1: Resumo das informações presentes nas bases de dados utilizadas.

Para evitar que a magnitude dos valores dos atributos interfiram negativamente no cálculo da distância, uma normalização dos dados foi utilizada seguindo a Equação 4.1, sendo x o valor do atributo a ser normalizado e x_{min}, x_{max} o valor mínimo e máximo, respectivamente, para o mesmo atributo.

$$x = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.1}$$

Após a normalização, as bases de dados são divididas utilizando uma validação cruzada estratificada 10 *folders*, e para cada *fold* foi calculada sua taxa de erro. Ao final de cada execução, a média e o desvio padrão da taxa de erro foram calculados entre todos os *folders*. Para garantir que todos os métodos sejam executados sob as mesmas circunstâncias, os *folders* uma vez gerados, são mantidos os mesmos.

Para avaliar nossa metodologia proposta, utilizamos como *benchmark* o algoritmo de seleção de protótipos DROP3 - *Decremental Reduction Optimization Prodedures*. Baseado no estudo [13], a técnica DROP3 apresenta bons resultados para taxa de acerto e taxa de redução sobre diversas bases de dados, além de ser uma técnica de seleção de protótipos comumente utilizada.

Uma descrição das tecnologias utilizadas será feita a seguir a fim de apresentar características que proporcionaram sua escolha, bem como os algoritmos de geração de protótipos utilizados.

4.1.1 *Python*

Python é uma linguagem expressiva, em que é fácil traduzir o raciocínio em um algoritmo. Criada por Guido van Rossum em 1991, *Python* suporta múltiplos paradigmas de programação, possuindo tipagem dinâmica e forte. Suas principais características resume-se na utilização de baixo caracteres especiais, no uso do *garbage collector* e por sua identação em blocos. Além disso, *Python* possui uma comunidade ativa, onde seus contribuidores podem disponibilizar módulos prontos para resolver diversos problemas. E por ser *open-source*, seus módulos estão em constantes atualizações, tornando-os mais robustos e seguros em diversas situações.

Scikit-Learn

Scikit-learn é um módulo *Python* que integra uma ampla gama de algoritmos de aprendizagem de máquina para problemas supervisionados e não supervisionados [12]. Simples e eficiente, utiliza os pacotes *Numpy/Scipy* e *Matplotlib*. O pacote *Numpy/Scipy* é básico, oferecendo suporte para trabalhar com arranjos e vetores, similar a linguagens de alto nível como Matlab, porém com maior eficiência. O pacote *Matplotlib* oferece suporte a construção de gráficos 2D. Diferentemente de outras plataformas, *Scikit-learn* dispõe de uma interface simples, com licença BSD, e com foco em programação imperativa. Desta biblioteca são utilizados os algoritmos de Árvore de Decisão e k-NN.

4.1.2 *Data Complexity Library*

Proposto por Ho e Basu em 2002, complexidade de dados é uma medida de avaliação baseados em 3 princípios: a sobreposição dos dados, a separabilidade entre as classes e a geometria, topologia e densidade de *manifold*. Baseado nesse estudo, Albert Orriols-Puig et al. propuseram, em 2010, a biblioteca *Data Complexity Library*. Implementada em C++, oferece suporte para problemas supervisionados, possui conjuntos de dados com atributos numéricos, visto que a

custo computacional das medidas de complexidade de dados dependem do cálculo da distância desses atributos [1].

4.1.3 Algoritmos de Geração de Protótipos

Com base no estudo de Triguero et al. [9], foi selecionado 6 técnicas de geração de protótipos: SGP2 [3], LVQ3 [17], RPOCNN [18], GENN [5], ICPL2 [19]. O critério utilizada para a escolha dessas técnicas foi o de selecionar os métodos que apresentaram melhores taxa de acerto e taxa de redução por família, como apresentado na Figura 4.1.

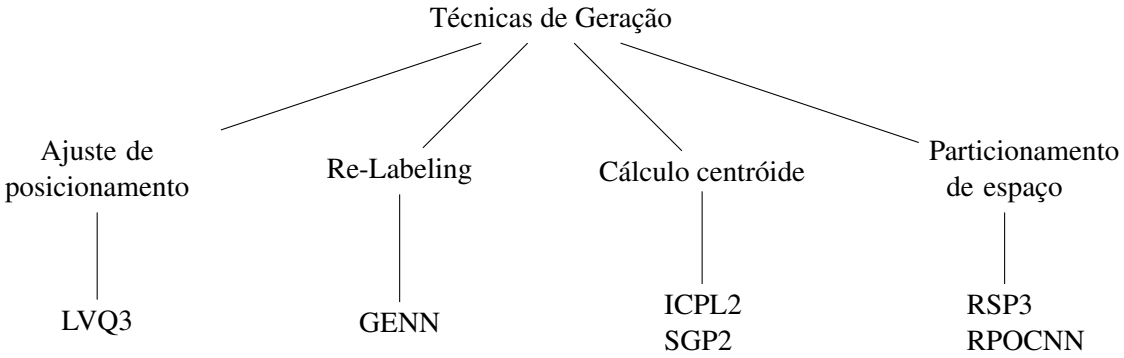


Figura 4.1: Famílias das técnicas de geração de protótipos selecionadas para o experimento.

4.1.4 Parâmetros do Experimento

Para alcançar um melhor desempenho em nosso estudo, iremos utilizar para cada técnica de geração de protótipos os parâmetros definidos em seus respectivos artigos, assumindo que esses valores são os ideias para cada método. Por fim, definimos os parâmetros para as máquinas de aprendizagem utilizadas. Para o algoritmo k-NN, o valor de k foi definido como $k = 1$. Como apresentado em [9], [13], o classificador 1-NN apresenta ótimos desempenhos sobre diversos cenários. Para o algoritmo Árvore de Decisão nenhum parâmetro foi definido, visto que o método é não paramétrico. A Tabela 4.2 apresenta os respectivos valores.

Técnica	Parâmetros
SGP	$R_{mis} = 0{,}1$; $R_{min} = 0{,}08$
LVQ	*Iterações* = 50; $\alpha = 0{,}65$; *width* = 0,2; $\varepsilon = 0{,}1$
POC-NN	*Alpharatio* = 0,05
GENN	k = 7; $k^{'} = 4$
RSP3	*Não paramétrico*
ICPL2	*Método de filtragem = ENN*
Árvore de Decisão	*Não paramétrico*
k-NN	$k = 1$

Tabela 4.2: Parâmetros utilizados em cada técnica durante os experimentos

4.2 Resultados

Nesta seção comparamos os resultados encontrados para as 17 bases de dados selecionadas quando a metodologia proposta e o sistema de classificação são aplicados.

4.2.1 Metodologia Proposta

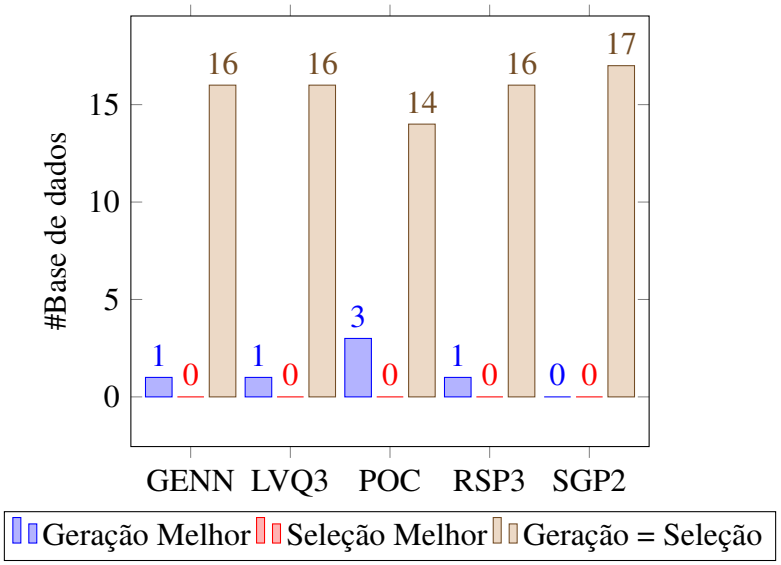


Figura 4.2: Comparação dos Resultados obtidos pela técnicas de Geração x Seleção.

A Figura 4.2 apresenta o resultado do teste *T-Student* quando aplicada a metodologia proposta. Logo, quando o teste é realizado podemos concluir que a técnica de seleção de protótipos VMPMC obteve resultados semelhantes quando comparada as técnicas de geração de protótipos, ou seja, no conjunto de treinamento Γ existe instâncias que podem ser selecionadas para aumentar a acurácia de uma máquina de aprendizagem, evitando a utilização de alguma técnica de geração de protótipos.

	Geração	Seleção	Geração = Seleção
GENN	5,8%	0	94,1%
LVQ3	5,8%	0	94,1%
POC	17,6%	0	82,3%
RSP3	5,8%	0	94,1%
SGP2	0	0	100%
Média	7,08%	0	**92,92%**

Tabela 4.3: Porcentagem das bases para cada técnica.

A Tabela 4.3 apresenta a porcentagem das bases de dados quando utilizada a metodologia proposta. Em **92,92%** das bases, a classificação utilizando o conjunto de protótipos selecionados C_S encontrou taxa de erro estatisticamente igual às técnicas de geração de protótipos, no qual valida nossa metodologia.

Em seguida, avaliamos a metodologia proposta sobre os *folders* de cada base de dados, separadamente. A Figura 4.3 apresenta o resultado do teste *T-Student*.

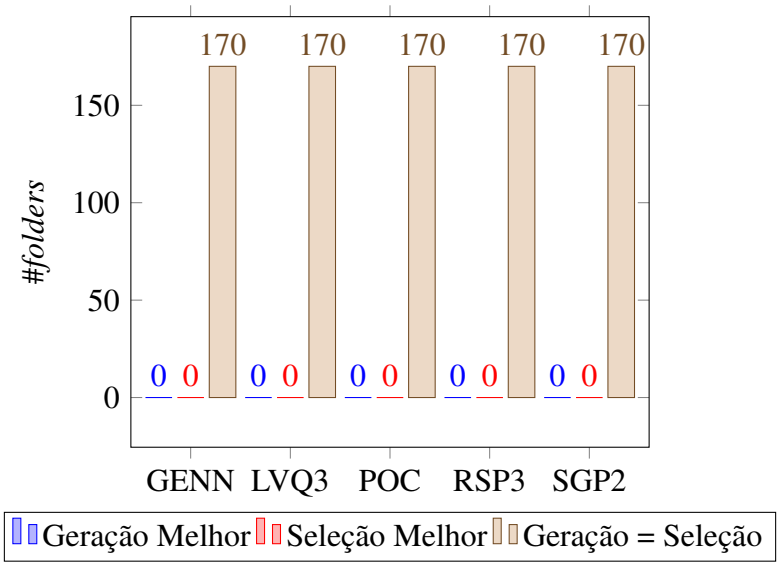


Figura 4.3: Comparação dos Resultados obtidos pela técnicas de Geração x Seleção

Analisando os resultados percebemos que as taxas de erros encontradas para a técnica de seleção de protótipos quando comparada às técnicas de geração de protótipos foram estatisticamente iguais, ou seja, no conjunto de treinamento Γ existe instâncias que podem ser selecionadas para aumentar a acurácia do classificador, evitando a utilização de alguma técnica de geração de protótipos, e diminuindo o tempo de processamento.

	Geração	Seleção	Geração = Seleção
GENN	0	0	100%
LVQ3	0	0	100%
POC	0	0	100%
RSP3	0	0	100%
SGP2	0	0	100%
Média	0	0	**100%**

Tabela 4.4: Porcentagem dos *folders* para cada técnica

A Tabela 4.4 apresenta a porcentagem dos *folders* quando utilizada a metodologia proposta. Como pode-se notar, em **100%** dos *folders*, a classificação utilizando o conjunto de protótipos

selecionados C_S encontrou taxa de erro estatiticamente semelhantes às técnicas de geração de protótipos.

Por fim, avaliamos a metodologia proposta alterando o algoritmo de seleção de protótipos VMPMC e utilizando o algoritmo do estado-da-arte DROP3. Na Figura 4.4 é exibido o resultado do teste de hipótese *T-Student*.

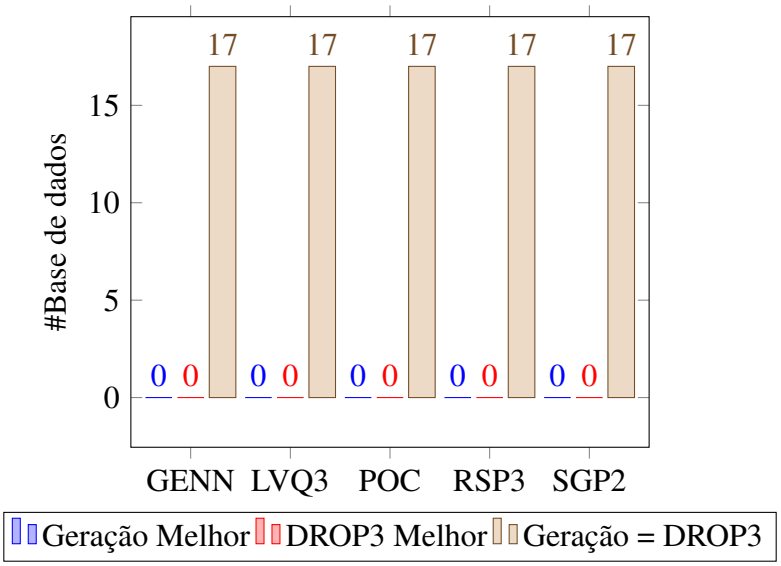


Figura 4.4: Comparação dos Resultados obtidos pela técnicas de Geração x DROP3

Como pode-se notar, em todos os cenários, as taxas de erros encontradas para a técnica de geração quando comparada a técnica de seleção DROP3 foram estatisticamente iguais, ou seja, no conjunto de treinamento Γ existe instâncias relevantes que podem substituir o conjunto de protótipos C_P, evitando assim o uso de técnicas de geração, no qual aumenta a taxa de acerto do classificador, e consequentemente, diminuindo o tempo de processamento, visto que algoritmos de seleção de protótipos, no geral, possuem tempo de execução menor quando comparados à algoritmos de geração de protótipos.

4.2.2 Sistema de Classificação

Comparamos os resultados obtidos entre os 2 classificadores utilizados a fim de estimar qual teve o melhor poder de generalização sobre o problema proposto. Para isso, utilizamos uma validação cruzada estratificada 10 *folders* sobre a base de dados gerada na **Fase de Construção** e, como realizado na metodologia proposta, para cada *fold* foi calculada sua taxa de erro. A Tabela 4.5 apresenta os resultados encontrados.

A matriz de confusão para cada classificador encontra-se em Anexo.

Fold	Árvore de Decisão	$k-NN$
Fold 1	**100%**	**100%**
Fold 2	**100%**	**100%**
Fold 3	**100%**	**100%**
Fold 4	**100%**	**100%**
Fold 5	97,05%	**100%**
Fold 6	**100%**	**100%**
Fold 7	**100%**	**100%**
Fold 8	**100%**	**100%**
Fold 9	**100%**	**100%**
Fold 10	94,11%	**100%**
Média	99,11%	**100%**

Tabela 4.5: Acurácia obtida para a Árvore de Decisão e $k-NN$ para os testes. Os melhores resultados estão em negrito.

Baseado nesses resultados, podemos concluir que o classificador $1-NN$ alcançou melhores resultados quando comparado ao classificador Árvore de Decisão, mesmo com uma diferença (0,89%) mínima quando observamos apenas a média das acurácias.

CAPÍTULO 5

Considerações Finais

Neste trabalho uma metodologia foi proposta para avaliar se técnicas de Seleção de Protótipos podem alcançar taxas de erros semelhantes às técnicas de Geração de Protótipos. A partir dos experimentos realizados com 17 bases de dados, foi possível verificar que as taxas de erros obtidas utilizando os conjuntos de protótipos selecionados, na maioria dos casos, foram semelhantes as encontradas quando os conjuntos de protótipos gerados são aplicados. Como *benchmark* utilizamos o algoritmo de seleção DROP3 e comparamos com as técnicas de geração. Os resultados mostram que a técnica DROP3 encontra taxas de erros semelhantes às técnicas de geração.

Além disso, foi proposto um sistema de classificação com o objetivo de predizer qual técnica de redução de dados, geração ou seleção, deve ser aplicada à uma determinada base de dados. Para tal, foi aplicado uma base de dados construída utilizando a biblioteca *DCol* e a metodologia proposta nas máquinas de aprendizagem: Árvore de Decisão e k-NN. Com isso, foi avalia a média para cada algoritmo, no qual a Árvore de Decisão apresentou uma média de 99,11% e o k-NN com 100%.

5.1 Trabalhos Futuros

Como possíveis trabalhos futuros, pode-se apontar:

- Aumentar a quantidade de bases de dados, utilizando bases com atributos categóricos;
- Avaliar os conjuntos de protótipos sobre outros algoritmos de aprendizagem de máquina;
- Investigar novas características que possam ser extraídas das bases de dados;

APÊNDICE A

Apêndice

Com o objetivo de facilitar o leitor, o pseudocódigo da metodologia proposta é apresentado a seguir.

Algoritmo 2: Metodologia Proposta

Entrada: Conjuntos Γ e Ψ

```
1  início
2      C_P ← Geração(Γ)
3      C_S ← VMPMC(Γ, C_P)
4      [e_P, s_P] ← NPC(C_P, Ψ)
5      [e_S, s_S] ← NPC(C_S, Ψ)
6      se T-Student (e_P, e_S, s_P, s_S, α) então
7          /* Seleção e Geração obtiveram resultados semelhantes */
8      fim
9      senão
10         se e_P < e_S então
11             /* Geração obteve melhor resultado */
12         fim
13         senão
14             /* Seleção obteve melhor resultado */
15         fim
16     fim
17 fim
```

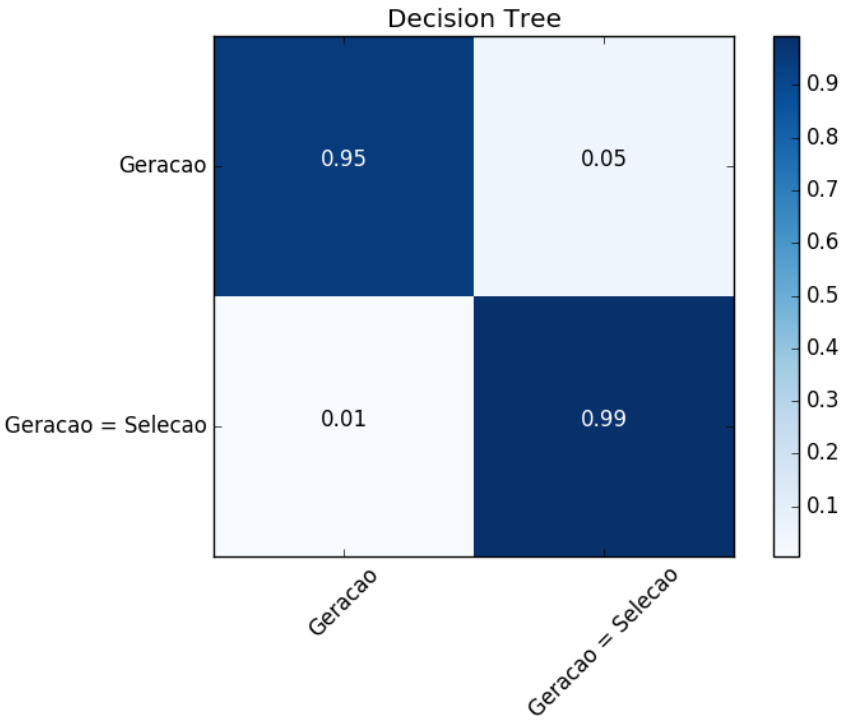


Figura A.1: Matriz de Confusão para o Algoritmo Árvore de Decisão.

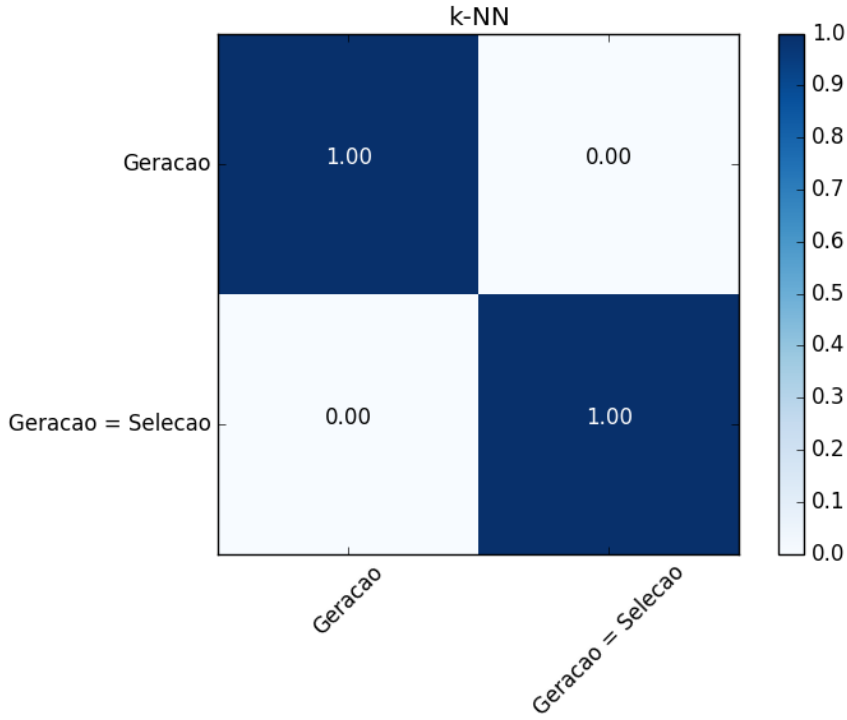


Figura A.2: Matriz de Confusão para o Algoritmo $k-NN$, cujo $k = 1$.

Referências Bibliográficas

[1] Data Complexity Library in C++. Dísponivel em: http://dcol.sourceforge.net/.

[2] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, Jan 1991.

[3] H.A. Fayed, S. R. Hashem, and A. F. Atiya. Self-generating prototypes for pattern classification. *Pattern Recognition*, 40(5):1498–1509, 2007.

[4] J. Alcal-Fdez, A. Fernandez, J. Luengo, J. Derrac and S. Garca. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Multiple-Valued Logic and Soft Computing*, 17(2-3):255–287, 2011.

[5] J. Koplowitz and T.A. Brown. On the relation of performance to editing in nearest neighbor rules. *Pattern Recognition*, 13:251–255, 1981.

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

[7] J.S. Sanchez. High training set size reduction by space partitioning and prototype abstraction. *Patter Recognition*, 37:1561–1564, 2004.

[8] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[9] L. Triguero, J. Derrac, S. Garc, and F. Herrera. A taxonomy and experimental study on prototype generation for nearest neighbor classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 42(1):86–100, 2012.

[10] Luciano S. Pereira e George D. C. Cavalcanti. Metodologia para avaliar técnicas de redução de protótipos: Protótipos gerados versus protótipos selecionados. *Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*, pages 336–341, 2014.

[11] Mitchell, Thomas M. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

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

[13] S. García, J. Derrac, J.R. Cano and F. Herrera. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):417–435, 2012.

[14] S-W. Kim and B. J. Oommen. A brief taxonomy and ranking of creative prototype reduction schemes. *Pattern Analysis and Applications*, 6(3):232–244, 2003.

[15] S.Seo, M.Bode, and K. Obemeyer. Soft nearest prototype classification. *IEEE Transactions on Neural Networks*, 14(2):390–398, 2003.

[16] T. K. Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002.

[17] T. Kohonen. Learning vector quantization for pattern recognition. *Technical Report TKK-F-A601, Helsinki University of Technology, Espoo, Finland, 1986*.

[18] T. Raicharoen and C. Lursinsap. A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (poc-nn) algorithm. *Pattern Recognition*, 26(10):1554–1567, 2005.

[19] W. Lam and C.K. Keung and D.Liu. Discovering useful concept prototypes for classification based on filtering and abstraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(8):1075–1090, 2002.

Este volume foi tipografado em LaTeX na classe UFPEThesis (www.cin.ufpe.br/~paguso/ufpethesis).