UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET
CÂMPUS GUARAPUAVA

FABIO KENJI OSHIRO TAKATUZI

ADAPTIVE-IDT: ALGORITMO INCREMENTAL PARA APRENDIZAGEM DE ÁRVORES DE DECISÃO ADAPTATIVAS

TRABALHO DE CONCLUSÃO DE CURSO

GUARAPUAVA
1º Semestre de 2017

FABIO KENJI OSHIRO TAKATUZI

ADAPTIVE-IDT: ALGORITMO INCREMENTAL PARA APRENDIZAGEM DE ÁRVORES DE DECISÃO ADAPTATIVAS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Tecnologia em Sistemas para Internet – TSI – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Guarapuava, como requisito parcial para obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Prof. Dr. Eleandro Maschio

GUARAPUAVA
2017

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Guarapuava
Curso Superior de Tecnologia em Sistemas Para a Internet

ATA DE DEFESA DE MONOGRAFIA DE TRABALHO DE CONCLUSÃO DE CURSO DO CURSO DE TSI

No dia 21 de junho de 2017, às 13:30 horas, nas dependências da Universidade Tecnológica Federal do Paraná Câmpus Guarapuava, ocorreu a banca de **defesa da monografia** de Trabalho de Conclusão de Curso intitulada: "**Adaptive-IDT: Algortimo Incremental para Aprendizagem de Árvores de Decisão Adaptativas**" do acadêmico **Fabio Kenji Oshiro Takatuzi** sob orientação do professor **Prof. Dr. Eleandro Maschio krynski** do Curso de Tecnologia em Sistemas para Internet.

Banca Avaliadora	
Membro	**Nome**
Orientador	Prof. Dr. Eleandro Maschio krynski
Coorientador	
Avaliador 1	Prof. Dr. Roni Fabio Banaszewski
Avaliador 2	Prof. Dr. Diego Marczal

Situação do Trabalho	
Situação	(x) Aprovado ( ) Aprovado com ressalvas ( ) Reprovado ( ) Não Compareceu
Encaminhamento do trabalho para biblioteca	(x) Pode ser encaminhado para biblioteca. ( ) Manter sigilo para publicação ou geração de patente.

Guarapuava, 21 de junho de 2017.

A Folha de Aprovação assinada encontra-se na Coordenação do Curso (ou Programa).

AGRADECIMENTOS

Primeiramente agradeço a Deus por me conceder a oportunidade de ter chego até aqui, por ter me dado disposição, discernimento, paciência e conhecimento para concluir este curso.

Agradeço imensamente a Professora Renata Luiza Stange Carneiro Gomes que desde o princípio esteve comigo, sendo paciente, compreensiva e prestativa, sempre disposta a responder minhas dúvidas e questionamentos da forma mais gentil e sensata, acreditando sempre no meu potencial e me incentivando a melhorar.

Aos meus familiares, por todo o apoio, amor, compreensão e paciência nesta etapa tão importante da minha vida, e que apesar de todas as dificuldades, sempre encontraram uma maneira de me fazer seguir em frente.

Ao Professor Eleandro Maschio por se disponibilizar para ser orientador deste trabalho.

Aos meus amigos, pelo apoio, pelas risadas e por todos os bons momentos proporcionados dentro da universidade e fora dela.

Aos professores da banca examinadora por todas as contribuições e pelo tempo dedicado ao meu trabalho.

A todos, Muito Obrigado!

RESUMO

TAKATUZI, Fabio K. O. *Adaptive*-IDT: Algoritmo Incremental Para Aprendizagem de Árvores de Decisão Adaptativas. 67 f. Trabalho de Conclusão de Curso – Curso Superior de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná – UTFPR. Guarapuava, 2017.

A indução de árvores de decisão é hoje um dos métodos mais utilizados para a resolução de problemas de classificação, tomada de decisão e aprendizagem de máquina (MITCHELL, 1997). Por meio deste método é possível criar uma árvore de decisão baseada em um conjunto de dados de treinamento e, posteriormente, classificar novos exemplos com base na sua estrutura. A classificação gera uma saída, denominada valor classe, que age como resposta aos valores de entrada. No entanto, existem diversas situações em que os dados de treinamento se encontram em constante mudança, como nos problemas de aprendizagem incremental, em que processo de aprendizagem considera a dinamicidade do conjunto de treinamento. Para problemas como este, a adaptatividade apresenta uma solução bastante aderente, pois permite que uma estrutura se automodifique em resposta a estímulos externos, incorporando a ela novas informações. Foi proposto neste trabalho um algoritmo para indução de árvores de decisão, denominado *Adaptive*-IDT, baseado na união de técnicas estatísticas com a tecnologia adaptativa. Por meio da sua especificação e implementação, espera-se que o mesmo possa apresentar uma solução alternativa para os métodos de aprendizagem de máquina tradicionais, além de fornecer um embasamento para futuros trabalhos que possuam enfoque na utilização da adaptatividade aplicada a métodos de classificação e aprendizagem de máquina.

Palavras-chave: Adaptatividade. Aprendizagem de Máquina. Árvores de Decisão. Aprendizado Incremental.

ABSTRACT

TAKATUZI, Fabio K. O. *Adaptive*-IDT: Incremental Algorithm for Learning Adaptive Decision Trees. 67 f. Course Completion Work - Superior Course in Technology in Internet Systems, Federal Technological University of Paraná - UTFPR. Guarapuava, 2017.

The induction of decision trees is nowadays one of the most used methods for solving problems of classification, decision making and machine learning (MITCHELL, 1997). Using this method you can create a decision tree based on a training data set and then sort new examples based on their structure. Classification generates an output, called a class value, that acts in response to input values. However, there are several situations in which training data is constantly changing, such as in the problems of incremental learning, in which learning process considers the dynamism of the training set. For problems such as this, the adaptability presents a very adherent solution, since it allows a structure to self-modify in response to external stimuli, incorporating new information. This paper proposes an algorithm for induction of decision trees, called *Adaptive*-IDT, based on the union of statistical techniques with adaptive technology. Through its specification and implementation, it is expected that it may present an alternative solution to traditional machine learning methods, as well as providing a foundation for future work that focuses on the use of adaptability applied to classification and learning methods Machine.

Keywords: Adaptability. Machine Learning. Decision Trees. Incremental Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A resolução de um problema computacional bem especificado pode ser dada através de um algoritmo. Um algoritmo é uma sequência de instruções computacionais que produzem, a partir de um valor de entrada, ou um conjunto deles, um valor ou conjunto de valores como saída (CORMEN et al., 2009). No entanto, determinadas tarefas ou problemas existentes não são passíveis de serem realizadas ou resolvidos utilizando paradigmas de programação tradicionais, tais como os imperativos, funcionais ou orientados por objetos (PISTORI, 2003). Em um sistema para reconhecimento de caracteres manuscritos, por exemplo, não cabe ao desenvolvedor obter uma representação computacional implementável da solução do problema a ser resolvido. Neste caso, o projetista do sistema deve criar um sistema computacional capaz de obter uma solução automática, alcançada a partir de exemplos de como instâncias particulares do problema são resolvidas (MITCHELL, 1997).

Nesse contexto, a Aprendizagem de Máquina (AM) é uma área da Inteligência Artificial (IA) que estuda a capacidade que os sistemas computacionais possuem de aprender e modificar seu comportamento através da experiência adquirida a fim de melhorar seu desempenho em execuções posteriores (ALPAYDIN, 2010).

Existem diversos métodos que são utilizados para a busca de solução de problemas em aprendizagem de máquina, tais como: o Aprendizado Bayesiano (DUDA et al., 2001), baseado no Teorema de Bayes; as Redes Neurais Artificiais (RNA), inspiradas no funcionamento do neurônio biológico (BISHOP, 1995); os métodos de indução de árvores de decisão, que utilizam medidas estatísticas para construir uma árvore de decisão que representa o aprendizado (QUINLAN, 1986).

Neste trabalho, o enfoque é dado às árvores de decisão, que são um dos métodos mais práticos e utilizados em aprendizagem de máquina (MITCHELL, 1997). As árvores de decisão são estruturas de dados simples, compostas de atributos e valores, sendo representados na árvore, respectivamente, por nós e ramos. O primeiro nó da árvore é chamado raiz, e por convenção fica no topo da estrutura. Os últimos nós são denominados folha ou terminais, os demais nós da árvore são chamados nós internos. As árvores de decisão possuem esse nome justamente pelo fato de que sua estrutura se assemelha a uma árvore real, como mostra a Figura 1.

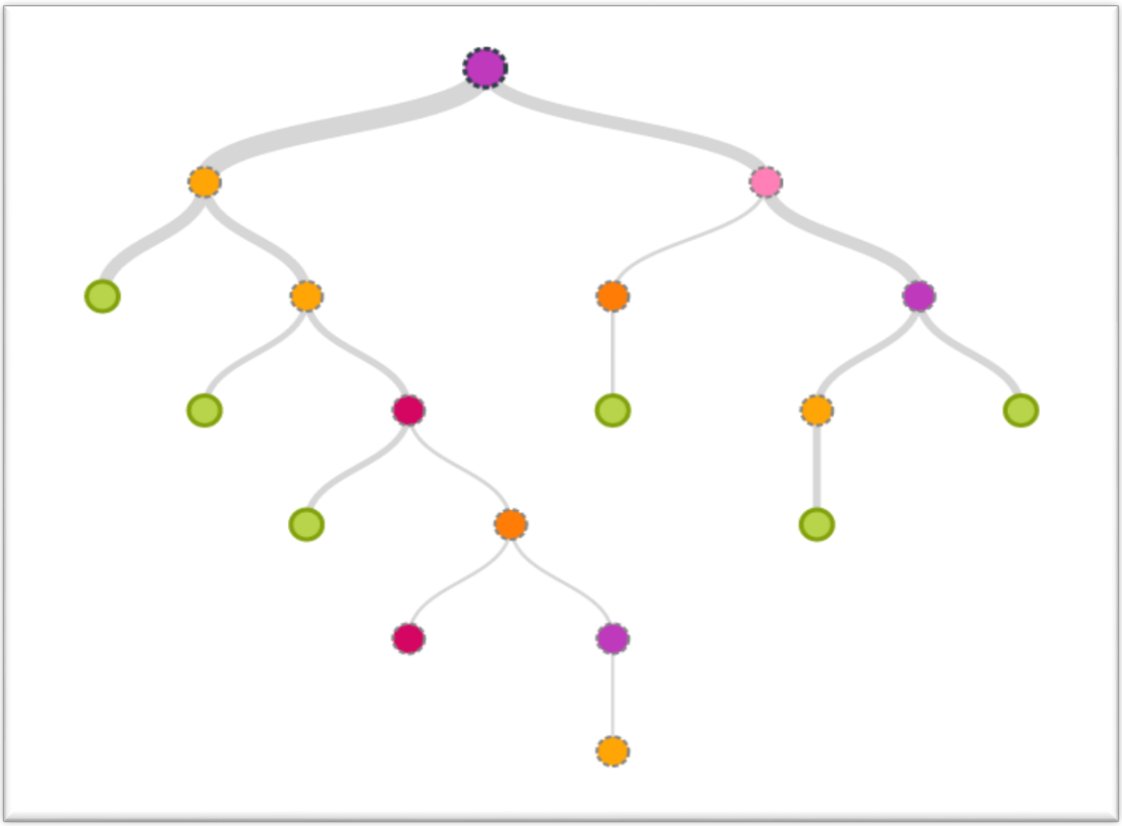


Figura 1: Exemplo de árvore de decisão.
Fonte: https://blog.bigml.com

A construção de uma árvore, similarmente a outros métodos, pode ocorrer de forma incremental ou não incremental. A maioria dos algoritmos são não-incrementais, como o ID3 (QUINLAN, 1986) e o C4.5 (QUINLAN, 1993), ou seja, não possuem a capacidade de incluir novos exemplos de como instâncias particulares do problema são resolvidas ao longo de sua execução. Alguns exemplos de algoritmos de indução árvores de decisão incrementais são o CART incremental (CRAWFORD, 1989), o ID5R (UTGOFF, 1989) e o ITI (UTGOFF; BERKMAN; CLOUSE, 1997).

A necessidade de buscar soluções alternativas de aprendizagem incremental é de grande relevância (ROSS, 2007; ELWELL; POLIKAR, 2011; CALINON; BILLARD, 2007), pois existem diversas aplicações em que novas informações vão surgindo ou se alterando ao longo do tempo, formando assim um cenário em que a utilização de algoritmos incrementais é mais conveniente. (PISTORI; NETO, 2002). Para exemplificar, considere um sistema de monitoramento climático, o qual realiza

constantemente leituras de velocidade do vento, temperatura média do ar, etc. Esse sistema deve identificar quais condições climáticas são apropriadas para que um avião possa realizar um determinado trajeto de forma mais segura. Considerando que as condições climáticas sofrem uma modificação frequente, consequentemente o conhecimento adquirido também deve ser atualizado ao logo da execução do sistema, caso contrário, tal conhecimento pode se tornar desatualizado em um curto período de tempo.

Uma forma não incremental de resolver isso é, sempre que uma nova informação é disponibilizada, o processo de aprendizagem é repetido e a estrutura que representa o conhecimento é reconstruída. De acordo com Yoshida (2007), o problema é que reconstrução da estrutura implica em um custo computacional elevado. Dessa forma, a utilização de métodos não incrementais se torna ineficiente, uma vez que as informações repassadas ao sistema necessárias para aprendizagem estão em constante modificação. Neste caso, torna-se evidente a necessidade de implementar processos dinâmicos baseados na aprendizagem incremental, de maneira que a base de conhecimento possa ser atualizada sem que todo o processo de aprendizagem necessite ser repetido.

Um conceito que tem sido adotado na solução de problemas com características dinâmicas é a adaptatividade (STANGE; CEREDA; NETO, 2017; FARIA; PEREIRA, 2017, LEME; MARTINS; VICTOR, 2017). De acordo com Neto (2011) a adaptatividade é definida como a capacidade que os sistemas possuem de promover espontaneamente alterações em seu próprio comportamento, de acordo com a necessidade, em função de seu comportamento corrente e dos valores de suas entradas. Em um sentido amplo, os computadores aprendem quando há uma mudança de comportamento a fim de executar melhor uma tarefa específica. Para um problema de aprendizagem incremental, a utilização de técnicas adaptativas pode representar uma solução bastante expressiva (NETO, 2000).

A Tecnologia Adaptativa consiste na utilização da adaptatividade de maneira prática e corresponde ao conjunto de métodos, técnicas e ferramentas que visam a resolução de problemas utilizando modelos baseados em dispositivos adaptativos (NETO, 2011). Um dispositivo adaptativo por sua vez é composto por um dispositivo não adaptativo (ex.: árvore de decisão, autômato finito), chamado de dispositivo subjacente, e uma camada adaptativa, que confere a este dispositivo a capacidade de automodificação (NETO, 2011). São exemplos de dispositivos adaptativos as

Gramáticas Adaptativas (IWAI, 2000), as Árvores de Decisão Adaptativas (PISTORI, 2003; PISTORI, H.; NETO, 2003), as Tabelas de Decisão Adaptativas (TCHEMRA, 2009) entre outros.

As Árvores de Decisão Adaptativas, as quais são o foco do presente trabalho, foram formalizadas por (PISTORI, 2003; PISTORI, H.; NETO, 2003), e permitem que a estrutura hierárquica de uma árvore de decisão convencional seja dinamicamente alterada durante o processo de decisão, ou seja, quando a árvore é percorrida da raiz para as folhas.

O primeiro algoritmo de aprendizagem baseado em árvores de decisão adaptativas foi proposto por Pistori e Neto (2002), denominado *Adaptree*. O algoritmo incremental utiliza conceitos da teoria dos autômatos e da tecnologia adaptativa, para a construção de uma árvore de decisão. Um pouco mais recente, Catae e Rocha (2011) apresentam uma possibilidade de modelar operações de paridade, ou-exclusivo (XOR) e multiplexação, através de árvores de decisão adaptativas, tais problemas não apresentavam bons resultados em estruturas de árvores de decisão tradicionais. Os demais trabalhos que envolvem árvores de decisão adaptativas se referem à aplicação das árvores de decisão adaptativa. Pistori e Souza (2010), fizeram uso das árvores de decisão adaptativas, por meio do algoritmo Adaptree para realizar estudos de caso relacionados com a área da biotecnologia. Ganzieli, et al. (2010) criam um sistema de pré-diagnóstico de câncer de pele com o uso de processamento de imagem e árvores de decisão adaptativas. Barros, et al. (2015) propõe o uso das árvores de decisão adaptativas para a criação de um classificador das condições do ambiente interior relativas ao conforto térmico e a qualidade do ar, com o objetivo de auxiliar na definição de estratégias de operações sustentáveis de sistemas de ar condicionado central, minimizando as operações manuais e intervenção humana. Já em Silva et al. (2016) foi apresentado um trabalho que propõe o uso das árvores de decisão adaptativas para classificar um determinado fato como positivo ou negativo, a partir de sentimentos exibidos nos tweets da rede social Twitter.

Neste trabalho foi proposto um algoritmo para indução de árvores de decisão denominado *Adaptive*-IDT. O mesmo, baseado no funcionamento do primeiro algoritmo de indução de árvores de decisão adaptativas de nome *Adaptree*, foi capaz de gerar, a partir de um conjunto de dados de treinamento, uma árvore de decisão não determinística como resultado. Tal árvore possui características adaptativas que permitem que sua estrutura interna seja alterada durante o processo de classificação,

possibilitando um novo enfoque para a aprendizagem incremental. O presente trabalho também descreveu uma série de conceitos a respeito da aprendizagem de máquina, tecnologia adaptativa e indução de árvores de decisão, os quais são fundamentais para a compreensão do funcionamento, estrutura e formulação da solução proposta. Por meio da disponibilização do *Adaptive*-IDT de maneira *open source*, a comunidade de pesquisa nas áreas abrangidas pelo trabalho, possui um maior embasamento para a elaboração de novas propostas, ou até mesmo para uma melhoria no algoritmo apresentado neste trabalho.

1.1 OBJETIVOS

Nesta Seção serão apresentados o objetivo geral e os objetivos específicos do trabalho.

1.1.1 Objetivo Geral

O objetivo principal deste trabalho é propor uma solução alternativa para a aprendizagem incremental de árvores de decisão utilizando a tecnologia adaptativa, com a finalidade de construir modelos de classificação e avaliar o desempenho dessa solução com algoritmos de árvores de decisão já existentes.

1.1.2 Objetivos Específicos

1. Desenvolver um algoritmo para indução de árvores de decisão adaptativas baseado no Adaptree;

2. Avaliar os modelos de classificação gerados pelo algoritmo através de experimentos padronizados.

1.2 JUSTIFICATIVA

A utilização da tecnologia adaptativa como solução para o problema de aprendizagem incremental é justificada, inicialmente, pela aderência conceitual da adaptatividade com um aspecto fundamental da aprendizagem (PISTORI, 2003): a adaptação dinâmica das regras de aprendizagem em função da sua interação com o ambiente.

As principais ferramentas disponíveis para uso da tecnologia adaptativa são: 1) ADAPTOOLS (PISTORI; NETO, 2003), para autômatos adaptativos; 2) STAD e STAD-S (ALMEIDA JUNIOR, 1995), para statecharts adaptativos; 3) LASSUS (BASSETO, 2000), para geração de música por computador usando Redes de Markov adaptativas; 4) PGPT (PICAGLI, 2008), para traduzir textos escritos em português para a representação fonética de fala; e 5) RSW (PEREIRA, 1997), utilizada para a aplicação de formalismos adaptativos em sistemas Windows. Nenhuma dessas ferramentas permite a utilização de árvores de decisão adaptativas, portanto a implementação de um algoritmo para construção de árvores de decisão adaptativas e a sua disponibilização para *download*, pode ser bastante proveitosa para a comunidade de pesquisa nesta área. Além disso, o mesmo será distribuído de maneira *open source* auxiliando assim, a construção de outros algoritmos para essa finalidade.

1.3 METODOLOGIA

A presente Seção apresenta os passos metodológicos utilizados para o desenvolvimento do trabalho, organizadas da seguinte maneira:

1. Estudo acerca dos conceitos gerais de aprendizagem de máquina e adaptatividade:

   a. Estudo do comportamento de dispositivos adaptativos aplicados à árvores de decisão;

   b. Estudo do funcionamento dos algoritmos de aprendizagem de máquina clássicos (ex. ID3, C4.5).

2. Especificação de um algoritmo para árvores de decisão adaptativas baseado no conceito da adaptatividade e aprendizagem incremental:

   a. Proposta de uma forma de generalização de conceito baseado em técnicas estatísticas;

   b. Estudo do funcionamento do algoritmo para indução de árvores de decisão adaptativas, *Adaptree*.

3. Implementação do algoritmo:

   a. Utilização da linguagem de programação JAVA;

   b. Utilização do ambiente de desenvolvimento integrado (*Integrated Development Environment - IDE*) Eclipse;

   c. Integração com a ferramenta WEKA (*Waikato Environment for Knowledge Analysis*), por meio da utilização de sua biblioteca;

   d. Testes do algoritmo realizados por meio da utilização de medidas de precisão e taxas de acerto médias.

4. Comparação do algoritmo proposto com outros algoritmos de aprendizagem, incrementais e não incrementais.

5. Treino e teste dos algoritmos utilizando conjuntos de dados disponíveis no UCI, que é um repositório de *data sets* para experimentos em aprendizagem de máquina.

2 FUNDAMENTAÇÃO TEÓRICA

Este trabalho envolve de forma bastante geral duas abordagem: a adaptatividade e a aprendizagem. Particularmente, o trabalho enfoca na utilização de árvores de decisão adaptativas na aprendizagem de máquina. A seguir são apresentados os conceitos fundamentais sobre a aprendizagem de máquina e a tecnologia adaptativa.

2.1 APRENDIZAGEM DE MÁQUINA

De acordo com Mitchell (1997) um programa aprende, a partir de uma experiência *E* em respeito a uma classe de tarefas *T* e uma medida de desempenho *P*, se seu desempenho nas tarefas *T*, segundo a medida *P*, melhora com a experiência *E*.

Um exemplo de programa que se comporta dessa maneira pode ser um sistema capaz de identificar dentre um conjunto de e-mails, quais são spams ou não. Todos os dias, diversos e-mails de diferentes assuntos chegam à nossa caixa de entrada, entretanto, cabe a nós verificar quais deles são úteis. De maneira geral, um problema computacional como este pode ser solucionado com a utilização de um algoritmo de inferência indutiva.

A inferência indutiva ou indução é um conceito lógico que permite se obter conclusões genéricas a partir de um conjunto particular de exemplos (ANGLUIN; SMITH, 1983). Por exemplo, o objetivo principal da inferência indutiva no caso dos e-mails, é aprender uma solução que seja capaz de determinar corretamente a classe de um determinado e-mail, ou seja classificá-lo como spam ou não spam.

Os paradigmas que norteiam o aprendizado indutivo podem ser divididos basicamente em dois grupos: supervisionado e não-supervisionado (RUSSELL; NORVING, 1995).

Aprendizado supervisionado: é fornecido ao algoritmo de aprendizagem um conjunto de dados de treinamento nos quais a classe associada a cada dado é

conhecida por um supervisor externo. Para as classes que possuírem valores de dados discretos, o problema é chamado de classificação e para as classes de valores contínuos, o problema é chamado de regressão.

Aprendizado não-supervisionado: neste tipo de abordagem, as classes do conjunto de treinamento não são conhecidas. O algoritmo analisa os dados disponíveis e utilizando medidas probabilísticas, realiza um agrupamento dos dados com características similares. A Figura 2 a seguir, apresenta, de maneira geral, os tipos de aprendizagem de máquina existentes.

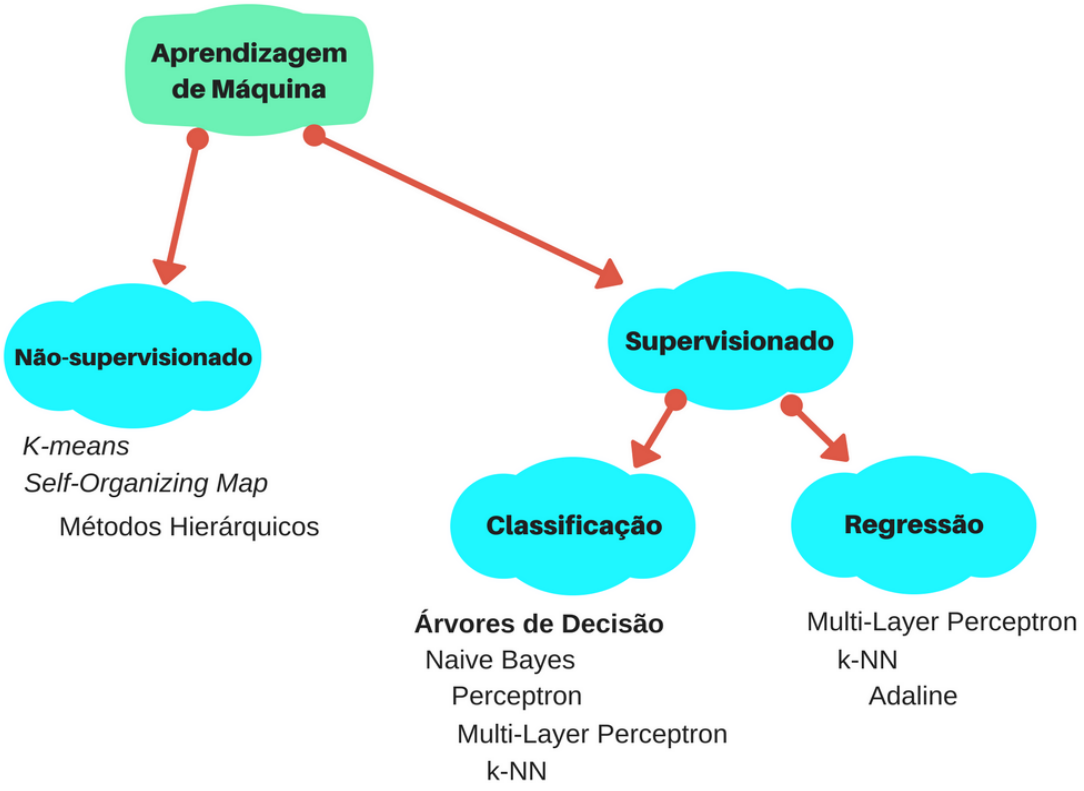


Figura 2: Tipos de aprendizagem de máquina.

Fonte: Autoria Própria

2.1.1 Conceitos e Definições

Nesta Seção serão abordados alguns dos principais conceitos existentes na área da aprendizagem de máquina e tomada de decisão.

Conjunto de exemplos: é composto por informações como: atributos, incluindo o atributo classe, e os valores possíveis para cada atributo. Além disso, um conjunto de exemplos deve possuir uma diversidade de instâncias possuindo: valor para cada um dos atributos e um valor de classe associado como resposta.

Classificador ou Hipótese: recebendo como entrada um conjunto de exemplos de treinamento, um indutor obtém como saída um classificador, de maneira que, dado uma nova instância de teste, o classificador saiba predizer, com grande precisão, a qual classe o exemplo pertence.

Ruído: imperfeições encontradas nos conjuntos de exemplos, sendo que as mesmas podem ser geradas a partir de diversos fatores. Por exemplo, o processo que gerou o conjunto, a maneira como o mesmo foi construído, a forma como os exemplos foram obtidos, erros na rotulação das classes, ou até mesmo erros semânticos cometidos pelo próprio supervisor dos dados.

Valores ausentes: valores que se encontram fora do escopo da aplicação. Tais valores podem ser desconhecidos, não-registrados, irrelevantes, etc., e podem ser ocasionados por razões como: mau funcionamento do equipamento que coletou os dados, mudanças nas definições do experimento, incapacidade de mensuração e avaliação.

Taxa de Erro de um classificador: também conhecida como taxa de classificação incorreta (**err(h)**), compara a classe real de cada exemplo com o rótulo atribuído pelo classificador induzido. Será retornado o valor 1 pelo operador || *E* ||, caso a expressão *E* seja verdadeira ($(y_j \neq h(x_j))$ = verdadeiro). Caso contrário, será retornado zero (0). *n* representa o total de exemplos (REZENDE, 2003).

$$\mathrm{err}(h) = \frac{1}{n}\sum_{i=1}^{n} || y_i \neq h(x_i) ||$$

Precisão do Classificador: complemento da taxa de erro denotado por acc(h).

$$\mathrm{acc}(h) = 1 - \mathrm{err}(h)$$

As fases do processo de aprendizagem de máquina são divididas em: treinamento e classificação.

Treinamento: são apresentados os exemplos de treinamento, obtidos do conjunto de treinamento. A partir disso, o sistema aprende baseado em todas as instâncias de treinamento.

Classificação: novos exemplos de teste são apresentados ao sistema. A partir do processo de aprendizagem, o mesmo deve realizar uma generalização e predizer a classe do exemplo.

Generalização: o processo de aprendizagem não deve ser realizado com base na memorização, mas sim por meio de um método que seja capaz de encontrar um fator determinante para a criação do classificador. No entanto, existem muitas formas de generalização, tornando-se assim um processo fundamentalmente mais difícil.

***Overfitting* e *Underfitting*:** *overfitting* é um problema que ocorre em um classificador, no qual a sua complexidade é muito elevada, mas seu poder de generalização é muito baixo. Quando um algoritmo testado com base em seu conjunto de treinamento apresenta bons resultados, mas quando o mesmo é submetido ao conjunto de testes e seu desempenho é inferior, estamos diante de um caso de *overfitting*. Analogamente, quando um algoritmo é testado sobre o conjunto de treino e teste, apresenta taxas de erro significantes e parecidas entre si, estamos diante de um caso de *underfitting*. Dessa forma, ainda é possível melhorar sua classificação, no entanto, deveremos mexer em alguns parâmetros do algoritmo.

2.1.2 Árvore de Decisão

O campo da aprendizagem de máquina, conforme dito anteriormente, possui diversos métodos de aprendizagem, particularmente neste trabalho tratamos do aprendizado supervisionado para a indução de árvores de decisão.

Uma árvore de decisão é uma estrutura hierárquica composta por nós e links. O primeiro nó é chamado raiz, e por convenção, se localiza no topo da árvore. A partir do nó raiz, se estendem ramificações sucessivas denominadas links ou ramos, que por sua vez, conectam os demais nós. Os últimos nós da árvore são denominados terminais ou folhas e representam uma decisão a ser tomada (MITCHELL, 1997).

Na estrutura dessa abordagem, cada nó da árvore especifica testes que devem ser realizados sobre algum atributo da instância de dados, e cada ramo descendente a partir desse nó, corresponde a um dos possíveis valores que este atributo pode assumir (QUINLAN 1986). Para classificar uma nova instância de um problema, a árvore é percorrida da raiz para as folhas de acordo com os valores dos atributos testados em nós sucessivos e, quando uma folha é alcançada a instância é classificada de acordo com a classe que rotula a folha (WITTEN & EIBE, 2005).

Os algoritmos de aprendizagem de árvores de decisão utilizam a estratégia dividir para conquistar, de maneira que um problema mais complexo é decomposto em subproblemas mais simples e, dessa maneira, recursivamente, a mesma estratégia é aplicada para cada subproblema estruturando assim a árvore. Os atributos de maior relevância vão sendo inseridos na árvore de decisão a partir do nó raiz até os nós folha, conforme alguma medida estatística (ex.: ganho de informação) (MITCHELL, 1997).

O método para indução de árvores denominado *Top Down Induction Decision Tree* (*TDIDT*) define um processo recursivo para a construção de árvores. Dado um conjunto de exemplos de treinamento em um nó, declarar o nó como folha, ou encontrar outra maneira de dividir o conjunto em um novo subconjunto. Essa maneira de dividir o conjunto é uma das características que diferem entre si os algoritmos de indução de árvores. *TDIDT* é base para vários algoritmos de indução de árvores de decisão, dentre os mais populares podemos citar o ID3, C4.5 e CART (BREIMAN et al., 1984).

O ponto principal desses algoritmos está em definir um critério de escolha para o melhor atributo que particionará o conjunto de exemplos, a fim de se obter subconjuntos mais homogêneos a cada iteração do algoritmo. De acordo com Mitchell (1997), para obter árvores compactas e com maior capacidade de tomar decisões corretamente, deve-se escolher o teste sobre algum atributo que gere nós mais “puros”. Um nó é chamado “puro” quando possui todos seus exemplos de treinamento pertencentes a uma mesma classe. Por outro lado, um nó é denominado “impuro”, se possui seus exemplos de treinamento pertencentes a classes distintas. Para isso, definem-se índices para cálculo da impureza, tais como*, Misclassication impurity, Entropy impurity, Gini impurity e twoing.*

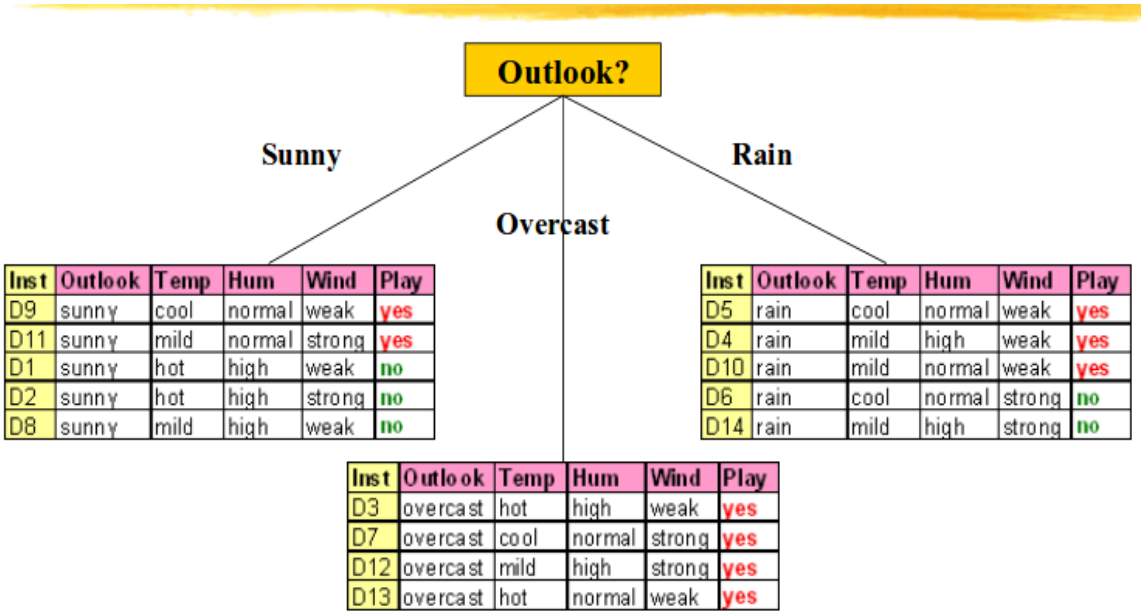
Um arcabouço para a construção de uma árvore de decisão a partir de um conjunto de treinamento P é descrito a seguir (PRATI, 2009):

1. Se S contém um ou mais exemplos, todos pertencentes à mesma classe, então a árvore de decisão para S é um nó folha rotulado por essa classe;

2. Se S contém exemplos que pertencem a diferentes classes então dividir S em subconjuntos mais “puros”;

3. Os passos 1 e 2 são aplicados recursivamente para cada subconjunto de exemplos de treinamento de maneira que, em cada nó, as arestas levam para as subárvores construídas a partir do subconjunto de exemplos S.

A Figura 3 a seguir, demonstra um exemplo de uma árvore de decisão construída a partir do conceito de pureza mostrado anteriormente.



Inst	Outlook	Temp	Hum	Wind	Play
D9	sunny	cool	normal	weak	yes
D11	sunny	mild	normal	strong	yes
D1	sunny	hot	high	weak	no
D2	sunny	hot	high	strong	no
D8	sunny	mild	high	weak	no

Inst	Outlook	Temp	Hum	Wind	Play
D3	overcast	hot	high	weak	yes
D7	overcast	cool	normal	strong	yes
D12	overcast	mild	high	strong	yes
D13	overcast	hot	normal	weak	yes

Inst	Outlook	Temp	Hum	Wind	Play
D5	rain	cool	normal	weak	yes
D4	rain	mild	high	weak	yes
D10	rain	mild	normal	weak	yes
D6	rain	cool	normal	strong	no
D14	rain	mild	high	strong	no

Figura 3: Exemplo de árvore de decisão construída a partir do conceito de pureza.

Para escolher os atributos que melhor separam os exemplos de treinamento é necessário a utilização de alguma medida estatística. Dessa forma é definida uma propriedade estatística denominada ganho de informação, que mede como um determinado atributo separa os exemplos de treinamento de acordo com sua classificação. Algoritmos como o ID3 (QUINLAN, 1986) e C4.5 (QUINLAN, 1993), usam esta medida.

O cálculo do ganho de informação depende, primeiramente, do cálculo de uma outra medida chamada entropia. A entropia, no contexto da indução de árvores, indica a homogeneidade dos exemplos do conjunto de treinamento (MITCHELL, 1997). Dado um conjunto S, mostrado na Tabela 1, de dados contendo exemplos positivos e negativos, a entropia relativa a este conjunto é definida por:

$$\text{Entropia} = -\text{p} \oplus \log_2 \oplus - \text{p} \ominus \log_2 \ominus$$

Onde:

$p \oplus$ é a proporção de exemplos positivos em S;

$p \ominus$ é a proporção de exemplos negativos em S.

Para o cálculo de entropia, toda expressão 0 log 0 é definida como sendo 0. A entropia será nula se todos os exemplos de dados contidos em S pertencerem à uma mesma classe. Dessa forma, se todos os membros de S são positivos ($p \oplus = 1$), então por consequência $p \ominus = 0$, implicando em Entropia (S) = -1 $\log_2(1)$ - 0 $\log_2 0$ = -1. 0 - 0 $\log_2(0)$ = 0.

Após definida a medida de impureza, neste caso a entropia, pode-se então definir o ganho de informação para medir a efetividade de classificação de um determinado atributo. O ganho de informação de um atributo A, relativo a uma coleção de dados S é definida como:

$$\text{Ganho}(\text{S}, \text{A}) = \text{Entropia}(\text{S}) - \sum_{\text{v} \in \text{Valores(A)}} \frac{|\text{S}_\text{v}|}{|\text{S}|} \text{Entropia}(\text{S}_\text{v})$$

Onde:
Valores (A) é o conjunto de todos os valores possíveis para o atributo A,
Sv é o subconjunto de S no qual o atributo A possui valor v.

Exemplificando, assuma que S é a coleção de exemplos da Tabela 1. Este conjunto contém 14 exemplos, dos quais 9 são positivos e 5 negativos, representados respectivamente por [9+, 5-]. Considere o cálculo do ganho de informação para o atributo Vento, onde S = [9+, 5-], $S_{fraco} \leftarrow$ [6+, 2-] e $S_{forte} \leftarrow$ [3+, 3-]:

$$\text{Ganho}(\text{S}, \text{A}) = \text{Entropia}(\text{S}) - \sum_{\text{v} \in (\text{Fraco,Forte})} \frac{|\text{S}_\text{v}|}{|\text{S}|} \text{Entropia}(\text{S}_\text{v})$$

= Entropia(S) – (8/14) Entropia(S_{Fraco}) - (6/l4) Entropia(S_{Forte})
= 0.940 - (8/14)0.811 - (6/14)1.00
= 0,048

O mesmo procedimento é aplicado aos atributos “Tempo”, “Umidade” e “Clima”, que possuem respectivamente, $Ganho(S, \text{Tempo}) = 0.24$, $\text{Ganho}(S, \text{Umidade}) = 0.151$, $\text{Ganho}(S, \text{Clima}) = 0.029$. Na primeira iteração do algoritmo, o atributo “Tempo” é escolhido como o atributo que melhor particiona o

conjunto S, portanto é definido como nó raiz da árvore. Um ramo para cada possível valor do atributo “tempo” é anexado ao nó (ex.: ensolarado, nublado e chuvoso).

O processo de selecionar um atributo e dividir os exemplos de treinamento repete-se a cada nó descendente não folha. Os atributos que já estiverem incorporados na árvore (ex.: “Tempo”) são excluídos do cálculo, de modo que aparecem apenas uma vez na árvore. O processo continua para cada novo nó descendente até que uma das duas condições seja atingida: todos os atributos foram incluídos na árvore ou os exemplos de treinamento associados a este nó possuem o mesmo valor de atributo (ex.: entropia é zero). O resultado final do algoritmo que constrói a árvore de decisão pode ser visto na Figura 4.

Após a geração da árvore, a primeira pergunta a ser feita ao jogador poderia ser se o tempo está ensolarado, chuvoso ou nublado. A seguir, dependendo da resposta, outras perguntas podem ser feitas até que se chegue a uma conclusão, obtida em um nó terminal.

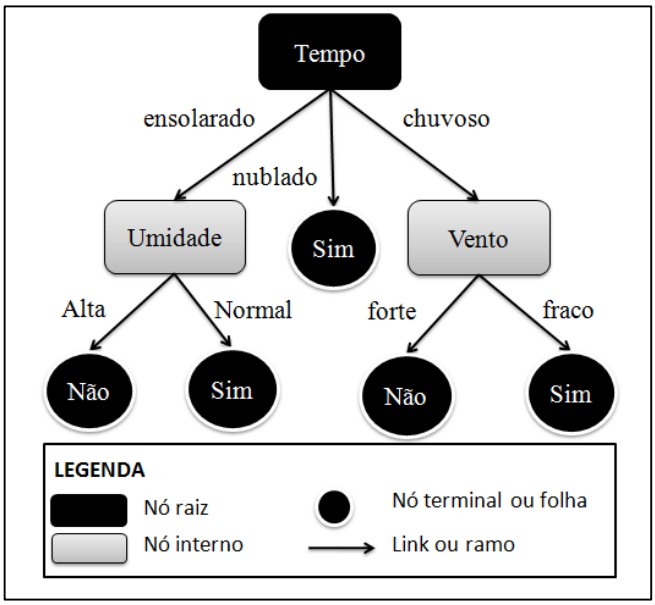


Figura 4: Árvore de decisão construída a partir do ID3.

Fonte: STANGE, 2011.

CONJUNTO DE TREINAMENTO					
Identificador da instância	**Atributos**				
	Tempo	**Clima**	**Umidade**	**Vento**	**Jogar Tênis**
1	ensolarado	quente	alta	fraco	não
2	ensolarado	quente	alta	forte	não
3	nublado	quente	alta	fraco	sim
4	chuvoso	regular	alta	fraco	sim
5	chuvoso	frio	normal	fraco	sim
6	chuvoso	frio	normal	forte	não
7	nublado	frio	normal	fraco	sim
8	ensolarado	frio	normal	fraco	não
9	ensolarado	frio	normal	fraco	sim
10	chuvoso	regular	normal	forte	sim
11	ensolarado	regular	normal	forte	sim
12	nublado	regular	alta	forte	sim
13	nublado	quente	normal	fraco	sim
14	chuvoso	regular	alta	forte	não

Tabela 1: Conjunto de dados de treinamento.

Fonte: Autoria própria.

2.2 TECNOLOGIA ADAPTATIVA

A concepção de programas automodificáveis é bastante antiga, sendo ela advinda da programação dos primeiros computadores e perdurando até os dias atuais. A ideia da adaptatividade, tratada neste trabalho, é um conceito razoavelmente novo, introduzida por pesquisadores do LTA[1] – Laboratório de Linguagens e Técnicas Adaptativas da Escola Politécnica da Universidade de São Paulo.

O LTA foi criado em 1985, sendo impulsionado pelo interesse pedagógico no desenvolvimento de novas tecnologias e pela criação de linguagens de alto nível. Em 1993, as pesquisas realizadas pelo grupo até então, se consolidaram com a introdução do conceito de autômato adaptativo, publicado em Neto (1994). Esse

[1] Site do LTA - http://www.pcs.usp.br/~lta

formalismo representou um marco nas pesquisas em adaptatividade, isso porque foi provado em (ROCHA, 2001) ser equivalente, em poder computacional, às máquinas de Turing.

A partir daí, uma das ideias centrais no formalismo adaptativo é que dispositivos mais expressivos, podem ser obtidos a partir de uma pequena modificação nos recursos oferecidos por um dispositivo mais simples (PISTORI, 2003). Basicamente, qualquer dispositivo que tenha seu comportamento determinado por conjuntos variáveis de regras (NETO, 2011), pode ser acrescido de uma camada adaptativa.

Atualmente, diversos são os dispositivos automodificáveis em uso, podemos citar como exemplo, além das Árvores de Decisão Adaptativas e dos Autômatos Adaptativos, as Redes de Markov Adaptativas (BASSETO, 2000), as Gramáticas Adaptativas (IWAI, 2000) as Redes de Petri Adaptativas (GOMES; CAMOLESI, 2008) e as Tabelas de Decisão (TCHEMRA, 2009).

Todo o arcabouço teórico produzido por pesquisadores do LTA, resultaram em diversos trabalhos em diferentes áreas de pesquisa tais como: processamento da língua natural (NETO; MORAES, 2003), representação do conhecimento e aprendizagem (SILVA; INOJOSA; 2011), aplicações educacionais (BALDI et al, 2016), métodos de diálogos em sistemas computacionais (ALFENAS, 2014) entre outros.

2.2.1 Formulação dos Dispositivos Adaptativos

Os dispositivos guiados por regras são definidos pela característica de possuírem sua representação e especificação guiada por um conjunto finito de regras. Para todo dispositivo que possua esta característica, tais regras comandam seu funcionamento, além de definir como o mesmo será estruturado ao longo de sua execução.

O funcionamento de um dispositivo guiado por regras é iniciado em uma dada configuração. A partir daí, são aplicadas, sucessivamente, cada uma das regras do conjunto de regras. Por meio da leitura de cada estímulo de entrada ou aplicação de uma regra, deve-se gerar um valor como resposta. As aplicações das regras ocorrerão até o ponto em que não existam mais regras ou estímulos a serem lidos, ou até o

momento em que os mesmos não possam ser mais ser aplicados (PISTORI; NETO, 2011).

Um dispositivo guiado por regras adaptativo, por sua vez, passa a ter seu conjunto de regras variável durante sua leitura. Todavia, essa capacidade é conferida a uma outra camada, denominada camada adaptativa. Tal camada funciona sobre o conjunto de regras original, modificando-o através de inserções e remoções de regras. Dessa forma, temos que o dispositivo guiado por regras adaptativo, ou simplesmente dispositivo adaptativo, é composto por duas camadas. A primeira delas, denominada subjacente, é composta por um dispositivo guiado por regras tradicional (árvores de decisão, autômato de pilha estruturado, etc.). A segunda camada é a adaptativa, a qual confere à camada subjacente a propriedade automodificadora. Essa segunda camada é capaz de transformar um dispositivo não adaptativo em um que seja capaz de realizar mudanças em sua estrutura durante a operação (PISTORI, 2003).

Um dispositivo adaptativo é formulado a partir de uma dupla DA = (CS, CA), onde CS representa um dispositivo guiado por regras, não adaptativo (camada subjacente) e CA representa a camada adaptativa.

De acordo com Neto (2009), a descrição de um dispositivo não adaptativo guiado por regras, pode ser dada por meio de uma quíntupla de seguinte formulação:

$$CS = (C, R, S, c_0, A)$$

Onde:

C é o conjunto de todas as possíveis configurações para o dispositivo.

R é uma relação de mudança de configuração para CS: $R \subseteq C \times (S \cup \{ \varepsilon \}) \times C$.

S corresponde ao conjunto fixo e finito de eventos dados como estímulos válidos de entrada para CS.

$\mathbf{c_0} \in C$ representa a configuração inicial do dispositivo.

A $\subseteq$ C representa o conjunto de todas as configurações de aceitação do dispositivo.

A camada adaptativa (CA) é definida por:

$$CA = (B, Z)$$

Na qual:

B é um conjunto de ações adaptativas, que contém também o valor nulo, o qual representa as ações que não causam modificações na camada subjacente.
Z: R → B² é a representação de uma função que mapeia cada regra da camada subjacente em um par ordenado de ações adaptativas.

Esta camada adaptativa é representada por um conjunto de funções que são acopladas às regras da camada não adaptativa. Tais funções são capazes de realizar eventuais alterações no conjunto de regras do dispositivo subjacente implicando em mudanças no comportamento do dispositivo adaptativo (STANGE, 2011). Dessa maneira, a formulação para tal dispositivo pode ser descrita por uma sêxtupla (NETO, 2009) da seguinte forma:

$$AD = (C, RA, S, AA, c_0, RA_0, A), \text{ onde:}$$

C é o conjunto de todas as possíveis configurações para o dispositivo adaptativo.
RA é o conjunto de regras adaptativas de AD.
S corresponde ao conjunto de estímulos válidos de entrada para AD.
$\mathbf{C_0} \in C$ representa a configuração inicial e única do dispositivo adaptativo.
$\mathbf{RA_0}$ é o conjunto de regras iniciais, fixas e adaptativas de AD.
$\mathbf{A} \subseteq C$ é o conjunto de todas as configurações de aceitação de AD.
AA é o conjunto de funções adaptativas de AD.

O objetivo das funções adaptativas é definir quais mudanças serão realizadas na camada subjacente quando uma determinada ação adaptativa for acionada. Dessa maneira, pode-se dizer que uma ação adaptativa é definida pela chamada de uma função adaptativa. Uma função adaptativa possui como característica principal uma lista de ações adaptativas sendo elas: ações de consulta, inserção e remoção de regras do conjunto que permeia a camada subjacente (PISTORI, 2003). Formalmente uma função adaptativa é definida por um 9-upla:

$$FA = (F, P, V, G, C, R, I, A, B), \text{ na qual:}$$

F representa o nome da função adaptativa.
P é a ênupla (p1, p2, p3, …, pp) de parâmetros formais.

V é a lista (v1, v2, v3, …, vv) de variáveis. Tais variáveis têm seus valores preenchidos uma única vez através das ações adaptativas de consulta.
G é a lista (g1, g2, g3, …, gg) de geradores. Os geradores têm seus valores preenchidos a cada nova chamada da função adaptativa.
C é a lista das ações de consulta.
R representa a lista de ações de remoção.
I representa as ações de inserção.
A é uma função adaptativa inicial que pode, opcionalmente ser executada antes de F.
B é uma função adaptativa final que pode, opcionalmente ser executada depois de F.

2.2.2 Árvores de Decisão Adaptativas

Pistori (2003) apresenta um novo enfoque para algoritmos de indução árvores de decisão, baseada na teoria dos dispositivos adaptativos. Este enfoque apresenta um dispositivo adaptativo denominado Árvore de Decisão Adaptativa, cujo mecanismo subjacente é Árvore de Decisão Não-Determinística (Árvore DND). Uma árvore de decisão não-determinística é uma árvore de decisão (QUINLAN , 1986).

Em árvores de decisão convencionais, uma decisão é obtida quando é estabelecido um caminho completo da raiz para algum nó folha. Neste caso, quando algum valor de atributo é ausente ou inconsistente ocorre a interrupção do processo, sem que qualquer decisão seja obtida. Ao contrário das árvores de decisão tradicionais, com as árvores DND, a busca de um caminho que liga a raiz da árvore a alguma de suas folhas pode continuar, indeterministicamente, mesmo na ausência ou inconsistência de valores para determinados atributos (PISTOIR, 2003).

A fim de exemplificar a estrutura das árvores-DND, a Figura 5 apresenta uma árvore de decisão com objetivo de determinar uma ação para um casal (sair ou não-sair) em um determinado dia da semana, baseado nos atributos tempo (chuvoso ou ensolarado) e clima (frio, quente, ameno ou úmido). De acordo com a árvore, se o tempo para um determinado dia for desconhecido e o clima úmido, o processo de decisão seria interrompido, pois não existe um caminho correspondente a esta situação na árvore. No entanto, com a utilização da árvore de decisão não-determinística, a árvore de decisão pode fornecer mais de uma resposta para a

mesma questão. Neste caso, a resposta final pode ser uma distribuição probabilística estimada através das ocorrências das diferentes respostas.

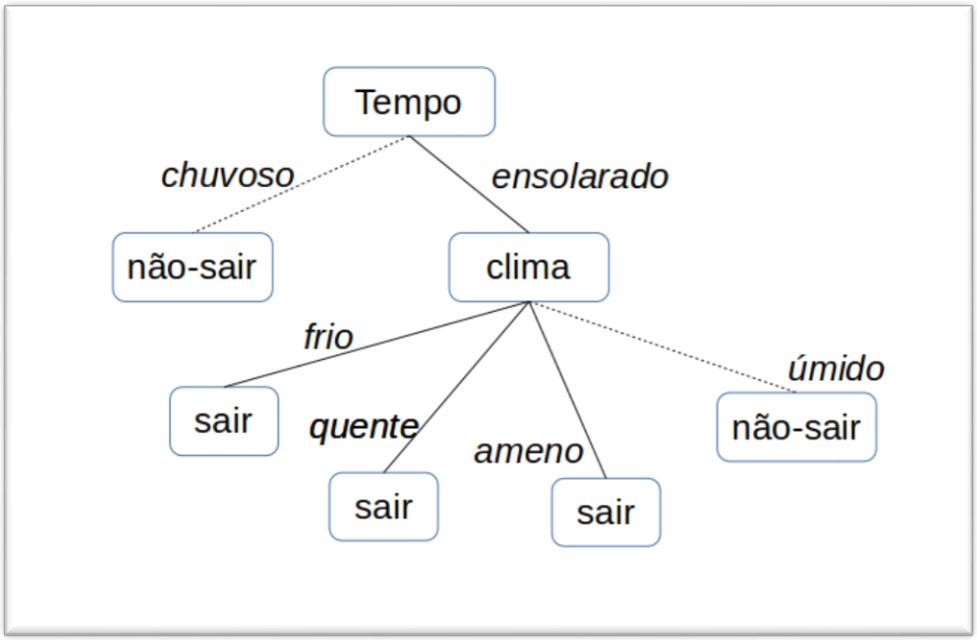


Figura 5: Árvore de decisão para a ação *sair* ou *não sair*.
Fonte: Autoria própria.

A formulação de uma árvore-DND é dada através de uma 7-upla T = (I, Σ, Γ, f, c, A, S) (PISTORI, 2003) na qual:

I, Σ, Γ são conjuntos não-vazios representando, respectivamente, exemplos, atributos e valores. Σ e Γ são conjuntos finitos ao contrário de I que é contavelmente infinito. O conjunto Γ possui o valor "?" (valor ausente) que tem como objetivo representar valores ausentes, desconhecidos ou inconsistentes.
f : Σ → 2 Γ é uma função que mapeia cada atributo contido em Σ em um conjunto de valores possíveis para esse atributo, representando assim o domínio de cada atributo.
c ∈ Σ representa o atributo classe o qual têm seus valores mostrados nas folhas de uma árvore de decisão.
A: I × Σ → Γ é uma função binária que têm como objetivo descrever cada elemento do conjunto de exemplos, criando uma associação entre exemplos, atributos e valores.

S é uma estrutura hierárquica finita, denominada sub-árvore, e definida recursivamente como:
Folha: uma dupla (id, v), onde $v \in f(c)$ representa um valor para o atributo classe, e id é um identificador único.
Não Folha: Uma (n+2)-upla (id, a, (v_1, S_1), ..., (v_n , S_n)), na qual id é um identificador, $a \in \Sigma - \{c\}$ é um atributo, $v_i \in f(a)$, $1 \leq i \leq n$ são valores e todos os elementos S_i , $1 \leq i \leq n$ são sub-árvores.

A árvore de decisão apresentada na Figura 5 pode ser formalmente representada por T = (I, Σ, Γ, f, c, A, S) onde:

I é um conjunto de exemplos de dias da semana e decisões relacionadas a dias, por exemplo: {segunda, terça, decisão1, decisão2}.

Σ = {tempo, clima, ação}

Γ = {chuvoso, ensolarado, frio, quente, ameno, úmido, sair, não-sair, ?}

f é definido através da tabela:

Σ	**f**
tempo	{chuvoso, ensolarado, ?}
clima	{frio, quente, ameno, úmido, ?}
ação	{sair, não-sair, ?}

c = {ação}

A é definida pela seguinte tabela:

I	tempo	clima	ação
segunda	chuvoso	frio	não-sair
terça	ensolarado	quente	sair
decisão1	?	ameno	?
decisão2	ensolarado	úmido	?

- **S** é dada por:
  - (S_0, tempo, (chuvoso, S1), (ensolarado, S2))
  - (S_1, não-sair)
  - (S_2, clima, (frio, S_3), (quente, S_4), (ameno, S_5), (úmido, (S_6, não-sair)))
  - (S_3, sair)
  - (S_4, sair)
  - (S_5, sair)

Através do identificador (id) é possível referenciar toda a sub-árvore. No exemplo acima, a sub-árvore S_6 foi alocada dentro de R_2, demonstrando que é possível descrever uma sub-árvore dentro de outra.

Com a utilização da tecnologia adaptativa na estrutura da árvore de decisão apresentada anteriormente, obtêm-se um dispositivo guiado por regras adaptativo denominado Árvore de Decisão Não-Determinística Adaptativa, ou Árvore-DND Adaptativa. A camada adaptativa da Árvore-DND Adaptativa permite que a estrutura S da árvore seja modificada a partir de um conjunto de ações adaptativas, especificadas pelas funções adaptativas (PISTORI, 2003).

No caso das árvores-DND adaptativas, as ações adaptativas podem ser representadas através de sub-árvores genéricas, cercadas de parênteses e precedidas pelos símbolos que correspondem ao tipo da ação elementar: “+” para inserção, “-” para remoção e “?” para consulta. Para completar e facilitar a especificação da camada adaptativa, também utiliza-se o símbolo ɾ para denotar o exemplo corrente no conjunto de exemplos de treinamento. Dessa forma, um exemplo de uma ação adaptativa de consulta pode ser dado da seguinte maneira: *?[((ɾ, ?atributo), ?valor)].* Substituindo-se *?atributo* pelo nome do atributo, obtêm-se os valores para o atributo especificado (PISTORI, 2003).

De maneira geral, a formulação de uma árvore de decisão adaptativa é dada pela mesma representação do dispositivo adaptativo já apresentada anteriormente, através de uma dupla ADT = (CS, CA). Entretanto, neste caso o mecanismo subjacente CS é representado por uma árvore-DND, onde CS = (I, Σ, Γ, f, c, A, S). A camada adaptativa CA é constituída pelo conjunto de ações adaptativas já definidas anteriormente.

A operação de uma árvore-DND adaptativa segue de maneira análoga ao mecanismo subjacente, até o momento em que uma chamada de função adaptativa ocorra. Da mesma forma, a estrutura de uma função adaptativa segue a definição geral das funções adaptativas definidas para os dispositivos adaptativos, no entanto, com as alterações propostas acima.

Para exemplificar a estrutura de uma árvore de decisão não-determinística adaptativa, utilizaremos a árvore-DND, que determina se um casal deve ou não sair de casa em um determinado dia da semana. Este exemplo estenderá a árvore anteriormente apresentada através de uma função adaptativa que acrescentará a ela uma nova regra: para dias da semana em que o tempo for chuvoso e o clima ameno, a ação para o casal neste caso será de sair, ao contrário do que a árvore sugeria originalmente.

A árvore-DND adaptativa que denota o caso exposto acima é dada por ADT = (CS, CA) em que o dispositivo subjacente CS é representado por T apresentada anteriorimente. No entanto, será acoplada uma ação adaptativa A_1 à sub-árvore S_1 de S e a mesma passará a ser: [A_1] (S_1, não-sair). A função adaptativa A_1 tem por objetivo substituir a sub-árvore S_1 que determina a ação não-sair, por uma outra sub-árvore um pouco mais complexa que, mesmo possuindo o atributo tempo com valor chuvoso, também leva em consideração o clima, alterando a ação para sair, se o mesmo possuir valor ameno. A função adaptativa A_1, descrita na Tabela 4, possui também uma função adaptativa A_2 que têm como finalidade remover a nova sub-árvore corrente e chamar a função A_3, que por sua vez, retorna a árvore à sua situação inicial. As Figuras 6 e 7 apresentam, respectivamente a árvore de decisão após a realização da função A_1 e após a realização das funções A_2 e A_3.

1	**A_1**	**Cabeçalho da função**
2	+ [(S7, clima, (ameno, (S8, sair) [A2]), (frio, (S9, não-sair) [A2]), (quente, (S10, não-sair) [A2]), (úmido, (S11, não-sair) [A2]))]	Ação elementar de inserção
1	A_2	Cabeçalho da função
2	- [(S7, clima), [A3]]	Ação elementar de remoção
1	A_3	Cabeçalho da função
2	+ [[A1] (S1, não-sair)]	Ação elementar de inserção

Tabela 2: Funções adaptativas do exemplo de Árvore-DND adaptativa.

Fonte: Autoria própria.

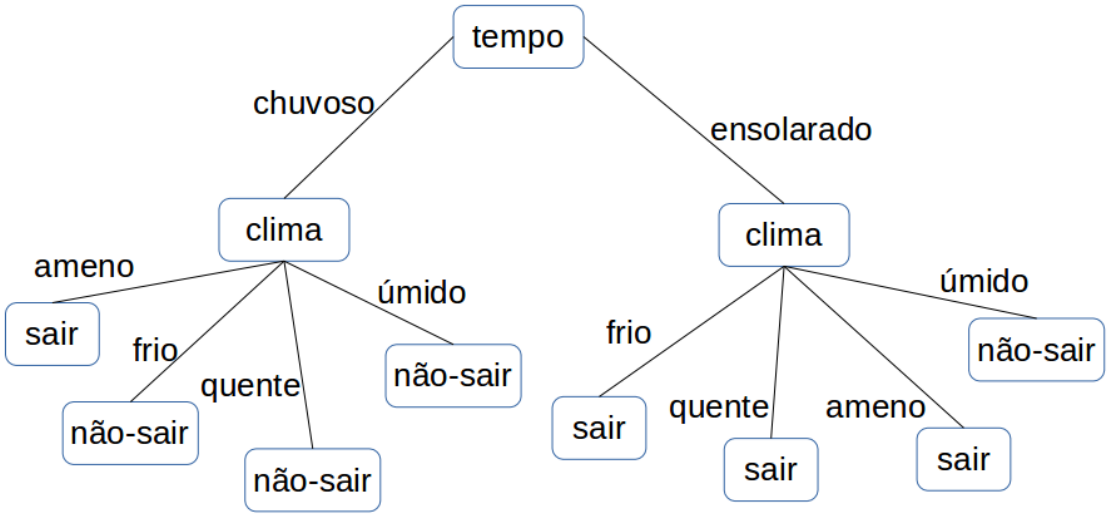


Figura 6: Árvore após a execução da função [A_1].

Fonte: Autoria própria.

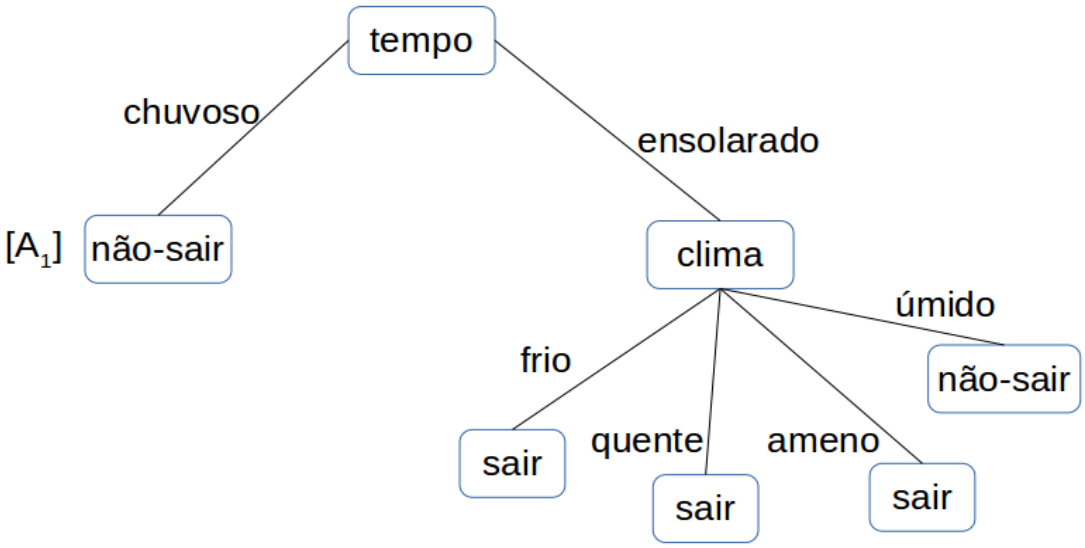


Figura 7: Árvore após a execução das funções [A2] e [A3].

Fonte: Autoria própria.

3 PROJETO E IMPLEMENTAÇÃO

O desenvolvimento do projeto está baseado na construção de um algoritmo para indução de árvores de decisão utilizando técnicas adaptativas, denominado *Adaptive*-IDT (*Adaptive Incremental Decision Tree*). Para isso, é utilizado um dispositivo adaptativo do tipo árvore-DND como camada subjacente. A camada adaptativa é composta basicamente de funções adaptativas, que permitem a modificação da estrutura da árvore de decisão através da inserção e remoção de sub-árvores. Este capítulo têm como objetivo descrever o algoritmo *Adaptive*-IDT, bem como seu funcionamento.

3.1 CARACTERÍSTICAS GERAIS ALGORITMO *ADAPTIVE*-IDT

O *Adaptive*-IDT (*Adaptive Incremental Decision Tree*) é um algoritmo de aprendizagem de máquina desenvolvido através da combinação de técnicas estatísticas e da tecnologia adaptativa. Este algoritmo foi baseado no *Adaptree*, proposto por (PISTORI; NETO, 2002), no qual a ideia geral é transformar cada exemplo de treinamento em um caminho completo partindo da raiz até a folha e assim, armazená-los. Diferentemente da maioria dos algoritmos de indução de árvores de decisão, tanto o *Adaptive*-IDT quanto o *Adaptree,* não buscam encontrar exatamente uma árvore mínima. Desse modo, a estrutura da árvore do *Adaptive*-IDT torna-se similar a uma árvore de prefixos, na qual cada nível da árvore corresponde a um atributo do conjunto de dados de treinamento e vice-versa.

Para um melhor entendimento do algoritmo faz-se necessário a compreensão de algumas particularidades a respeito do algoritmo. Assim, assume-se as seguintes características:

- **Aprendizagem supervisionada:** a tarefa de aprendizagem conta com um supervisor externo, o qual já possui conhecimento dos conjuntos de dados, tanto de treinamento quanto de teste, além de conhecer a saída (valor classe) associada a cada instância de teste. Baseado neste conhecimento prévio, o

supervisor pode realizar uma avaliação nas capacidades de predição do modelo (GAMA et al, 2015).

- **Aprendizagem incremental:** as instâncias de treinamento podem ser fornecidas separadamente, ao longo da execução do algoritmo. Isso permite intercalar exemplos de treinamento e teste durante a construção da árvore de decisão.

- **Crescimento da árvore:** à medida que os exemplos de treinamento vão sendo lidos, a estrutura da árvore cresce até uma altura **(h)** máxima igual ao número de atributos **(Σ)**. Dado que cada nível da árvore corresponde a um atributo no conjunto de dados, é possível inferir que o crescimento da árvore se concentrará mais em largura do que em altura.
- **Atributos ordenados:** utiliza uma função inicial que impõe uma ordem aos atributos, de acordo com a forma que estão dispostos no conjunto de treinamento. Esta ordem é mantida ao longo de toda a execução do algoritmo. Por convenção, o primeiro atributo a ser lido é alocado na raiz da árvore, os demais atributos são os nós internos e o último atributo é a classe.

- **Atributos categóricos:** os conjuntos de dados que podem ser lidos pelo *Adaptive*-IDT, devem possuir atributos categóricos, não sendo possível, a *priori*, o tratamento de atributos numéricos.

- **Capacidade de generalização:** a capacidade de generalização do algoritmo é determinada por meio das funções adaptativas, onde através da natureza não determinística da árvore–DND em conjunto um mecanismo estatístico torna-se capaz de generalizar além dos exemplos de treinamento.

- **Não determinismo**: através da utilização de medidas estatísticas, o algoritmo determina qual nível da árvore deve conter uma função adaptativa que, em

caso de não determinismo, permite que um novo caminho seja criado durante o processo de classificação.

- **Valores ausentes ou inconsistentes:** o *Adaptive*-IDT não aceita, a priori, a utilização de valores ausentes ou inconsistentes, sendo necessária um pré-processamento dos dados por parte do supervisor externo, por meio da remoção ou correção das inconsistências e/ou ausências de dados.

- **Mecanismo adaptativo:** A árvore de decisão adaptativa (árvore-DND adaptativa) do *Adaptive*-IDT, pode ser formalmente definida da seguinte forma (PISTORI, 2003): T = ((*I, Σ, Γ, f, c, A, S), Φ*), onde o conjunto de atributos Σ possui uma ordenação arbitrária podendo ser representada por uma sequência a_1, ..., a_j, na qual j = número total de atributos, c = a_j representa o atributo classe, a estrutura inicial S da árvore contém um valor inicial desconhecido ? na raiz, e a camada adaptativa Φ é composta por um conjunto de ações adaptativas $A_{1, ..., A_k}$, podendo estas serem ações de inserção, remoção ou consulta. Para o *Adaptive*-IDT, as funções adaptativas estão acopladas a alguns ramos específicos e, quando acionadas, possuem uma única ação elementar de consulta seguida de uma de inserção, especificadas na seção seguinte.

3.2 ESTRUTURA E FUNCIONAMENTO DO *ADAPTIVE*-IDT

Como a ideia do algoritmo baseia-se na leitura, processamento e armazenamento de cada instância de treinamento, será mostrado na presente seção, como o *Adaptive*-IDT transforma cada uma delas em um caminho completo da raiz até as folhas. A fim de facilitar a compreensão do algoritmo, seu funcionamento pode ser divido em três camadas distintas: entrada, processamento e classificação.

3.2.1 Camada de Entrada

Na entrada, o algoritmo realiza a leitura e um pré-processamento no conjunto de dados, executando uma função inicial que têm por objetivo obter os atributos para o algoritmo baseados na ordenação em que estão alocados no conjunto de dados.

Para o treinamento do algoritmo será considerado o conjunto de dados mostrado na Figura 8. Seguindo o exemplo do conjunto de dados apresentado, o atributo que ocuparia a raiz seria “tempo”, e o atributo classe seria “jogar”.

myTestTrain.arff

```
@relation myTestTrain.symbolic

@attribute tempo {ensolarado, nublado, chuvoso}
@attribute clima {quente, ameno, frio}
@attribute umidade {alta, normal}
@attribute vento {forte, fraco}
@attribute jogar {SIM, NAO}

@data
ensolarado, quente, alta, forte, NAO
ensolarado, ameno, alta, fraco, NAO
nublado, quente, alta, fraco, SIM
chuvoso, ameno, alta, fraco, SIM
chuvoso, frio, normal, fraco, SIM
chuvoso, frio, normal, forte, NAO
nublado, frio, normal, forte, SIM
ensolarado, quente, alta, fraco, NAO
ensolarado, frio, normal, fraco, SIM
chuvoso, ameno, normal, fraco, SIM
ensolarado, ameno, normal, forte, SIM
nublado, ameno, alta, forte, SIM
nublado, quente, normal, fraco, SIM
chuvoso, ameno, alta, forte, NAO

```

Figura 8: Conjunto de dados de treinamento.

Fonte: Autoria própria

3.2.2 Camada de Processamento

A próxima camada a ser executada pelo *Adaptive*-IDT é a de processamento. Em termos gerais, tal camada pode ser considerada a principal, pois é nela que a estrutura completa da árvore é construída. Basicamente, a camada de processamento recebe uma instância de treinamento obtida na camada de entrada e busca na estrutura da árvore um caminho que corresponda a instância que está sendo processada. Esta correspondência está baseada na comparação dos valores do atributo raiz e do atributo folha, ou seja, o algoritmo verifica se existe um caminho na estrutura S, cujo valor do atributo raiz e o valor do atributo folha sejam iguais aos valores da instância de treinamento. Essa comparação pode desencadear dois procedimentos distintos:

1. **Se não existir um caminho correspondente**, o algoritmo constrói um novo caminho.

2. **Se existir um caminho correspondente**, o algoritmo por meio do mecanismo estatístico, modifica convenientemente este caminho de forma a melhor generalizar a árvore.

Primeiramente, caso o *Adaptive*-IDT não encontre o caminho, a instância é processada e o mesmo é construído, de maneira que o valor do nó raiz e do nó folha assumem os valores contidos na instância de treinamento. Os valores de seus nós internos são denotados por um “?”, o qual representa um valor de atributo desconhecido. Além disso, aos ramos que possuem este símbolo também está acoplada uma função adaptativa A_k. As funções adaptativas possuem duas ações elementares sequenciais, sendo uma de consulta e outra de inserção, ambas apresentadas na Tabela 3.

1	A_k	Cabeçalho da Função
2	? [((r, ?atributo_corrente), ?valor)]	Ação elementar de consulta que obtém o valor do atributo corrente.
3	+ [(r*, ?valor)]	Ação elementar que insere o valor obtido em dois (2), ao nó atual.

Tabela 3: Funções Adaptativas do *Adaptive*-IDT.

Fonte: Autoria própria

Para ilustrar, considere o exemplo mostrado na Figura 9. Como é a primeira iteração do algoritmo, a estrutura S contém apenas um atributo desconhecido com valor ?, portanto, é necessário atualizar a estrutura da árvore construindo um novo caminho. A Figura 10 representa graficamente um caminho completo da árvore de decisão após o processamento da primeira instância de treinamento, este caminho corresponde a estrutura S atual.

```
10  ensolarado, quente, alta, forte, NAO
```

Figura 9: Primeira instância de treinamento.

Fonte: Autoria própria.

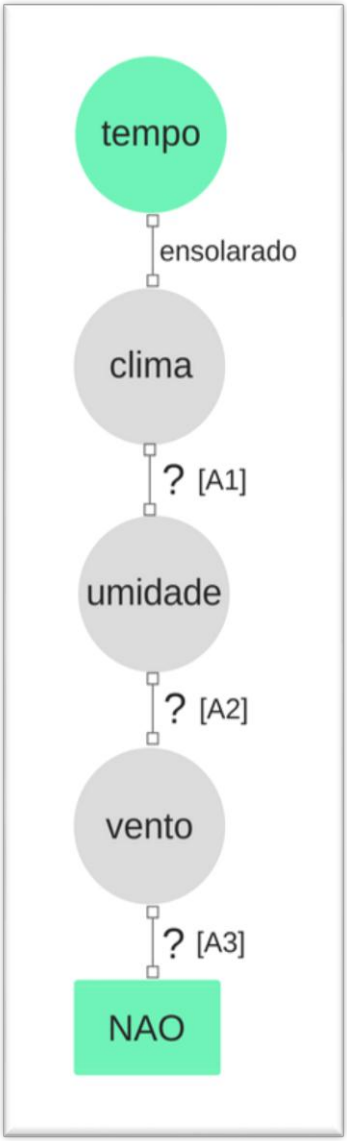


Figura 10: Primeiro caminho da árvore.

Fonte: Autoria própria.

Em outra situação, caso o algoritmo verifique que um caminho corresponde com o exemplo corrente, o *Adaptive-IDT* executa um mecanismo estatístico que busca o atributo de maior relevância. A busca por este atributo é baseada no cálculo da frequência relativa de cada valor lido em relação ao total de instâncias recebidas até o momento que possuam a configuração correspondente com o caminho. Dessa maneira, o atributo cujo valor possuir a maior frequência relativa, será o mais relevante.

Após executado o mecanismo estatístico, caso seja verificado que algum atributo passou a ser mais relevante que os demais, é realizada uma atualização no nó correspondente ao atributo verificado, substituindo-se o símbolo “?” e a função adaptativa A_K, pelo respectivo valor do atributo lido na instância de treinamento atual.

Para exemplificar, utilizaremos a segunda instância do conjunto de dados, mostrada a seguir na Figura 11.

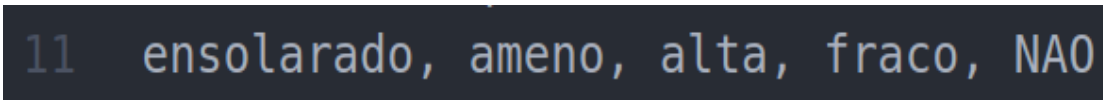


Figura 11: Segunda instância de treinamento.

Fonte: Autoria própria.

Uma vez realizada a leitura da segunda instância de treinamento, o *Adaptive*-IDT busca um caminho correspondente a ela, ou seja, um caminho cujo valor de raiz é “ensolarado” e o valor da classe é “NAO”. Como a primeira instância possui tais configurações, a busca pelo caminho retornará um resultado não nulo, passando assim, para a execução do mecanismo estatístico.

O mecanismo estatístico calcula para cada um dos valores possíveis dos nós internos, a frequência relativa em relação ao total de exemplos lidos até o presente momento com a configuração Tempo = “ensolarado” e Jogar Tênis = “NAO”. Sendo assim, visto que o total de exemplos lidos até o momento é dois (2), temos que a frequência relativa para os valores são as seguintes:

- *FR(clima = quente) = 0.5*,
- *FR(clima = ameno) = 0.5*,
- *FR(humidade alta) = 1.0*,
- *FR(vento = forte) = 0.5*,
- *FR(vento = fraco) = 0.5.*

Para os demais valores possíveis dos atributos, temos $FR(v_x) = 0.0$, uma vez que tais valores ainda não foram lidos pelo algoritmo. Dessa forma, torna-se fácil perceber que o atributo com maior relevância é “umidade”, visto que um dos seus

possíveis valores possui a maior frequência relativa dentre todas as calculadas. A Tabela 4, a seguir, sintetiza os dados obtidos a partir da leitura da segunda instância de treinamento.

Total de instâncias lidas até o momento:	2
Identificador da instância atual:	2
Atributo/Valor da raiz:	tempo: *ensolarado*
Atributo/Valor da folha:	jogar: *NAO*
Atributos/Valores possíveis:	clima: *quente, ameno, frio* umidade: *alta, normal* vento: *forte, fraco*
Valores lidos/Freq. Relativas:	quente: *0.5* ameno: *0.5* alta: *1.0* forte: *0.5* fraco: *0.5*
Valores possíveis restantes/Freq. Relativas:	frio: 0.0 normal: 0.0
Atributo com maior relevância:	umidade

Tabela 4: Informações à respeito dos exemplos lidos.

Fonte: Autoria própria.

O próximo passo da construção, após inferido o atributo de maior importância, é a atualização do valor do nó correspondente ao atributo umidade. A seguir, a Figura 12 representa o processo de atualização do nó, após a execução do mecanismo estatístico.

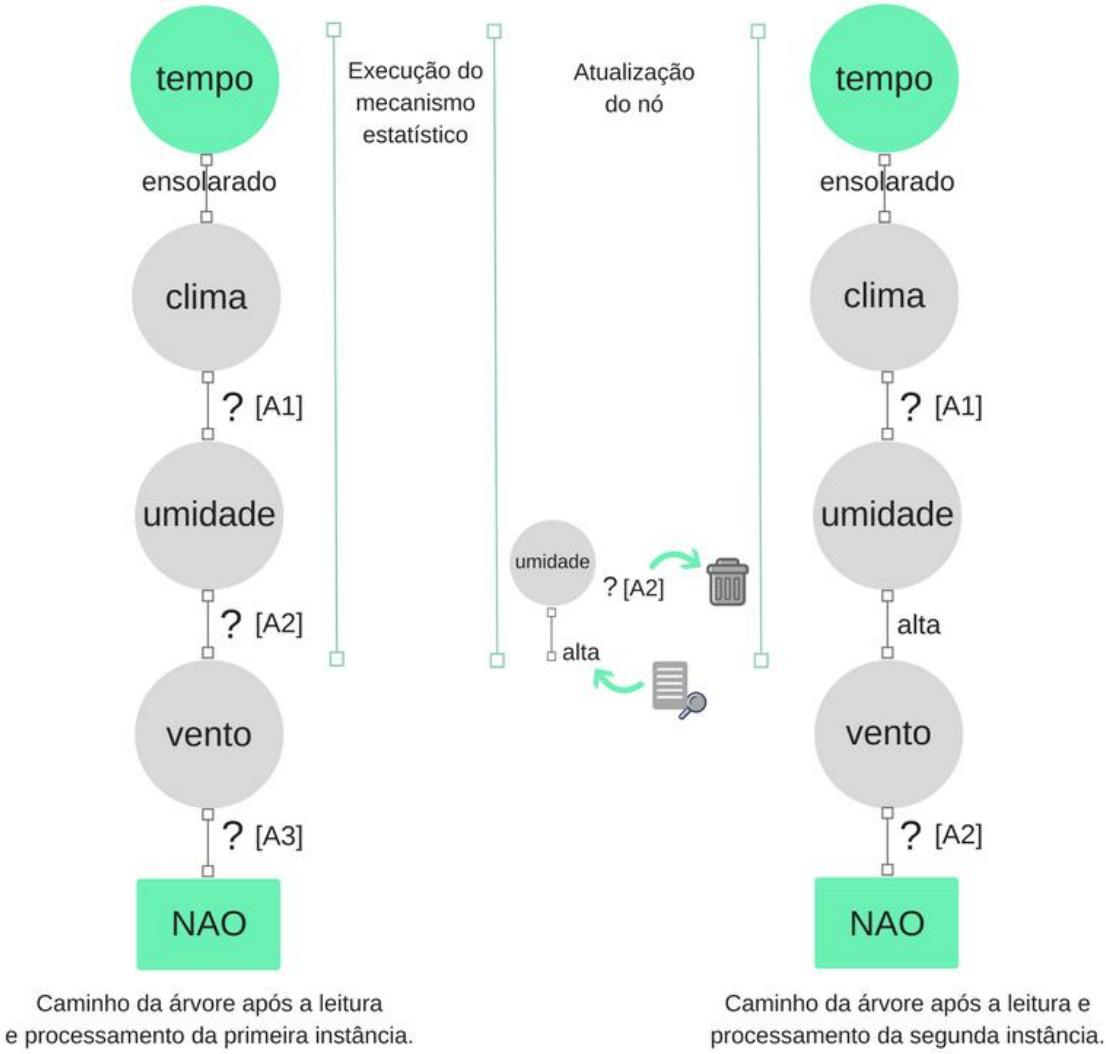


Figura 12: Processo de atualização de um nó.

Fonte: Autoria própria.

A Figura 12 ilustrou, de forma geral, como o *Adaptive*-IDT realiza o processo de atualização do nó. A partir da execução mecanismo estatístico, é obtido qual é atributo/ valor mais relevante, sendo possível localizar qual nó deve ter seu valor alterado. Dessa forma, durante o processo de classificação, a existência de tal valor no caminho da árvore pode ser um fator que determine a correta classificação do exemplo de teste.

Após lidas cada uma das instâncias, o algoritmo irá obter uma árvore de decisão completa, a qual é representada por um conjunto de caminhos armazenados. A árvore poderá ser utilizada para a classificação de novos exemplos, obtidos a partir

de um conjunto de testes. A Figura 13 a seguir, ilustra cada um dos caminhos armazenados após a leitura completa do conjunto de treinamento.

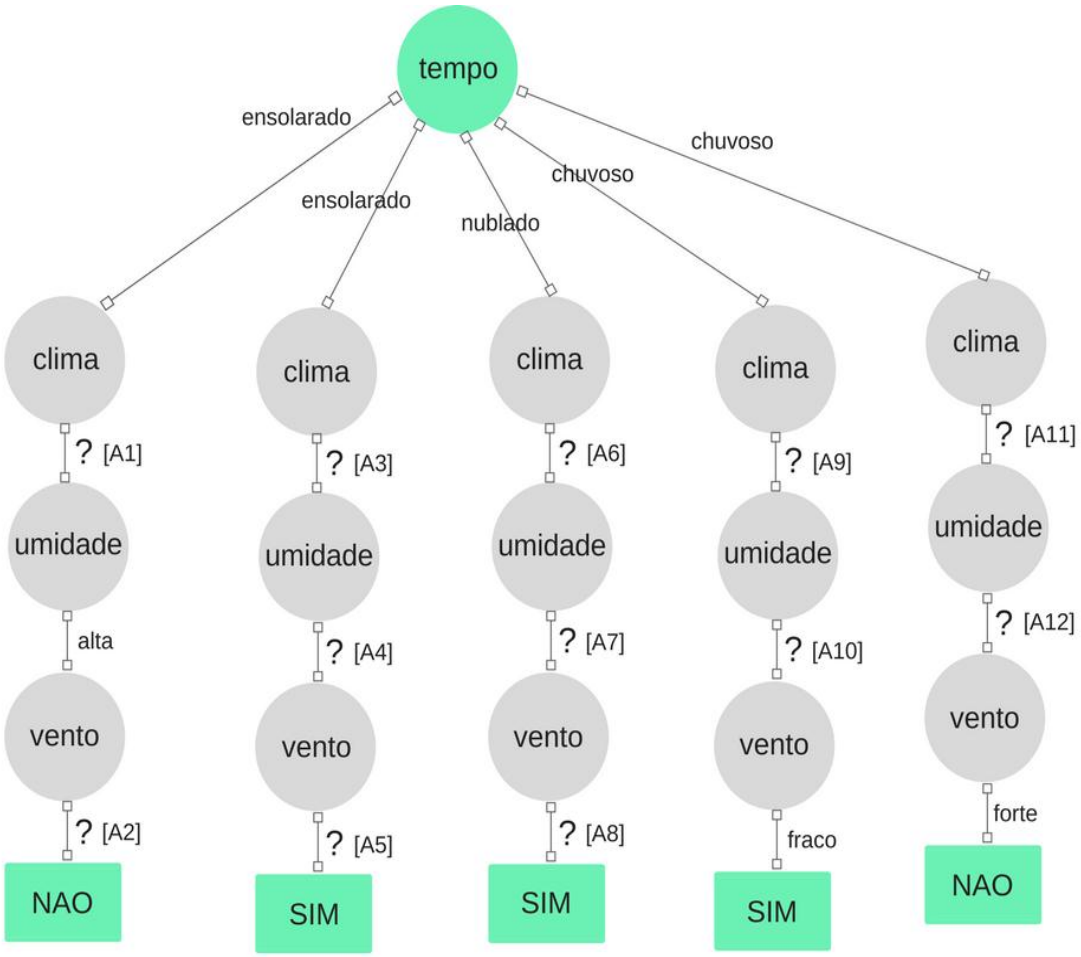


Figura 13: Caminhos da árvore gerada a partir do conjunto de treinamento.

Fonte: Autoria própria.

3.2.3 Camada de Classificação

A última camada a ser executada pelo *Adaptive*-IDT é a de classificação. Nela, novos exemplos de teste podem ser submetidos à árvore de decisão já processada, a fim de se obter, para cada um deles, uma resposta (classificação). Nesta etapa, um dos principais pontos abordados foi a utilização da tecnologia adaptativa. Por meio

dela, foi possível conferir à árvore a capacidade de automodificação durante a classificação, melhorando assim, a tomada de decisão.

O funcionamento desta etapa consiste basicamente em: dado uma nova instância para classificação realizar um busca em profundidade, percorrendo cada um dos caminhos da árvore até que se encontre um caminho capaz de classificar a nova instância. Por exemplo, digamos que uma instância de teste possua a seguinte configuração:

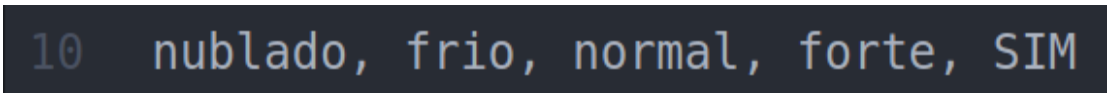


Figura 14: Exemplo de teste

Fonte: Autoria própria.

Se a mesma for submetida a um caminho cujo primeiro nó possua um valor “ensolarado”, é possível afirmar que tal caminho não é capaz de processar a instância fornecida e retornar uma resposta. Essa afirmação pode ser confirmada uma vez que o valor inicial da instância (“nublado”), é diferente do valor da raiz do caminho (“ensolarado”), sendo necessário assim, percorrer o próximo caminho da árvore.

Agora, supondo que o valor do nó seja “nublado”. Tal caminho pode ser um dos que classificará a instância, no entanto, para determinar isso com exatidão, é necessário percorrer cada um dos nós internos do mesmo e realizar esta verificação. Se todos os valores lidos do exemplo de teste coincidirem com os valores da raiz e nós internos do caminho, o algoritmo poderá retornar o valor contido em sua folha, ou seja, uma classificação para o exemplo. Caso algum valor não coincida durante este processo, a operação atual é finalizada e o próximo caminho deve ser processado. Este procedimento ocorrerá até que algum caminho retorne um valor como resposta. Se porventura nenhum caminho for encontrado, o algoritmo retornará o valor de classe que possuir a maior quantidade de exemplos de teste associadas a ele.

Para incrementar o processo de classificação, o *Adaptive*-IDT utiliza a tecnologia adaptativa. Como explicado anteriormente, existem alguns nós internos da árvore denotados por um símbolo ?, que representa um valor de variável *default* e a existência de uma função adaptativa. O objetivo dessa função é unicamente modificar

o valor do atributo ao qual ela está acoplada, afim de que o mesmo possa coincidir com o exemplo lido. Dessa maneira, o nó torna-se “adaptável” ao exemplo, permitindo que o percorrimento do caminho possa continuar. Esse mecanismo que trata uma condição de não determinismo, permite uma maior flexibilidade na árvore, podendo ajustar-se a uma instância durante a sua classificação.

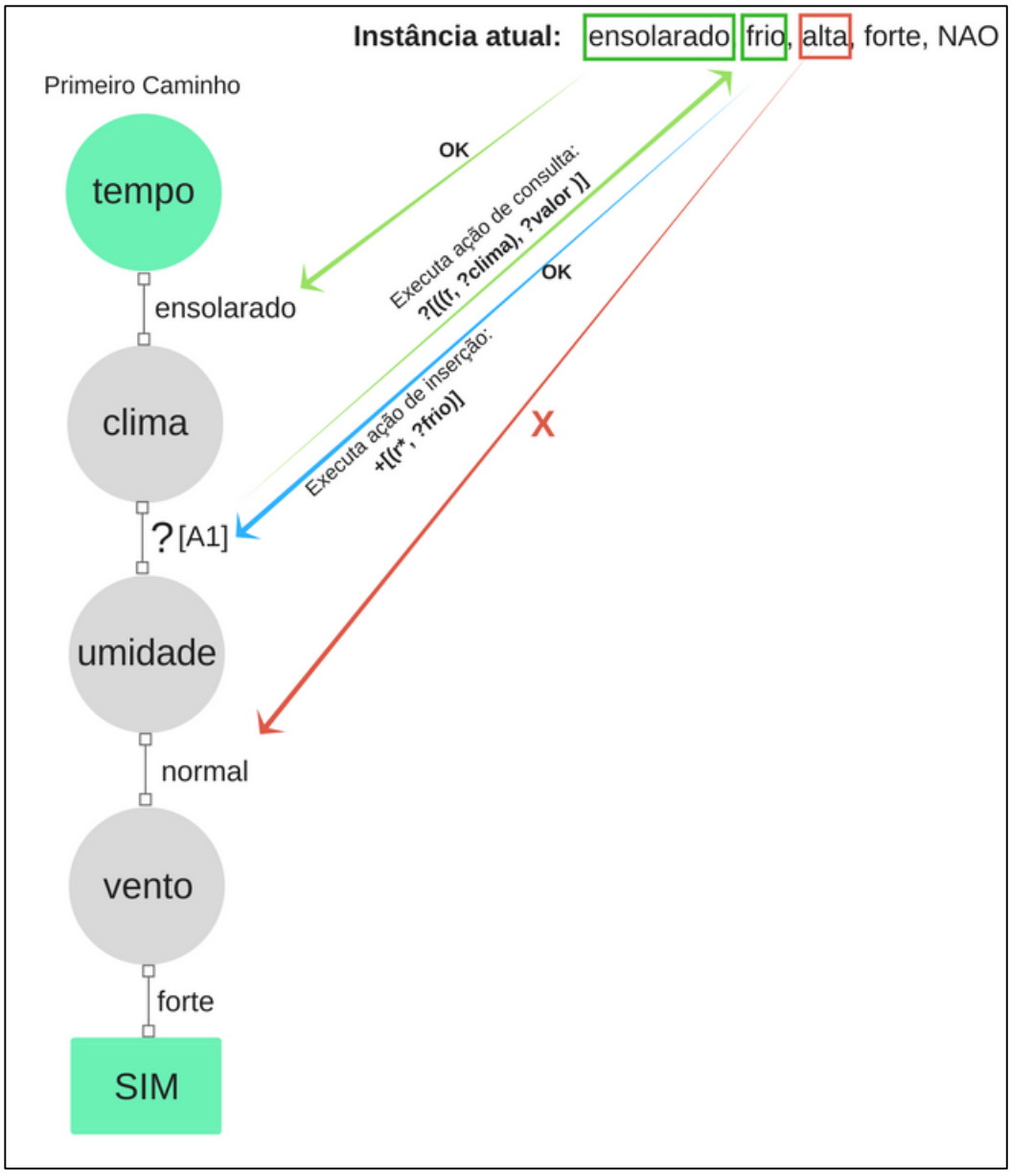


Figura 15: Processo de classificação (1).

Fonte: Autoria própria.

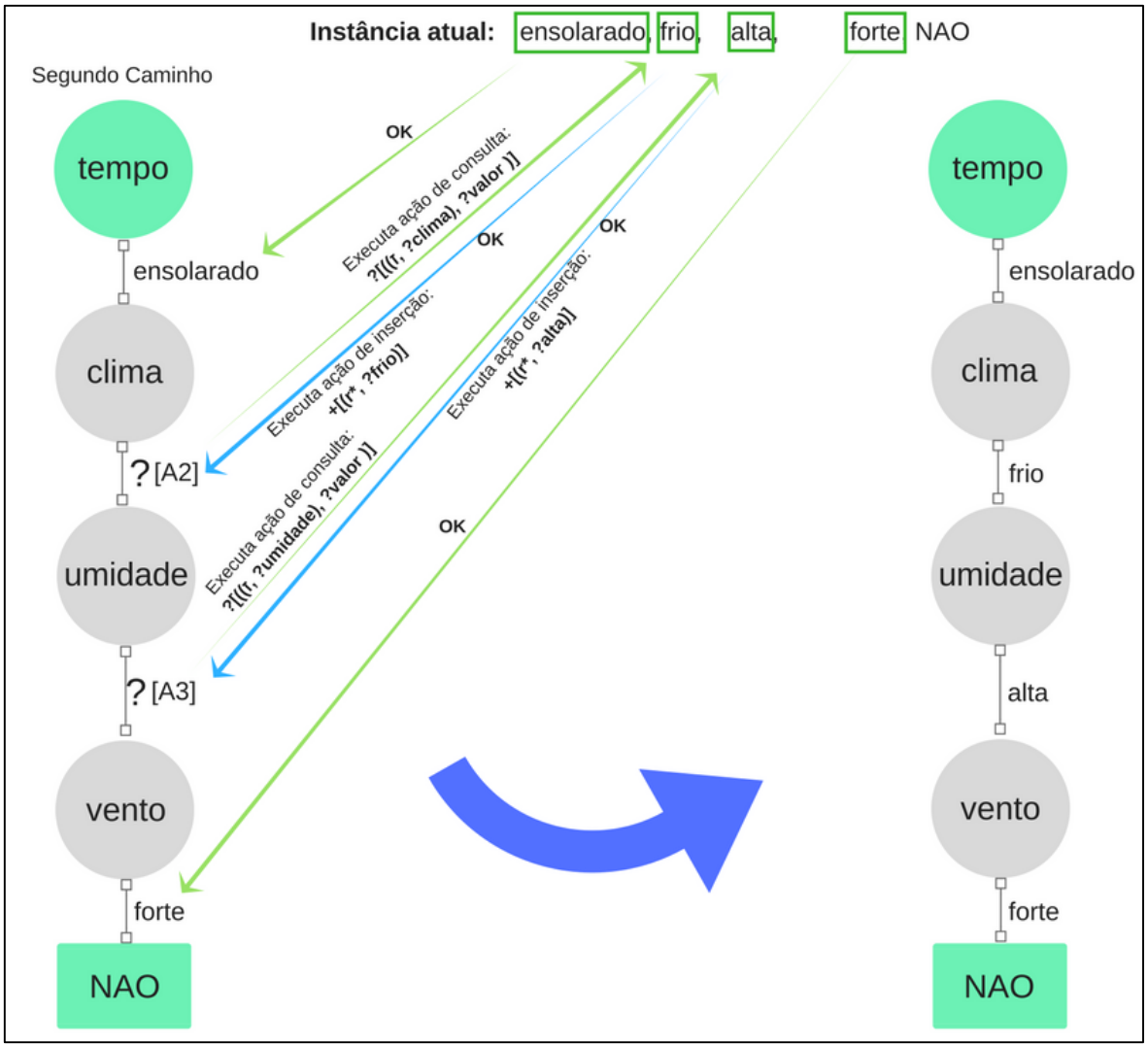


Figura 16: Processo de classificação (2).

Fonte: Autoria Própria.

As Figuras 15 e 16 ilustram uma situação em que o algoritmo deseja classificar uma instância de seguinte configuração:

- tempo = ensolarado,
- clima = frio,
- umidade = alta,
- vento = forte,
- jogar = NAO.

Para isso, a árvore disponibiliza dois caminhos. O primeiro deles, mostrado na Figura 15, possui o valor de raiz **tempo = ensolarado**, o primeiro nó interno de sua

estrutura denotado pelo símbolo ?, e os demais pelos valores *normal* e *forte*. Dessa maneira, a função adaptativa [A1] e suas ações elementares foi acionada. A ação de consulta realizou um busca no exemplo atual e verificou qual o valor do atributo *clima*. Após obtido o valor “frio”, a ação de inserção define tal valor para o nó *clima* do caminho (clima = frio), substituindo o valor ?. Devido a isso, a comparação entre os valores do exemplo de teste e do nó será verdadeira, possibilitando a verificação dos nós subsequentes. No entanto, como próximo nó do caminho possui valor normal (umidade = normal), e o valor da instância possui valor alta (umidade = alta), a operação para este caminho é finalizada e o próximo caminho será verificado.

A Figura 16, por sua vez, ilustra o segundo caminho da árvore. Tal caminho possui o valor de raiz **tempo = ensolarado**, e todos os seus nós internos compostos de funções adaptativas. Dessa maneira, todos os nós internos foram adaptados ao exemplo, e tiveram seus valores ?, substituídos pelos seus correspondentes na instância de teste, sendo possível assim, percorrer todo o caminho e retornar, corretamente, um valor classe (**jogar = NAO**).

Uma observação importante a ser feita é que, durante a etapa de classificação do *Adaptive*-IDT, para cada uma das instâncias de teste, no momento em que os caminhos e nós forem percorridos, o algoritmo priorizará os ramos que possuírem, em algum de seus nós internos, valores definidos. Caso o algoritmo percorresse primeiro um caminho composto apenas de nós com funções adaptativas, o resultado já seria obtido, sem que nenhum outro critério fosse avaliado. Percorrendo-se primeiro caminhos que contenham valores, a classificação de um determinado exemplo pode ser ainda mais específica, levando em consideração que o nó que possui o valor, foi determinante para a continuação ou interrupção das demais verificações.

A Figura 17, mostrada a seguir, apresenta de forma geral como ocorre a operação da camada de processamento.

1: **Início**
2: **para** Cada um dos exemplos de teste **faça**
3: Percorra cada um dos caminhos da árvore e, para cada caminho, percorra todos os nós, desde a raiz até a folha.
4: **para** Cada um dos nós **faça**
5: **se** Nó atual da árvore é folha **então**
6: Terminar a operação e retornar o valor do nó corrente.
7: **senão**
8: Continue a operação.
9: **fim se**
10: **se** Nó atual possui símbolo *?*, ou seja, possui função adaptativa **então**
11: Executar ação adaptativa de consulta: ?[((ɪ, ?atributo corrente), ?valor)] que obtém o valor do atributo atual do exemplo corrente.
12: Executar *ação* adaptativa de inserção: +[(r*, ?valor)] que substitui o símbolo ? do nó atual do caminho pelo valor obtido no passo 11.
13: **fim se**
14: **ler** Valor do atributo atual do exemplo corrente.
15: **se** Valor do atributo atual = Valor do nó corrente **então**
16: Retornar ao passo 4 e processar o próximo nó do caminho.
17: **senão**
18: Retornar ao passo 3 e processar o próximo caminho da árvore.
19: **fim se**
20: **fim para**

Figura 17: Operação da camada de classificação.

Fonte: Autoria própria.

4 EXPERIMENTOS

Este Capítulo têm como objetivo apresentar os primeiros experimentos realizados com o algoritmo *Adaptive*-IDT, bem como os resultados obtidos e discussões.

4.1 FERRAMENTA DE TESTES

O *Adaptive*-IDT foi implementado utilizando a linguagem de programação Java e reutiliza algumas classes do pacote WEKA (Waikato Environment for Knowledge Analysis) (WITTEN; FRANK, 2000). O WEKA é uma ferramenta open source software, que contempla uma série de algoritmos utilizados para realizar tarefas de aprendizagem de máquina e mineração de dados. Dentre suas funcionalidades, permite o acesso a relatórios com dados analíticos e estatísticos, tornando-se um grande atrativo para diversos tipos de aplicações.

4.2 CONJUNTO DE TREINAMENTO

Em relação aos conjuntos de dados utilizados, todos estão publicamente disponíveis no repositório do UCI[2]. Excepcionalmente, para fins didáticos, foi utilizado um conjunto de dados presente na ferramenta do WEKA. O primeiro conjunto utilizado foi o *play tennis*, bastante conhecido na área de AM (WANG; ZANIOLO, 2003; MITCHELL, 1997). Introduzido inicialmente por (QUINLAN, 1986) e retirado da biblioteca do WEKA para os testes, o mesmo conta com 14 instâncias de treinamento e 4 atributos nominais, indicando quais dias são mais apropriados para se jogar tênis. O segundo conjunto utilizado, *lenses*, possuindo 4 atributos e 24 instâncias, determina quais tipos de lentes devem ser utilizadas por pessoas com base nas características e sintomas apresentados por elas. Por último, foi utilizado o conjunto de dados

[2] Disponível em http://archive.ics.uci.edu/ml/

balloons, o qual possui 20 instâncias de treinamento e 4 atributos, já sendo utilizado anteriormente em (PAZZANI; 1991).

4.3 ALGORITMOS UTILIZADOS

A fim de mensurar a relevância do trabalho, foram realizadas comparações com alguns disponíveis na ferramenta WEKA: J48 e *Hoeffding Tree.*

Algoritmo de indução de árvore incremental: foi utilizado o *Hoeffding Tree* que é um algoritmo incremental de indução de árvores de decisão que, por meio de um conceito denominado Limite de *Hoeffding*, disponibiliza garantias de desempenho (HULTEN; SPENCER; DOMINGOS, 2001).

Algoritmo de indução de árvore não incremental: O J48 é a implementação do algoritmo C4.5 apresentado por (QUINLAN, 1993), sendo uma extensão do algoritmo ID3, também desenvolvido pelo mesmo autor. Tal algoritmo possui a capacidade de lidar com valores categóricos e numéricos, além de poder lidar com valores ausentes.

4.4 MEDIDA DE AVALIAÇÃO E MÉTODO DE VALIDAÇÃO

Para a realização dos primeiros experimentos do *Adaptative-IDT*, foram utilizadas medidas de desempenho baseadas na quantidade de exemplos classificados corretamente, e na porcentagem de acertos obtidos pelo algoritmo.

Quanto ao método de validação, foi utilizada a validação cruzada. Este método, também denominado *n-fold cross validation*, divide o conjunto de dados de tamanho m em n conjuntos disjuntos. O algoritmo é treinado n vezes, de modo que a cada treinamento, um dos conjuntos é utilizado para teste, e os demais para treino. Dessa forma, a medida de desempenho para este método é estimada através de acertos ou erros médios sobre estes n conjuntos de dados.

Portanto, para o trabalho, foram realizadas 5 execuções de validação cruzada para cada conjunto de dados, e as estatísticas foram obtidas a partir da média

aritmética dos resultados dessas execuções. A seguir, estão especificadas as configurações utilizadas para o método.

Conjuntos de dados	Tamanho do conjunto	Número de folds (n)	Tamanho dos folds	Número de execuções por algoritmo
play tennis	14	2	7	10
lenses	24	3	8	15
balloons	20	4	5	20

Tabela 5: Configurações da validação cruzada.

Fonte: Autoria própria.

4.5 RESULTADOS E DISCUSSÃO

No WEKA, foi utilizada a configuração padrão para ambos algoritmos, ou seja, sem nenhum parâmetro adicional. Para obter as medidas de desempenho à cerca dos mesmo, foi utilizado um recurso disponibilizado pela própria ferramenta, que permite a obtenção e avaliação de diversos dados analíticos e estatísticos.

A Tabela 6 apresenta as taxas de acerto médias para cada algoritmo, executados sobre cada um dos 3 conjuntos utilizados. Os dados indicam que para os conjuntos de dados *lenses* e *balloons*, o desempenho obtido pelo *Adaptive*-IDT foi relativamente inferior, apresentando diferenças variando, aproximadamente, entre 18% e 40%. Em contrapartida, para o conjunto de dados *play tennis*, o algoritmo apresentou um melhor desempenho. De modo geral, é possível observar a superioridade do algoritmo C4.5 na grande maioria dos casos. No entanto, por se tratar de um algoritmo não-incremental, a dinamicidade dos conjuntos de dados pode acarretar um potencial aumento no tempo de treinamento.

Conjuntos de dados	Algoritmos		
	***Adaptive*-IDT**	***J48 (C4.5)***	***Hoeffding-Tree***
play tennis	61,24 %	54,28 %	50,5 %
lenses	45,03 %	79,13 %	63,33 %
balloons	60 %	100.0%	100.0 %

Tabela 6: Comparação das taxas de acerto.

Fonte: Autoria própria.

Os dados apresentados estão sintetizados no gráfico a seguir.

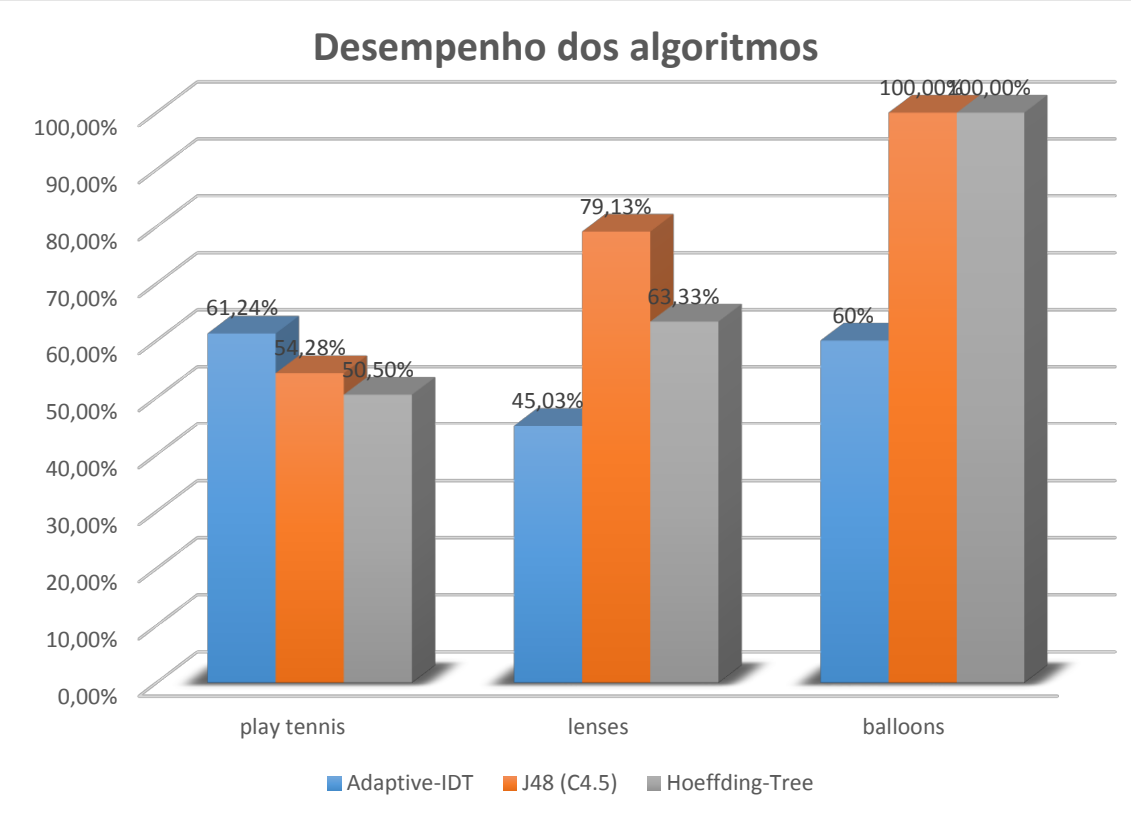


Figura 18: Gráfico comparativo do desempenho dos algoritmos.

Fonte: Autoria própria.

Vale ressaltar que, por conta do método de generalização utilizado, para determinados conjuntos de dados, o *Adaptive*-IDT pode apresentar resultados melhores ou piores. Para os conjuntos de dados que possuam, para cada uma de suas instâncias, uma grande variância nos valores de seus atributos, o algoritmo pode

apresentar um desempenho inferior aos demais. Portanto, para conjuntos de dados que possuem uma incidência padrão de determinados valores implicando em uma classificação em comum, a frequência relativa torna-se um ponto determinante, trazendo melhores resultados para o *Adaptive*-IDT.

Também é importante ressaltar que a cada nova instância lida, o *Adaptive-IDT* não precisa, necessariamente, construir um novo caminho. Também pode-se atualizar, convenientemente, os ramos já existentes, conferindo ao modelo, a característica de aprendizagem incremental sem a adição desnecessária de novas regras. Embora a minimalização da árvore não seja um dos grandes focos do algoritmo, tais atualizações permitem a criação de uma estrutura genérica significantemente simples, sem a necessidade de se possuir o conjunto de treinamento completo. No entanto, quanto maior a diversidade dos dados, melhores serão as estimativas.

5 CONSIDERAÇÕES FINAIS

A indução de árvores de decisão é hoje um dos métodos mais utilizados para a resolução de problemas de tomada de decisão, classificação e aprendizagem de máquina. Utilizando o conceito da adaptatividade, as árvores de decisão podem se tornar ferramentas eficazes para a inferência indutiva, podendo também serem utilizadas para a solução de problemas de aprendizagem incremental.

Foi apresentado neste trabalho o *Adaptive-IDT*, um novo algoritmo para indução de árvores de decisão adaptativas, visando à utilização da tecnologia adaptativa como uma possível solução para a aprendizagem incremental. O algoritmo, baseado no funcionamento do primeiro algoritmo de indução de árvores de decisão adaptativas de nome *Adaptree*, foi capaz de gerar, a partir de um conjunto de dados de treinamento, uma árvore de decisão não determinística como saída. Tal árvore possui características adaptativas que permitem que sua estrutura interna seja alterada durante o processo de classificação.

O trabalho também descreveu uma série de conceitos a respeito da aprendizagem de máquina, tecnologia adaptativa e indução de árvores de decisão, sendo estes fundamentais para a compreensão do funcionamento, estrutura e formulação do algoritmo proposto. Além da utilização da tecnologia adaptativa, o algoritmo também faz uso de conceitos de não-determinismo e técnicas estatísticas para o aprendizado e classificação de conjuntos de dados.

5.1 TRABALHOS FUTUROS

Os resultados do algoritmo foram comparados com outros algoritmos de árvores de decisão, sendo um incremental e outro não incremental. O algoritmo não foi comparado com nenhum algoritmo adaptativo, pois não foi encontrado nenhum disponível para *download*. Embora o desempenho do *Adaptive*-IDT tenha sido relativamente inferior aos algoritmos com os quais foi comparado, um estudo mais aprofundado acerca do seu método de generalização pode melhorar o seu desempenho. Isso porque existem métodos mais sofisticados que podem ser

utilizados, por exemplo o uso de ganho de informação ao invés da frequência relativa, sendo assim, um dos trabalhos futuros é a realização desta modificação no algoritmo.

Uma das limitações do algoritmo é a utilização de atributos categóricos e não lidar com atributos ruidosos. Isso gerou uma dificuldade em encontrar conjuntos de treinamento com essas características. Desta forma pretende-se buscar técnicas para tratamento de atributos numéricos, valores ausentes ou inconsistentes e incorporar essas funções no algoritmo a fim de obter maiores possibilidades de uso. Além disso, para o *Adaptive*-IDT, a utilização da adaptatividade limitou-se somente ao processo de classificação. Por conta disso, pretende-se também buscar outras meios de empregá-la, para que a mesma possa ter um papel fundamental tanto no treinamento quanto na classificação do algoritmo.

O uso da tecnologia adaptativa em conjunto com técnicas de aprendizagem de máquina, pode ser uma opção para problemas de aprendizagem incremental. A fim de dispor de um recurso prático para a comunidade de pesquisa em adaptatividade, pretende-se distribuir o *Adaptive*-IDT de maneira *open source*, uma vez que os algoritmos existentes para essa finalidade não o fazem. Espera-se também que o algoritmo possa ser uma alternativa aos demais algoritmos existentes, sendo utilizado em tarefas de aprendizado incremental, classificação e tomada de decisão.

REFERÊNCIAS

ALPAYDIN, Ethem. **Introduction to Machine Learning**. 2ª Edição. MIT Press, 2010. ISBN 978-0-262-01243-0.

ANGLUIN, D., SMITH, C. H. (1983) **Inductive Inference: Theory and Methods**. Comput. Surveys, v. 15, no. 3, pp. 237–269

ALFENAS, D. A.; BARRETTO, M.R.P.. Adaptatividade em robôs sociáveis, uma proposta de um gerenciador de diálogos. In: Sexto Workshop de Tecnologia Adaptativa - WTA 2012. EPUSP, 2012.

ALFENAS, Daniel A. **Aplicações da tecnologia adaptativa no gerenciamento de diálogo falado em sistemas computacionais**. Dissertação de Mestrado, EPUSP, São Paulo, 2014.

BALDI, Junior C., RODRIGUES, RODRIGUES. **Modelo adaptativo para um framework educacional**. Em: Memórias do X Workshop de Tecnologia Adaptativa - WTA 2016. EPUSP, São Paulo. ISBN: 978-85-86686-86-3, pp. 70-79. 28 e 29 de Janeiro, 2016.

BASSETO, Bruno A. Um sistema de composição musical automatizada, baseado em gramáticas sensíveis ao contexto, implementado com formalismos adaptativos. Dissertação de Mestrado, Escola Politécnica da USP, São Paulo, 2000.

BISHOP, C. M. **Neural Networks for Pattern Recognition**. Oxford University Press, 1995. ISBN 0-19-853864-2 (Paperback).

CALINON, S., BILLARD, A. (2007). Incremental Learning of Gestures by Imitation in a Humanoid Robot.

CATAE, Fabrício S.; ROCHA, Ricardo L. A. Introdução a Árvores de Decisão Adaptativas. In: WORKSHOP DE TECNOLOGIA ADAPTATIVA, 5. São Paulo. **Anais eletrônicos…** São Paulo: 2011. Disponível em: <http://lta.poli.usp.br/lta/publicacoes/artigos/2011/catae-e-rocha-2011-introducao-a-arvores-de-decisao-adaptativas/view>. Acesso em: 01 abr. 2011.

CEREDA, Paulo R. M.; NETO, João J. Utilizando linguagens de programação orientadas a objetos para codificar programas adaptativos. Memórias do IX Workshop de Tecnologia Adaptativa - WTA 2015. EPUSP, São Paulo, ISBN: 978-85-86686-82-5, pp. 2-9. Janeiro, 2015.

CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. **Introduction to Algorithms.** 3ª Edição. MIT Press, 2009. ISBN 978-0-262-03384-8.

DUDA, R. O.; HART, P. E.; STORK, D. G.: **Pattern Classification**. 2a Edição. Wiley, 2001. ISBN: 0471056693.

ElWELL, Ryan, POLIKAR, Roby**. Incremental learning of concept drift in nonstationary environments**. IEEE Transactions on Neural Networks 22.10 (2011): 1517-1531

FARIA, Francisco, PEREIRA, João. **Uma proposta de método adaptativo para seleção automática de preenchimento de texto (comunicação técnica)**. Em: Memórias do XI Workshop de Tecnologia Adaptativa - WTA 2017. EPUSP, São Paulo. ISBN: 978-85-86686-90-0, pp. 28-30. 26 e 27 de Janeiro, 2017.

GAMA, João; CARVALHO, André P. L.; FACELI, Katti; LORENA, Ana C.; OLIVEIRA, Marcia. **Extração de conhecimento de dados: data mining**. 2. ed. Lisboa: Edições Sílabo, 2015. 428 p. Disponível em: <http://www.producao.usp.br/handle/BDPI/51012>

GANZELI; BOTTESINI; PAZ; RIBEIRO. **SKAN, Skin Scanner, software para o reconhecimento de câncer de pele utilizando técnicas adaptativas**. In: Quarto Workshop de Tecnologia Adaptativa - WTA 2010. EPUSP, 2010.

GOMEZ, CAMOLESI. **Visualizador gráfico para redes de Petri adaptativa**. In: Segundo Workshop de Tecnologia Adaptativa - WTA 2008. EPUSP, 2008.

HULTEN, Geoff; Spencer, DOMINGOS, Laurie P. **Mining time-changing data streams**. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001.

IWAI, M. K. **Um formalismo gramatical adaptativo para linguagens dependentes de contexto**. Tese de Doutorado, Escola Politécnica da USP, São Paulo, 2000.

LEME, Flávio, MARTINS, Mozart F., VICTOR, Robério. **Uso da tecnologia adaptativa para reconhecimento do condutor de um veículo (comunicação técnica)**. Em: Memórias do XI Workshop de Tecnologia Adaptativa - WTA 2017. EPUSP, São Paulo. ISBN: 978-85-86686-90-0, pp. 10-14. 26 e 27 de Janeiro, 2017.

BARROS; MARE; DOTA; CUGNASCA, LEITE, Brenda C. C. **Operação Sustentável de Sistemas de Ar Condicionado Usando Tecnologia Adaptativa.** Em: Memórias

do IX Workshop de Tecnologia Adaptativa - WTA 2015. EPUSP, São Paulo, ISBN: 978-85-86686-82-5, pp. 106-112. 29 e 30 de Janeiro, 2015.

MITCHELL, Tom M. **Machine Learning.** 1ª Edição. McGraw-Hill, 1997. ISBN: 0070428077.

NETO, J. J.: **Solving complex problems with Adaptive Automata**. Lecture Notes in Computer Science. S. Yu, A. Paun (Eds.): Implementation and Application of Automata 5th International Conference, CIAA 2000, Vol.2088, London, Canada, Springer-Verlag, pp.340, 2000.

NETO, João J. Um levantamento da pesquisa em técnicas adaptativas na epusp. **Sistemas e Computação,** Salvador, v.1, n.1, p. 23-47, jan./jun. 2011.

NETO, João. J. Adaptive Automata for Context-Sensitive Languages. SIGPLAN NOTICES, Vol. 29, n. 9, pp. 115-124, September, 1994.

PISTORI, Hemerson; NETO, João J. **Adaptree – Proposta de um Algoritmo para Indução de Árvores de Decisão Baseado em Técnicas Adaptativas**. 2002. Anais Conferência Latino Americana de Informática - CLEI. Montevideo, Uruguai, 2002.

PISTORI, Hemerson. **Tecnologia Adaptativa Em Engenharia De Computação: Estado Da Arte e Aplicações.** 2003. 174p. Tese - Escola Politécnica da Universidade de São Paulo, São Paulo, 2003.

PISTORI, H., SOUZA, K. P. **Tecnologia adaptativa aplicadaa biotecnologia: Estudos de caso e oportunidades**. *WORKSHOP DE TECNOLOGIA ADAPTATIVA. Escola Politécnica da USP*. Vol. 4. 2010.

PICAGLI, Danilo S. ***Tradução grafema-fonema para a língua portuguesa baseada em autômatos adaptativos***. Diss. Universidade de São Paulo, 2008.

QUINLAN, J. R. **C4.5: Programs for Machine Learning**. Morgan Kaufmann Publishers, 1993.

QUINLAN, J. R. **Induction of Decision Trees**. *Machine Learning* 1 (1986): 81-106.

RESENDE, Solange O. **Sistemas Inteligentes: fundamentos e aplicações.** Barueri, São Paulo: Manole, 2005. ISBN: 85-204-1683-7.

ROCHA, Ricardo. L. A.; NETO, João. J. Autômato adaptativo, limites e complexidade em comparação com máquina de Turing. In: Proceedings of the second Congress of Logic Applied to Technology - LAPTEC 2000. São Paulo: Faculdade SENAC de Ciências Exatas e Tecnologia, p. 33-48, 2001.

ROSS, DAVID A., JONGWOO Lim, RUEI S. L., and Ming-Hsuan Yang. **Incremental learning for robust visual tracking**. International journal of computer vision 77, no. 1 (2008): 125-141.

RUSSELL, P.; NORVING, P. **Artificial Intelligence: A Modern Approach**. Prentice Hall, Englewood Cliffs, NJ (1995).

SILVA, Marcelino P. S. Mineração de Dados - Conceitos, Aplicações e Experimentos com Weka. 2004. Belo Horizonte, UFMG. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/erirjes/2004/004.pdf>

SILVA, A. M. ;ROCHA, R. L. A.; NETO, J.J.. Análise semântica de sentimentos utilizando árvores de decisão adaptativas. Em: Memórias do X Workshop de Tecnologia Adaptativa - WTA 2016. EPUSP, São Paulo. ISBN: 978-85-86686-86-3, pp. 37-45. 28 e 29 de Janeiro, 2016

STANGE, Renata L. **Adaptatividade em aprendizagem de máquina: conceitos e estudo de caso**. 2011. 98p. Dissertação - Escola Politécnica da Universidade de São Paulo, São Paulo, 2011.

STANGE, Renata L, CEREDA, Paulo R. M., NETO João J. **Agentes adaptativos reativos: formalização e estudo de caso**. Em: Memórias do XI Workshop de Tecnologia Adaptativa - WTA 2017. EPUSP, São Paulo. ISBN: 978-85-86686-90-0, pp. 63-71. 26 e 27 de Janeiro, 2017.

TCHEMRA, Angela H. Tabela de Decisão Adaptativa na Tomada de Decisão Multicritério. Tese de Doutorado, EPUSP, São Paulo, 2009.

UTGOFF, P. E.: *Incremental Induction of Decision Trees*. Machine Learning, Machine Learning, 4, 161-186, 1989.

WANG, Haixun; ZANIOLO, Carlox. "Atlas: A native extension of sql for data mining." *Proceedings of the 2003 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 2003.

WITTEN, I. H., EIBE, F. **Data Mining: Practical Machine Learning Tools and Techniques**. 2a Edição. Morgan Kaufmann, 2005. ISBN 0-12-088407-0.

YOSHIDA, Murilo L. **Aprendizado supervisionado incremental de Redes Bayesianas para mineração de dados.** 2007. 132p. Dissertação - São Carlos : UFSCar, 2007.

APÊNDICE A – DIAGRAMA DE CLASSES DO ALGORITMO