UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE UM SISTEMA DE EXTRAÇÃO DE REGRAS SIMBÓLICAS DE REDES NEURAIS ARTIFICIAIS UTILIZADAS NA TAREFA DE CLASSIFICAÇÃO EM DATA MINING

ALEXANDRE DANIEL DALABRIDA

BLUMENAU
2007

2007/2-02

ALEXANDRE DANIEL DALABRIDA

PROTÓTIPO DE UM SISTEMA DE EXTRAÇÃO DE REGRAS SIMBÓLICAS DE REDES NEURAIS ARTIFICIAIS UTILIZADAS NA TAREFA DE CLASSIFICAÇÃO EM DATA MINING

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Mauro Marcelo Mattos, Doutor - Orientador

BLUMENAU
2007

2007/2-02

PROTÓTIPO DE UM SISTEMA DE EXTRAÇÃO DE REGRAS SIMBÓLICAS DE REDES NEURAIS ARTIFICIAIS UTILIZADAS NA TAREFA DE CLASSIFICAÇÃO EM DATA MINING

Por

ALEXANDRE DANIEL DALABRIDA

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

	_____
Presidente:	Prof. Mauro Marcelo Mattos, Doutor – Orientador, FURB
	_____
Membro:	Prof. Marcel Hugo, Mestre – FURB
	_____
Membro:	Prof. Claudio Loesch, Doutor – FURB

Blumenau, 14 de dezembro de 2007

Dedico este trabalho a todos que de alguma forma me apoiaram durante a realização do mesmo, aos amigos, à minha mãe e à minha avó materna, e ao meu orientador.

AGRADECIMENTOS

Aos meus amigos, também pelo apoio e incentivo para conclusão do trabalho.

Ao meu orientador, Mauro Marcelo Mattos, por ter acreditado na conclusão deste trabalho, e por ter se dedicado para possibilitar isso.

RESUMO

Este trabalho tem como objetivo principal avaliar a aplicabilidade das técnicas de extração de regras de redes neurais artificiais, mais especificamente na tarefa de classificação de dados em *data mining*. Para avaliar esta aplicabilidade foi implementado um protótipo de sistema de *data mining* específico para a tarefa de classificação de dados, baseado em rede neural artificial, treinando a mesma para classificar dados oriundos de uma fonte de dados. Através de algoritmos de extração de regras específicos aplicados sobre a rede treinada, foram extraídas e articuladas regras de classificação mais apropriadas ao entendimento humano. Na construção do protótipo, foram analisadas técnicas e tarefas de *data mining,* redes neurais artificiais e sobre extração de regras de redes neurais. Como conseqüência do desenvolvimento deste trabalho, demonstrou-se que a extração das regras implícitas na rede neural auxilia claramente no entendimento do modelo de classificação de dados, evidenciando a hipótese inferida pela rede de uma forma mais clara ao entendimento.

Palavras-chave: D*ata mining.* Redes neurais artificiais. Extração de regras.

ABSTRACT

This work has a main goal to evaluate the applicability of rule extraction techniques over the artificial neural nets, more specifically in the data mining task of data classification. To evaluate this applicability was implemented an archetype of data mining system specifically for the classification task based on artificial neural net, training the same to classify data for external data sources. Through applied specific rule extraction algorithms over the trained net, had been extracted and articulated more appropriate rules of classification to the human agreement. In the construction of the archetype, had been analyzed techniques and tasks of data mining, artificial neural nets and on extraction of rules of neural nets. As consequence of the development of this work, was demonstrated that the extraction of the implicit rules in the neural net assists clearly in the agreement of the model of classification of data, evidencing the hypothesis inferred for the artificial neural network of a clearer form to the agreement.

Key-words: Data mining. Artificial neural networks. Rule extraction.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

IA – Inteligência Artificial

DLL – *Dynamic Link Library*

KDD – *Knowledge Discovery in Databases*

LMS – *Least Mean-Square*

MLP – *Multilayer Perceptron*

OLAP – *On Line Analysis Processing*

RNA – Rede Neural Artificial

RTTI – *Run Time Type Information*

SQL – *Structured Query Language*

UML – *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

No decorrer dos últimos anos, a capacidade de gerar e coletar dados tem crescido rapidamente. A informatização cada vez mais maciça e os avanços no armazenamento e coleta de dados têm provido uma imensa quantidade de dados e tal volume tem claramente ultrapassado a habilidade humana de interpretar os mesmos. Tornou-se óbvia a necessidade urgente de novas técnicas ou ferramentas que possam, de forma inteligente e automática, transformar os dados processados em informação útil e conhecimento. Essas técnicas e ferramentas são assunto do campo da descoberta de conhecimento em bancos de dados, ou *Knowledge Discovery in Databases* (KDD) (FAYYAD et al., 1996, p. 2).

Por descoberta de conhecimento em bases de dados entende-se encontrar regularidades e informações interessantes de alto nível, que são extraídas de um conjunto relevante de dados e investigadas de diferentes ângulos. O conhecimento descoberto pode ser aplicado a sistemas de informação, processamento de consultas, como suporte à tomada de decisões, controle de processos e muitas outras aplicações (CHEN, HAN e YOU, 1996, p. 866).

Nesse contexto, a *data mining* apresenta-se como uma classe de métodos e técnicas que são usadas em algumas das etapas do processo de KDD (FAYYAD et al., 1996, p. 3). Conforme Chen, Han e You (1996, p. 875), pesquisas são feitas em muitos campos da área da computação, como sistemas gerenciadores de bancos de dados, aprendizado de máquina, estatística, aquisição de conhecimento, e fornecem para a *data mining* as mais variadas ferramentas e técnicas para a execução das suas tarefas.

Uma das tarefas de *data mining* é a tarefa de classificação de dados em um grande conjunto de dados. Segundo Chen, Han e You (1996, p. 875), a classificação de dados é o processo ou a tarefa que procura propriedades comuns entre um conjunto de objetos em uma base de dados e então classifica estes objetos em diferentes classes pré-definidas, de acordo com um modelo ou regras de classificação. Aplicações da classificação incluem, por exemplo, diagnósticos médicos, predição de desempenho e seleção de mercado.

A classificação de dados utiliza como ferramentas artefatos das áreas de estatística, aprendizado de máquina, redes neurais e sistemas especialistas (KULIKOWSKI e WEISS, 1991, p. 271). Vários algoritmos de classificação resultantes desses estudos foram então desenvolvidos. Segundo Kecman (2001, p. 35), as redes neurais artificiais representam uma família de modelos que tem sido utilizada em uma ampla variedade de problemas de classificação em *data mining*, onde as seguintes vantagens estão presentes:

a) alto rendimento da generalização perante os exemplos não vistos durante a fase de aprendizado;
b) robustez na presença de imprecisão ou ruído nos dados de entrada;
c) habilidade para aprender relações não lineares, a qual é uma característica importante para manipular bases de dados reais.

Uma das limitações, porém, nos modelos neurais artificiais, é que o conhecimento derivado do processo de aprendizagem encontra-se sintetizado em uma forma não simbólica (matrizes de pesos), o que torna difícil sua interpretação.

Para superar essa limitação das redes neurais em relação a sua falta de transparência, a hipótese gerada por estes modelos pode traduzir-se a uma forma interpretável. Os métodos que realizam essa transformação são conhecidos como algoritmos de extração de regras (SETIONO; LEOW; ZURADA, 2002, p. 564). Conforme Berthold e Hand (1999, p. 53), ainda que o requisito de uma solução compreensível dependa das necessidades da aplicação e dos usuários finais do sistema, as vantagens que se obtêm com estas transformações são:

a) provisão da capacidade de explicação, a qual resultará em um melhor entendimento do problema e da sua solução;
b) exploração de dados, ao revelar os relacionamentos entre as variáveis de sistema que foram descobertas durante a aprendizagem e que se encontram codificadas em um formato incompreensível.

Segundo Andrews, Diederich e Tickle (1995, p. 373), o conhecimento capturado por uma rede neural durante sua aprendizagem está representado em termos da topologia da rede, das funções de ativação utilizadas pelos neurônios e pelos parâmetros associados com as conexões (pesos). As técnicas de extração devem então interpretar estes elementos e a partir dos mesmos gerar uma descrição mais compreensível que seja funcionalmente equivalente à rede neural.

Conforme Andrews, Diederich e Tickle (1995, p. 380), os tipos de técnicas de extração de regras podem ser subdividas em: decomposicional, pedagógica e eclética. A técnica decomposicional procura extrair as regras no nível das unidades individuais da rede neural (neurônios intermediários e de saída), tendo uma visão interna e transparente da rede. Já a abordagem pedagógica não explora o interior da rede neural, considerando então a extração de regras das redes neurais como uma tarefa de aprendizado, onde o conceito alvo é a função computada pela rede e as características das entradas são simplesmente as entradas da rede neural. Assim, as técnicas pedagógicas auxiliam a extrair regras que mapeiam entradas diretamente em saídas e a motivação é usar a rede neural para gerar exemplos para um

algoritmo de aprendizado simbólico. A abordagem eclética utiliza-se das características das duas abordagens, utilizando o conhecimento sobre a arquitetura interna da rede treinada (abordagem decomposicional) para complementar um algoritmo de aprendizado simbólico.

De acordo com Azevedo, Brasil e Oliveira (2000, p. 197), as técnicas de extração de regras procuram esclarecer ao usuário como a rede chegou a uma decisão, decodificando o(s) estado(s) interno(s) da rede. A maioria dos esforços tem sido direcionada no sentido de apresentar as explanações como um conjunto de regras expressas como lógica simbólica convencional, isto é, valores booleanos, na forma `se...então...senão...`, o que torna as regras bem compreensíveis do ponto de vista humano.

A aplicação de algoritmos de extração de regras de redes neurais artificiais ajuda a realizar uma aprendizagem a partir de exemplos representativos do problema, passando estas regras a formar parte da base de conhecimento. Além do mais, as regras extraídas podem ser usadas para validar a acurácia da rede ou refinar regras simbólicas já existentes para um determinado problema.

Reconhecendo então a capacidade das redes neurais na tarefa de classificação de dados em *data mining* e visto que é possível a explanação do conhecimento adquirido pelas mesmas através das técnicas de extração de regras, propõem-se especificar e desenvolver um protótipo de uma ferramenta de software que permita construir e treinar redes neurais artificiais na tarefa de classificação em *data mining* e então aplicar algoritmos de extração de regras sobre estas redes. A ferramenta deverá então exibir as regras extraídas e permitir avaliar as mesmas em relação à acurácia, ou seja, a taxa de acerto de classificação das mesmas, e também indicar a complexidade das regras extraídas. Os algoritmos de extração de regras a serem aplicados serão o RX (também conhecido como NEURORULE), de abordagem decomposicional e o TREPAN, da abordagem pedagógica.

Como estudos de caso para demonstrar a funcionalidade da ferramenta, serão utilizados conjuntos de dados de domínio público, que representam problemas específicos para a tarefa de classificação de dados em *data mining*.

O trabalho justifica-se pela necessidade de desenvolvimento e aprimoramento de tecnologias computacionais que permitam a provisão da capacidade de explanação em sistemas que utilizam técnicas baseadas em inteligência artificial, mais especificamente de redes neurais artificiais, cuja natureza conexionista torna difícil a interpretação das conclusões obtidas pela rede. A necessidade de explanação torna-se mais óbvia em sistemas de *data mining* que utilizam redes neurais como modelos de classificação, onde a eficácia do modelo proposto pode ser aferida com base nas explanações providas pela regras extraídas, além de

oferecer respaldo e segurança nas tomadas de decisões baseadas nas informações do modelo já em uso.

1.1 OBJETIVOS DO TRABALHO

O objetivo principal deste trabalho é demonstrar a aplicabilidade do emprego das técnicas e algoritmos de extração de regras de redes neurais treinadas, com ênfase em redes neurais usadas para a tarefa de classificação em *data mining*.

Os objetivos específicos do trabalho são:

a) identificar um domínio de problema de classificação de dados em data mining onde será utilizada a técnica de redes neurais para a tarefa de classificação;
b) construir e treinar uma rede neural para a resolução do problema de classificação;
c) implementar um algoritmo de extração de regras da abordagem decomposicional e um algoritmo da abordagem pedagógica sobre a rede neural treinada;
d) avaliar a acurácia e a complexidade das regras extraídas da rede neural treinada.

1.2 ESTRUTURA DO TRABALHO

O presente trabalho está estruturado em quatro capítulos, onde o segundo capítulo apresenta a fundamentação teórica sobre os temas desse trabalho, abrangendo os conceitos, técnicas, tarefas e aplicações de *data mining*. Sobre as redes neurais artificiais são apresentados os conceitos, estrutura, modelos e as técnicas de aprendizado, além de uma abordagem sobre o problema da compreensibilidade. Por fim, sobre a extração de regras de redes neurais artificiais, são abordados os conceitos, requerimentos e métodos de extração, as técnicas existentes e as técnicas analisadas para uso no trabalho.

No terceiro capítulo é apresentado o desenvolvimento do protótipo, partindo de uma visão geral do mesmo, seguida da apresentação dos requisitos do problema e de sua especificação. Este capítulo também discute a implementação do protótipo, bem como os estudos de casos desenvolvidos para demonstrar a operacionalidade da implementação e os resultados obtidos.

Finalizando, no quarto capítulo são apresentadas as conclusões e a lista de tarefas para continuidade do protótipo

2 FUNDAMENTAÇÃO TEÓRICA

Na fundamentação teórica, estão contempladas as definições e funções de *data mining,* bem como os fundamentos sobre redes neurais artificiais e, por fim, o tema da extração de regras de redes neurais artificiais é abordado.

2.1 DATA MINING

A técnica de *data mining*, que é uma das principais etapas dentro do processo de prospecção de conhecimento em bancos de dados ou *Knowledge Discovery in Databases*. Nesse contexto, os dois temas serão abordados, inicialmente sob um contexto histórico e em seguida é feita uma distinção entre os dois termos, através de suas definições. Cada tema é abordado em uma seção a parte: o processo de KDD e suas etapas e a seguir os assuntos relativos a *data mining*, suas tarefas, métodos, técnicas e algoritmos.

2.1.1 O processo de KDD

A revolução digital trouxe um problema relativo ao crescimento dos dados ao mundo da ciência, negócios e governo. Observa-se um grande crescimento na nossa capacidade de gerar e coletar dados. Avanços na tecnologia de armazenamento de dados, melhores sistemas gerenciadores de bancos de dados, a diminuição do custo dos equipamentos e a tecnologia de *data warehouse* tem permitido armazenar esta grande quantidade de dados.

Em vista disso, a capacidade para coletar e armazenar dados de todos os tipos tem ultrapassado largamente as habilidades humanas para analisar, resumir e extrair conhecimento destes dados. Métodos tradicionais de análise de dados, baseados principalmente no tratamento direto dos dados por especialistas, simplesmente não fazem escala aos volumosos conjuntos de dados. Enquanto a tecnologia de bancos de dados disponibilizou as ferramentas básicas para o eficiente armazenamento e consulta destes grandes conjuntos de dados, o fato de como facilitar o entendimento e análise humana sobre esses conjuntos de dados é um problema. Para tratar desse problema, uma geração inteira de técnicas e ferramentas para a

mineração de dados automática e de descoberta de conhecimento é necessária, originadas de pesquisas em diversas áreas, como inteligência artificial, estatística, *data warehouse*, *on-line analysis processing* (OLAP) e visualização de dados. A aplicação destas técnicas e ferramentas são assuntos do campo do KDD (FAYYAD et al., 1996, p. xiii).

O KDD foi definido por Frawley, Matheus e Piatetsky-Shapiro (1991, p. 58), como um processo não trivial para identificar válidos, potencialmente úteis e necessariamente desconhecidos padrões em dados. Já Fayyad et al. (1996, p. 9), em uma revisão desse conceito, definiu que a *data mining* é uma etapa no processo de KDD que consiste de um algoritmo particular de mineração de dados que, sob algumas limitações computacionais, gera uma lista de padrões encontrados nos dados. Já o KDD foi definido como o processo que utiliza métodos (algoritmos) de *data mining* para extrair (identificar) o que é considerado conhecimento, de acordo com medidas e limites especificados, de um conjunto de dados juntamente com algum pré-processamento, reduções e transformações desse conjunto de dados.

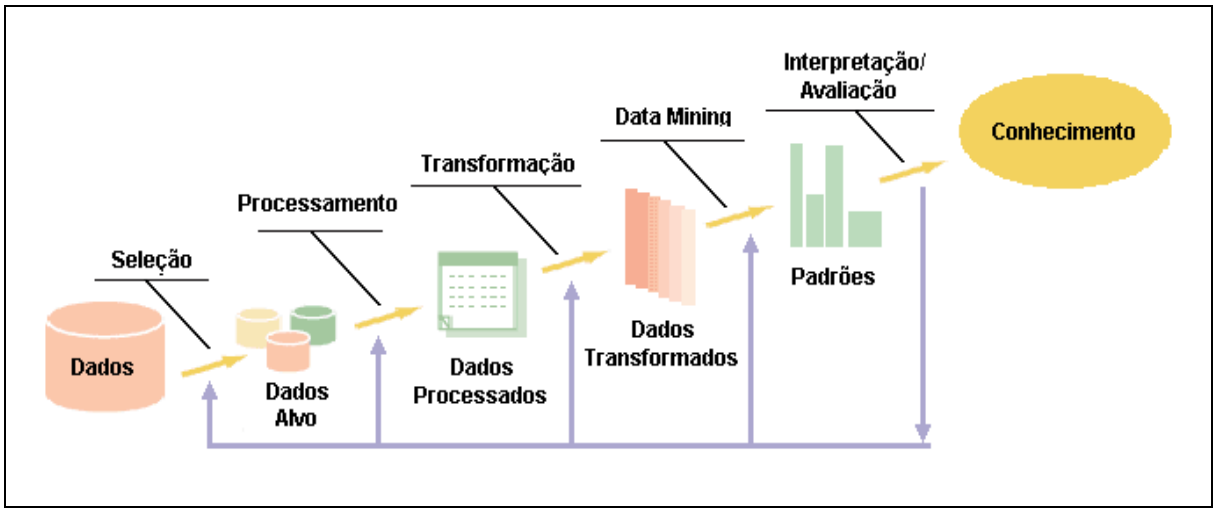
O processo total de KDD , conforme a figura 1, envolve numerosas etapas, interativas e iterativas, sendo que as principais são (MITRA; MITRA; PAL, 2002, p. 3):

a) o entendimento do domínio da aplicação, o relevante conhecimento prévio e os objetivos do usuário final;
b) a extração dos dados alvo, o que inclui a seleção de um conjunto de dados ou foco em um subconjunto de variáveis, nos quais será feita a prospecção do conhecimento;
c) a limpeza e pré-processamento dos dados, onde serão removidos dados irrelevantes, geradas estratégias para dados incompletos e coletadas informações necessárias para a escolha do algoritmo ou modelo de extração de regras;
d) a redução e projeção dos dados, que incluem a procura de métodos usuais de representação dos dados (dependendo do objetivo da tarefa) e métodos de redução ou transformação para reduzir o número efetivo de variáveis consideradas;
e) a escolha da tarefa de *data mining*, onde é decidido o objetivo do modelo derivado do algoritmo de *data mining* (sumarização, classificação, regressão, segmentação, etc.);
f) a escolha do algoritmo de *data mining*, onde é selecionado um método a ser usado para buscar padrões nos dados, bem como decidir quais os modelos e parâmetros são mais apropriados;
g) a *data mining* propriamente dito, onde é feita a procura por padrões de interesse

em uma forma particular de representação ou um conjunto de tais representações;

h) a interpretação dos padrões descobertos. Os padrões podem ser analisados de forma automática ou semi-automática para identificar os verdadeiros interesses ou padrões utilizáveis para o usuário;

i) a utilização do conhecimento descoberto, o que pode ser feito incorporando este conhecimento a algum sistema, executar ações baseadas a partir desse conhecimento ou simplesmente documentar e reportar às partes interessadas.



Fonte: adaptado de Fayyad et al. (1996, p. 10).

Figura 1 – Processo total do KDD e suas principais etapas

Todas estas etapas são interdependentes, pois os resultados de cada uma são as entradas da próxima etapa. Toda a abordagem é dirigida por resultados e cada estágio depende dos resultados do estágio anterior (HARRISON, 1998, p. 28). Porém, não existe uma ordem ou seqüência única para o andamento deste processo, e algumas etapas podem ser omitidas. Tudo isso é dependente das técnicas empregadas e dos dados sobre os quais o KDD está sendo aplicado (ÁVILA, 1998, p. 91). Em qualquer etapa pode ser necessário, por exemplo, voltar a outras etapas anteriores, desde que a técnica e os dados empregados permitam.

2.1.2 O processo de *data mining*

Conforme Mitra, Mitra e Pal (2002, p. 5), no processo de *data mining* são envolvidos modelos ajustados para dados observados ou modelos determinando padrões para estes dados. Os modelos ajustados executam a função de inferir conhecimento. Decidir se os modelos refletem ou não algum conhecimento utilizável é a parte do processo de KDD que ainda necessita do julgamento humano. Tipicamente, um algoritmo de *data mining* é constituído de

uma combinação de três componentes, os quais são:

a) um modelo, que possui uma função (como por exemplo, a classificação ou o *clustering*) e sua forma representacional (como por exemplo, a regressão linear ou redes neurais). Um modelo contém parâmetros ajustados pelos dados;
b) um critério de preferência, onde a base pela preferência de um modelo ou conjunto de parâmetros sobre outro, o que depende dos dados. O critério usualmente é uma forma de uma função de melhoramento do ajuste do modelo pelos dados, tomando o cuidado de prevenir um ajuste excessivo ou deficiente;
c) o algoritmo de busca, que é a especificação de um algoritmo para encontrar determinados padrões, quando já se possui os dados, os possíveis modelos e o critério de preferência.

Um algoritmo de *data mining* é usualmente uma instância dos três componentes, ou seja, um modelo, um critério de preferência e um algoritmo de busca. A combinação desses três elementos é utilizada para executar as tarefas de *data mining*.

Fayyad et al. (1996, p. 12) definiu o mais alto nível dos objetivos das tarefas de *data mining* em predição e descrição. A predição envolve algumas variáveis ou campos no conjunto de dados para predizer valores desconhecidos ou valores futuros para outras variáveis de interesse. Já a descrição tem como foco procurar padrões humanamente interpretáveis que descrevem os dados.

Os objetivos de predição e descrição são alcançados através das seguintes tarefas, assim enumeradas (MITRA; MITRA; PAL, 2002, p. 5):

a) a classificação de um determinado item do conjunto de dados em uma de diversas classes pré-definidas;
b) a regressão, que é o mapeamento de um item de dado em um valor real baseado em uma variável de predição;
c) o agrupamento (*clustering*), que é o mapeamento de um item de dado em um de vários agrupamentos, aonde os agrupamentos são naturalmente existentes no conjunto de dados por similaridades entre os atributos de seus itens ou por modelos de densidade de probabilidade;
d) a geração de regras, que é a extração de regras de classificação dos dados;
e) a descoberta de regras de associação, que descreve relacionamentos de associação entre diferentes atributos dos itens do conjunto de dados;
f) a sumarização, que é a descrição compacta de um subconjunto do conjunto de dados;

g) a modelagem de dependência, onde são feitas as descrições de dependências significantes entre as variáveis;
h) a análise de seqüências, para a geração de modelos de padrões seqüenciais, como análise de séries temporais. O objetivo é modelar os estados do processo de geração da seqüência ou extrair e reportar desvios e tendências dos dados, considerados sobre um intervalo de tempo.

A tarefa de classificação de dados em *data mining* e as técnicas computacionais utilizadas serão abordadas de forma mais destacada na subseção seguinte.

2.1.3 Classificação de dados em *data mining*

Conforme Kremer (1999, p. 21), a classificação é uma tarefa que consiste na aplicação de um conjunto de exemplos pré-classificados (ou seja, há um conhecimento *a priori* das classes) para desenvolver um modelo capaz de classificar uma população muito maior de dados. Em geral, algoritmos de classificação incluem árvores de decisão ou redes neurais artificiais, e começam com um treinamento a partir de exemplos. O algoritmo classificador usa estes exemplos para determinar um conjunto de parâmetros, codificados em um modelo, que será mais tarde utilizado para a discriminação do restante do conjunto de dados.

Uma vez que o algoritmo classificador foi desenvolvido de forma eficiente, ele será usado de forma preditiva para classificar novos registros naquelas mesmas classes pré-definidas (KREMER, 1999, p. 21). Alguns exemplos de classificação são:

a) classificar pedidos de crédito como de baixo, médio e alto risco;
b) esclarecer pedidos de seguro fraudulentos;
c) atribuir palavras chave a artigos jornalísticos.

A próxima subseção descreve algumas das técnicas computacionais utilizadas para a tarefa de classificação em *data mining*.

2.1.4 Técnicas computacionais para a tarefa de classificação em *data mining*

Para a tarefa de classificação de dados em *data mining*, é necessário produzir um modelo de classificação, um conjunto de dados para treinamento e outro para testes, que devem ser disjuntos. Para a implementação do modelo, algumas das técnicas das áreas da

estatística e da inteligência artificial mais comumente aplicadas encontram-se abaixo listadas (BRAGA, 2003, p. 6):

a) árvores de decisão e regras de produção;
b) regressão linear adaptadas para a classificação;
c) métodos de raciocínio baseados em casos (método do vizinho mais próximo);
d) métodos probabilísticos (Naive Bayes, redes probabilísticas Bayesianas);
e) regressão não-linear com redes neurais artificiais.

Ainda conforme Braga (2003, p. 6), tais métodos podem ser classificados em métodos paramétricos e não paramétricos. Nos métodos paramétricos, o modelo é fornecido pelo usuário, e a tarefa restringe-se a estimar os parâmetros (discriminantes lineares, redes Bayesianas, redes neurais). Nos métodos não-paramétricos, o modelo é construído pelo algoritmo com base na amostra do espaço de objetos (árvores de decisão, regras de produção). Nos parágrafos seguintes, as técnicas mais utilizadas são abordadas.

As árvores de decisão e regras de produção dividem os registros do conjunto de dados de treinamento em subconjuntos separados, cada um descrito por uma regra simples em um ou mais campos. Uma grande vantagem nestas técnicas é que o modelo é bem explicável, já que tem a forma de regras explícitas. Isto permite uma avaliação humana facilitada do resultado, pois identifica os atributos chave do processo (HARRISON, 1998, p. 54).

A regressão linear é um dos mais antigos métodos. Realiza classificação pela divisão do espaço amostral por uma ou várias séries de linhas retas em duas dimensões, as quais passam pela metade da distância dos centros de duas classes a separar, e cujas direções são determinadas pelas formas das classes (ROSA, 2003, p. 10).

O raciocínio baseado em casos é uma técnica que utiliza exemplos conhecidos para fazer previsões sobre exemplos desconhecidos. O raciocínio baseado em casos pode usar a técnica de busca dos vizinhos mais próximos nos exemplos conhecidos e combinar seus valores para atribuir valores de classificação ou de previsão (BERRY; LINOFF, 2004, p. 98).

As redes neurais artificiais são modelos simples de elementos interconectados, semelhantes ao cérebro biológico, adaptados para o uso em computadores. Elas aprendem com um conjunto de dados de treinamento, generalizando modelos para classificação, análise de seqüências, agrupamento, geração de regras e regressão linear e não linear. As redes neurais artificiais geralmente constroem superfícies equacionais complexas, através de repetidas iterações, ajustando os parâmetros que definem a superfície. Depois de muitas repetições, uma superfície que determina um possível padrão pode ser definida de forma que se aproxime de muitos pontos dentro do conjunto de dados (SANTOS, 2000, p. 60).

Uma das principais vantagens na utilização desta técnica é a sua variedade de aplicações. Elas são interessantes porque detectam padrões nos dados de forma análoga ao pensamento humano.

2.2 REDES NEURAIS ARTIFICIAIS

Esta seção apresenta as redes neurais artificiais, sendo abordadas suas definições, estruturas e modelos, além dos paradigmas e algoritmos de aprendizado das redes. Em particular são descritos o modelo de rede neural *multilayer perceptron* e o algoritmo de aprendizado *backpropagation*, por serem utilizados neste trabalho.

2.2.1 Definição de rede neural artificial

Segundo Abelém, Pacheco e Vellasco (1995, p. 109), redes neurais artificiais são sistemas inspirados nos neurônios biológicos e na estrutura massivamente paralela do cérebro, com a capacidade de adquirir, armazenar e utilizar conhecimento experimental. Esses sistemas são compostos de diversas unidades simples (os neurônios artificiais) ligados de maneira apropriada para obter comportamentos complexos e interessantes. O comportamento é determinado pela estrutura das ligações (topologia) entre os neurônios e pelos valores das conexões (pesos sinápticos).

Conforme Haykin (2001, p. 75), o conhecimento é adquirido pela rede através de um processo de aprendizado e através deste processo os pesos sinápticos são ajustados para armazenar o conhecimento desejado.

2.2.2 Estrutura de uma rede neural artificial

Uma rede neural artificial é um modelo matemático que consiste em várias unidades simples, semelhantes aos neurônios biológicos, denominados elementos de processamento. Estes elementos são organizados em camadas, onde cada elemento tem conexões ponderadas com outros elementos, sendo que estas conexões podem tomar diferentes configurações,

dependendo da aplicação particular desejada (LOESCH e SARI, 1996, p. 19).

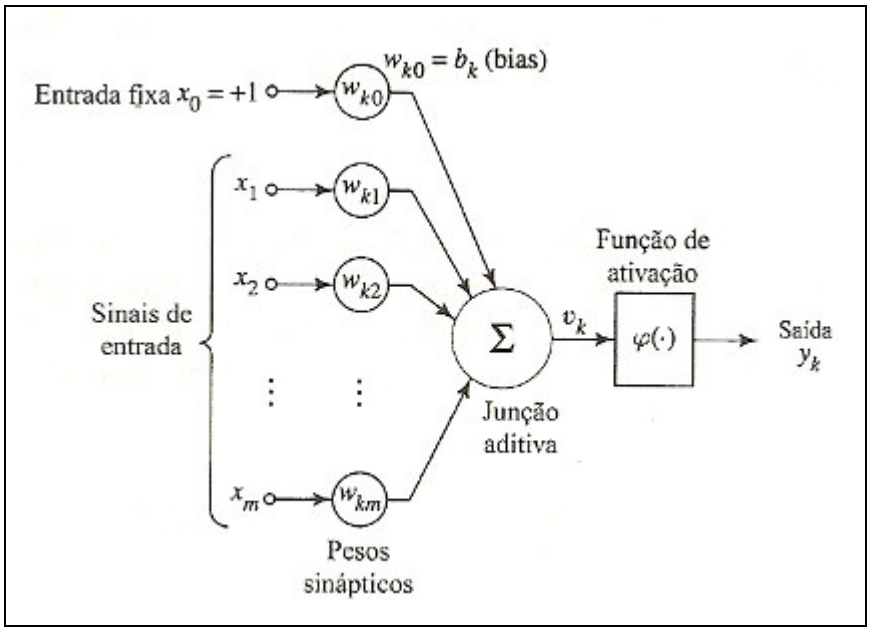
O neurônio é o elemento de processamento fundamental à operação da rede neural artificial. Na figura 2 é exibido um modelo para um neurônio onde é possível identificar três elementos básicos no modelo neural, descritos a seguir (HAYKIN, 2001, p. 36):

a) um conjunto de sinapses ou de conexões, sendo cada sinapse caracterizada por um peso. Especificamente um sinal x_j na entrada da sinapse j conectada ao neurônio k é multiplicado pelo peso sináptico w_{kj};

b) um somador para somar os sinais de entrada, já ponderados pelos pesos das respectivas sinapses dos neurônios;

c) uma função de ativação para limitar a amplitude do sinal de saída do neurônio.

O modelo exibido na figura 2 também inclui um *bias*, que é um parâmetro externo do neurônio, similar a uma sinapse, mas cuja entrada é fixa, com objetivo de aumentar o valor da entrada da função de ativação. De outra maneira, o valor da entrada da função de ativação pode ser diminuído, empregando um *threshold*, que seria o oposto de um *bias*. Em termos matemáticos, podemos descrever um neurônio pelos pares de equações descritos no quadro 1.



Fonte: adaptado de Haykin (2001, p. 38).

Figura 2 – Modelo básico de um neurônio

$$v_k = \sum_{j=0}^{m} w_{kj} x_j$$

e

$$y_k = \varphi(v_k)$$

onde

$x_1, x_2, ..., x_m$, são os sinais de entrada

$w_{k0}, w_{k1}, ..., w_{km}$ são os pesos sinápticos do neurônio k

v_k é a saída do combinador linear

$\varphi(v_k)$ é a função de ativação

y_k é o sinal de saída do neurônio

Fonte: adaptado de Haykin (2001, p. 38).

Quadro 1 – Pares de equações que descrevem um neurônio em termos matemáticos

A função de ativação define a saída do neurônio em termos do nível de atividade na sua entrada. Segundo Haykin (2001, p. 50), as funções mais encontradas na literatura são a função degrau, a função linear, a função sigmóide (logística) e a função tangente hiperbólica.

Os neurônios artificiais são agrupados em camadas e interconectados entre si, formando a rede neural. A estrutura completa, conforme Silva (1999, p. 20), envolveria então:

a) o número de camadas de neurônios;
b) o número de neurônios por camada;
c) os tipos de conexões entre os neurônios;
d) o grau de conectividade entre os neurônios.

Em geral, podem ser identificadas três classes diferentes de estruturas de redes: as redes alimentadas adiante (*feedforward*) de única camada, as redes alimentadas adiante com múltiplas camadas e as redes recorrentes. Nos próximos parágrafos, são descritas as definições das mesmas, conforme Haykin (2001, p. 46).

Nas redes alimentadas adiante de uma única camada, a organização dos neurônios ocorre em uma única camada de neurônios, onde temos uma camada de entrada que projeta os sinais de entrada em uma camada de neurônios de saída (que efetivamente fazem o processamento), mas que não há retorno para a camada de entrada.

As redes alimentadas adiante com múltiplas camadas se distinguem pela presença de camadas de neurônios intermediários, ou ocultos, entre a camada de entrada e a camada de saída da rede neural. A camada de entrada projeta os sinais de entrada sobre uma segunda

camada. Os sinais de saída dessa segunda camada são enviados para outras camadas subseqüentes e assim sucessivamente até encontrarem a camada de saída ou até, diretamente, para a camada de saída. Uma rede desse tipo pode ser totalmente conectada, onde todos os neurônios de uma camada se comunicam com todos os neurônios da próxima camada ou parcialmente conectada, onde nem todos os neurônios de uma camada se comunicam com todos os neurônios da próxima camada.

Nas redes recorrentes, a diferença de uma rede alimentada adiante é ocorrência de pelo menos um laço de realimentação, ou seja, os neurônios de uma camada apresentam a possibilidade de realimentarem outros neurônios da mesma camada ou de camadas anteriores. Essas redes podem ter também uma ou mais camadas ocultas e possuir ou não laços de auto-realimentação (a saída de um neurônio alimenta uma de suas próprias entradas).

2.2.3 O processo de aprendizado

Para que uma rede neural artificial adquira conhecimento, é necessário submeter a mesma a um aprendizado ou treinamento. Segundo Martinelli (1999, p. 13), essa aprendizagem ocorre através de um processo iterativo de ajustes de pesos aplicados aos pesos sinápticos. Esse ajuste é realizado através de um algoritmo de aprendizado seguindo algum paradigma de aprendizado.

Ainda conforme Martinelli (1999, p. 13), um algoritmo de aprendizado consiste de um conjunto de regras bem definidas para resolver um problema de aprendizagem. Os algoritmos diferem basicamente entre si pela forma de ajuste dos pesos e podem ser divididos em quatro classes principais:

a) aprendizado por correção de erro: os ajustes de pesos são efetuados de maneira a obter um erro mínimo;
b) aprendizado *Hebbiano*: o ajuste do peso de uma conexão entre dois neurônios ocorre somente quando estes neurônios estão simultaneamente ativos;
c) aprendizado competitivo: os neurônios competem entre si para serem ativados;
d) aprendizado de Boltzmann: é um algoritmo estocástico baseado em termodinâmica e teoria da informação.

Para uma rede neural artificial aprender o seu ambiente, são apresentados a ela exemplos desse ambiente. A maneira de apresentar esses exemplos para uma rede, ou seja, a interação existente entre o ambiente e uma rede neural artificial, é feita através de paradigmas

de aprendizado (MARTINELLI, 1999, p. 13). Haykin (1994, p. 85) considera duas classes básicas de paradigmas de aprendizado: aprendizado com um professor e aprendizado sem um professor.

No aprendizado com um professor, ou supervisionado, são apresentados os exemplos para a rede e a resposta da rede é avaliada em relação à resposta desejada (que é de conhecimento do professor). A diferença entre as duas respostas, conhecida como sinal de erro, é utilizada para ajustar os pesos sinápticos da rede. Esse procedimento é efetuado passo a passo até que a rede responda corretamente dentro de um senso estatístico (HAYKYN, 1994, p. 85).

Algoritmos de aprendizado sem um professor não requerem o conhecimento das saídas desejadas, ou seja, não são utilizados exemplos de entrada e saída a serem aprendidos pela rede, pois ela se auto-organiza (HAYKIN, 1994, p. 86). São destacados dois tipos de aprendizado, cada qual explicado nos próximos parágrafos: o aprendizado por sinal de reforço e o aprendizado não supervisionado propriamente dito.

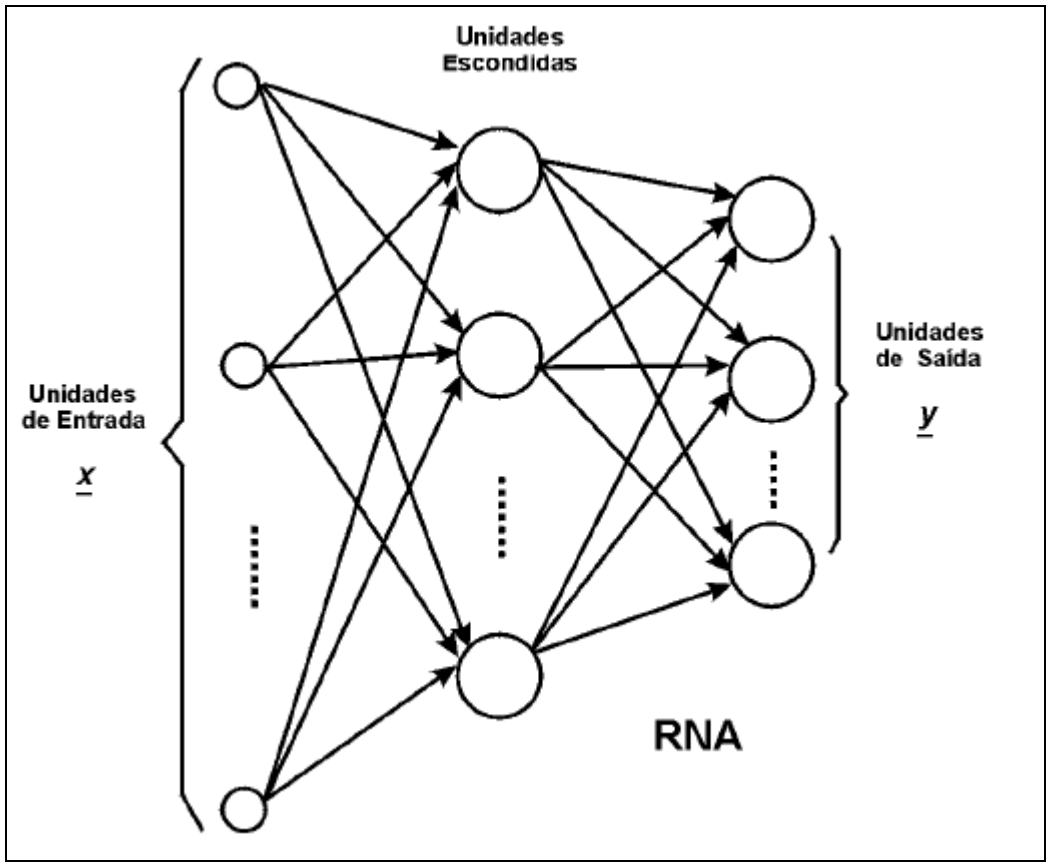
O aprendizado por sinal de reforço é efetuado via experimentação direta. O ambiente de aprendizagem é composto por dois elementos: o aprendiz e um processo dinâmico, onde, a passos de tempo sucessivos, o aprendiz faz uma observação do estado do processo, seleciona uma ação e aplica de volta ao processo. Sua meta é descobrir uma política de ações que controle o comportamento deste processo dinâmico, usando para isso sinais de reforço indicativos de quão bem está sendo executada a tarefa em questão. Estes sinais estão normalmente associados a alguma condição, como, por exemplo, a ação correta significa recompensa ou o fracasso significa castigo. A meta é aperfeiçoar seu comportamento baseado em uma medida de desempenho (função dos reforços recebidos) (BIANCHI, 2004, p. 8).

No aprendizado não supervisionado ou auto organizado, não há uma forma autônoma de avaliação do certo ou errado, como no caso do aprendizado por sinal de reforço. Ao invés disso, são dadas condições para realizar uma medida independente da qualidade da representação da tarefa que a rede deve aprender, e os parâmetros livres da rede são melhorados em relação a esta medida. Uma vez que a rede tenha se ajustado às regularidades estatísticas dos dados de entrada, ela desenvolve a habilidade de formar representações internas para codificar as características da entrada e, desse modo, criar automaticamente novas classes. (HAYKIN, 1994, p. 87)

2.2.4 O modelo de rede neural *multilayer perceptron*

Uma rede neural artificial do tipo *multilayer perceptron* (MLP) é constituída por um conjunto de nodos de entrada, os quais formam a camada de entrada na rede, de uma ou mais camadas ocultas e de uma camada de saída. Com exceção da camada de entrada, todas as outras camadas são constituídas por neurônios e, portanto, apresentam capacidade computacional. A rede MLP é uma generalização do modelo *perceptron* (HAYKIN, 2001, p. 183).

Um exemplo da arquitetura de uma rede neural MLP é exibido na figura 3, sendo que nesse exemplo a rede possui uma camada de entrada, uma camada oculta e uma camada de saída. Duas características são visíveis em tal estrutura: ela é uma rede alimentada adiante (*feedforward*) e pode ser parcialmente ou totalmente conectada.



Fonte: adaptado de Haykin (2001, p. 186).

Figura 3 – Arquitetura de uma rede neural artificial MLP

Segundo Castro e Castro (1994, p. 2), o número de nós na camada de entrada da rede é determinado pela dimensionalidade do espaço de observação, que é responsável pela geração dos sinais de entrada. O número de neurônios na camada de saída é determinado pela dimensionalidade requerida da resposta desejada. Assim, o projeto de uma rede MLP requer a

consideração de três aspectos: a determinação do número de camadas ocultas, a determinação do número de neurônios em cada uma das camadas ocultas e a especificação dos pesos sinápticos que interconectam os neurônios nas diferentes camadas da rede.

O primeiro e o segundo aspecto determinam a complexidade do modelo de RNA escolhido. A função das camadas ocultas da rede é a de influir na relação entrada-saída da rede de forma ampla. Uma RNA com uma ou mais camadas ocultas é apta a extrair as estatísticas de ordem superior de algum desconhecido processo subjacente, responsável pelo comportamento dos dados de entrada. A RNA adquire uma perspectiva global do processo aleatório, apesar de sua conectividade local, em virtude do conjunto adicional de pesos sinápticos e da dimensão adicional de interações neurais proporcionada pelas camadas ocultas.

O terceiro aspecto envolve a utilização de paradigmas e algoritmos de aprendizado supervisionado. Segundo Haykin (2001, p. 183) o algoritmo mais comumente usado é o de retropropagação do erro, conhecido na literatura como algoritmo *backpropagation*, que será abordado na próxima subseção.

2.2.5 O algoritmo de aprendizado *backpropagation*

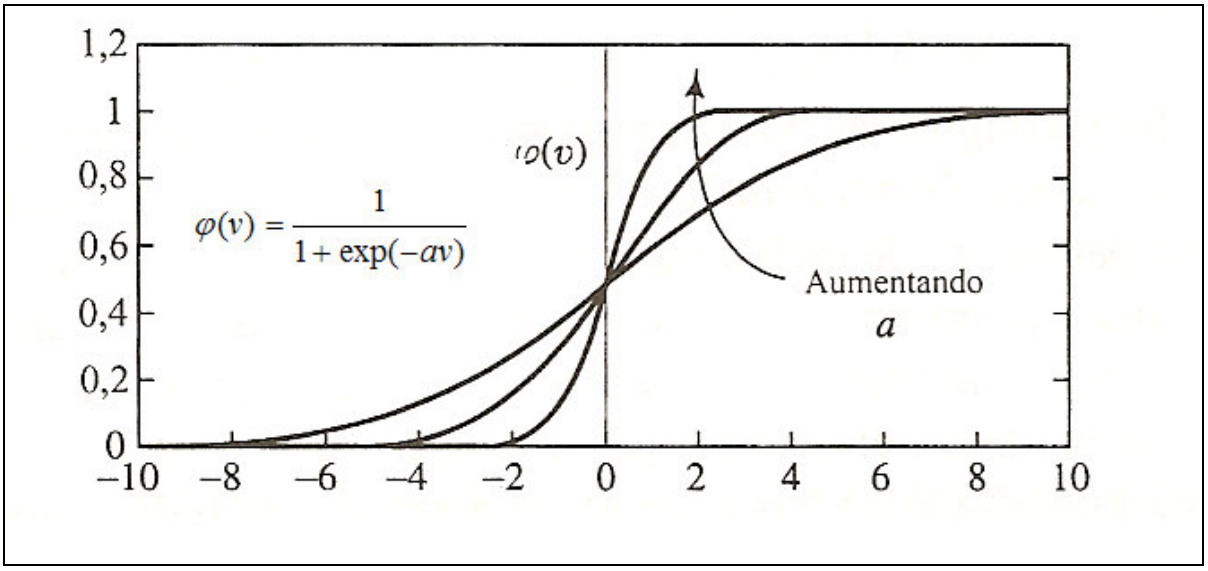
Segundo Haykin (2001, p. 183), o algoritmo de aprendizado *backpropagation* segue o paradigma do aprendizado supervisionado e baseia-se na heurística do aprendizado por correção de erro (em que o erro é retro-propagado da camada de saída para as camadas intermediárias da rede neural artificial). Este algoritmo pode ser considerado uma generalização do algoritmo dos mínimos quadrados médios (*least mean-square* – LMS), desenvolvido por Widrow e Hoff em 1960 e conhecido como regra delta.

Uma rede MLP apresenta três características distintas, de cuja combinação com a habilidade de aprender através da experiência (treinamento) deriva sua capacidade computacional (HAYKIN, 1994, p. 178):

a) o modelo de neurônio da rede MLP inclui uma função não-linear, cuja não linearidade é suave (ou seja, a função é diferenciável em qualquer ponto). Uma forma comumente usada de não-linearidade que satisfaz esse requisito é a não-linearidade sigmoidal definida pela função sigmóide logística, definida na figura 4;
b) a rede MLP contém uma ou mais camadas de neurônios ocultos que não são parte da camada de entrada ou de saída da rede. Estes neurônios ocultos possibilitam

que a rede aprenda tarefas complexas, extraindo progressivamente mais características significativas dos padrões de entrada;

c) a rede MLP exibe um alto grau de conectividade, determinado pelas sinapses da rede. Uma mudança na conectividade da rede requer uma mudança na população de conexões sinápticas.



Fonte: adaptado de Haykin (2001, p.39).

Figura 4 – Função sigmóide logística e seu gráfico

Segundo Han e Kamber (2000, p. 240), o algoritmo *backpropagation* aprende iterativamente pelo processamento de um conjunto de exemplos de treinamento, comparando a resposta da rede para cada exemplo com o rótulo de classe atualmente conhecido. Para cada exemplo de treinamento, os pesos são modificados de forma a minimizar o erro médio quadrático entre a saída da rede e a classe atual. Estas modificações são feitas na direção reversa, ou seja, da camada de saída através das camadas ocultas até as primeiras camadas. Daí o nome *backpropagation*, ou retro propagação. Ainda conforme Han e Kamber (2000, p. 241), o algoritmo sumarizado está descrito no quadro 2, e os passos principais do algoritmo estão resumidos nos próximos parágrafos.

Entradas: exemplos de treinamento, *exemplos*; taxa de aprendizado *l*; rede alimentada adiante, *rede*.
Saída: rede treinada para classificar novos exemplos.
Método:

1 inicializar os pesos da rede
2 **enquanto** o critério de parada não está satisfeito {
3 **para** cada exemplo de treinamento X em *exemplos* {
4 *// propagar o sinal de entrada para frente*
5 **para** cada neurônio oculto ou neurônio de saída *k*
6 $v_k = \sum_{j=1}^{m} w_{kj} x_j + b_k$ // computar a entrada do neurônio k
7 **para** cada neurônio oculto ou neurônio de saída k
8 $y_k = \frac{1}{1 + \exp(-av_k)}$ *// computar a saída do neurônio k*
9 *// retropropagar o sinal de erro*
10 **para** cada neurônio *k* na camada de saída
11 $e_k = v_k(1 - v_k)(t_k - v_k)$ *// computar o erro para o neurônio k*
12 **para** cada neurônio *k* nas camadas ocultas
13 $e_k = v_k(1 - v_k)\sum_{j=0}^{m} e_j w_{jk}$ *// computar o erro para o neurônio k*
14 **para** cada peso w_{kj} na rede {
15 $\Delta w_{kj} = ae_k v_k$ *// incremento do peso*
16 $w_{kj} = w_{kj} + \Delta w_{kj}$ *// atualização do peso*
17 }
18 }
19}

Fonte: adaptado de Han e Kamber (2000, p. 242).

Quadro 2 – Pseudocódigo do algoritmo *backpropagation*

O primeiro passo é inicializar os pesos de cada conexão entre os neurônios com pequenos valores aleatórios, geralmente entre 0 e 1 ou -1 e 1. O segundo passo é feito para cada exemplo de treinamento no formato de vetor `[X(k), D(k)]`, onde então o vetor `X(k)` é apresentado à camada de entrada e o vetor `D(k)` é apresentado à camada de saída.

O terceiro passo é a propagação para frente do sinal de entrada. Os valores do exemplo de treinamento são propagados da camada de entrada para a primeira camada oculta, e a saída

de cada neurônio dessa camada oculta é repassado a cada neurônio das próximas camadas, até atingir a camada de saída. A entrada de cada neurônio na camada oculta é uma combinação linear das saídas dos neurônios da camada anterior. Em um neurônio oculto ou de uma camada de saída, a entrada desta unidade é computada multiplicando cada valor da conexão pelo peso respectivo da conexão. Ao final, todos esses valores são somados e o valor assim obtido é aplicado a uma função de ativação, como, por exemplo, a função sigmóide logística.

O quarto passo é o cálculo do erro entre a saída desejada e a resposta da rede, e a retro propagação desse erro para as camadas ocultas até atingir a camada de entrada. No quadro 3 são demonstrados os cálculos de erro para as camadas de saída e para as camadas ocultas da rede. Na camada de saída, o erro é dado considerando o valor da saída real do neurônio contra o valor desejado, previamente conhecido pelo rótulo da classe do atual exemplo de treinamento. Para as camadas de ocultas, o erro é calculado considerando os pesos somados dos erros dos neurônios da próxima camada conectados ao neurônio da camada atual. Ambos os cálculos usam a derivada da função de ativação.

Cálculo dos erros dos neurônios no algoritmo backpropagation:

O erro em um neurônio na camada oculta é dado por

$$e_k = v_k(1-v_k)(t_k - v_k)$$

Onde:

e_k é o erro do neurônio de saída *k*

v_k é a saída atual do neurônio de saída *k*

$v_k(1-v_k)$ é a derivada da função logística

t_k é o valor da saída esperada do neurônio, baseada no valor conhecido do rótulo da classe do atual exemplo de treinamento

$(t_k - v_k)$ é a diferença entre a saída esperada da rede a saída atual

O erro de um neurônio oculto *k* considera os pesos somados dos erros dos neurônios da próxima camada, conectados ao neurônio *k*:

$$e_k = v_k(1-v_k)\sum_{j=0}^{m} e_j w_{jk}$$

Onde:

w_{jk} é o peso da conexão do neurônio k com o neurônio j da próxima camada

e_j é o erro na unidade j conectada

Fonte: adaptado de Han e Kamber (2000, p. 242).

Quadro 3 – Equações para cálculo do erro dos neurônios nas camadas de saída e ocultas

Nesse quarto passo ainda, é feita a atualização dos pesos das conexões do neurônio da

camada atual com os neurônios da próxima camada, para refletir os erros propagados. A atualização de pesos utiliza-se de duas equações, onde uma equação calcula o incremento ou o decremento a ser aplicado ao peso da conexão conforme o erro já calculado. A outra equação atualiza o peso com o valor encontrado. As equações estão dispostas no quadro 4.

Cálculo da atualização dos pesos das conexões dos neurônios baseados nos erros calculados, no algoritmo backpropagation:

O incremento do peso das conexões é calculado por:

$$\Delta w_{kj} = a e_k v_k$$

Onde:

e_k é o erro calculado para o neurônio *k*

v_k é a saída atual do neurônio *k*

Δw_{kj} é o valor de atualização do peso da conexão do neurônio k ao neurônio *j*

A atualização do peso das conexões do neurônio k ao neurônio j é feita por:

$$w_{kj} = w_{kj} + \Delta w_{kj}$$

Onde:

w_{kj} é o peso da conexão do neurônio *k* ao neurônio *j*

Fonte: adaptado de Han e Kamber (2000, p. 242).

Quadro 4 – Par de equações para cálculo e atualização do peso da conexão de um neurônio

Na equação que calcula o incremento ou decremento do peso da conexão existe uma constante que representa a taxa de aprendizado, que tipicamente possui um valor entre 0 e 1. O algoritmo *backpropagation* usa um método de gradiente descendente para encontrar um conjunto de pesos que pode modelar o corrente problema de classificação de forma a minimizar a distância quadrática média entre a saída atual da rede e a saída indicada no vetor atual de treinamento. A taxa de aprendizado é utilizada para que não ocorra um mínimo local no espaço de decisão (ou seja, a rede converge, mas não para a solução ótima) e auxiliar a encontrar o mínimo global.

O quinto e último passo é avaliar a condição de parada do treinamento. Cada apresentação de todos os exemplos de treinamento é chamada de época. Em cada época pode ser calculado um erro médio global (soma de todos os erros da época). Assim, alguns critérios de parada podem ser assumidos como, por exemplo, quando o erro médio global de uma época ultrapassa um mínimo estabelecido, ou o percentual de erros de classificação ultrapassa

também um mínimo estabelecido ou ainda se certo número de épocas foi executado.

Haykin (2001, p. 196), descreve que pode ser utilizada ainda um segunda variável além da taxa de aprendizado no melhoramento do desempenho da rede durante o treinamento. Trata-se do termo de momento, que está relacionado com a idéia de que as mudanças anteriores nos pesos têm influência na direção do ajuste atual. Com o termo de momento, uma vez que os pesos começam a ir para uma direção, incentiva-se que eles continuem indo na mesma direção. Ele pode ajudar a rede a não cair em mínimos locais, assim como pode aumentar a velocidade do aprendizado.

Ainda segundo Han e Kamber (2000, p. 242), existem muitas variantes e alternativas ao algoritmo *backpropagation*. Tais alternativas envolvem o uso de ajustes dinâmicos na topologia da rede, da alteração dinâmica da taxa de aprendizado, uso de diferentes funções de ativação ou funções de cálculo do erro. Segundo Haykin (2001, p. 273), o algoritmo surgiu como o método padrão para o treinamento das redes MLP. O nome do algoritmo deriva do fato de que as derivadas parciais da função de custo (medida de desempenho) em relação aos parâmetros livres (pesos sinápticos) da rede são determinadas por retro propagação dos sinais de erro a partir das camadas de saída para as camadas anteriores, camada a camada. O poder computacional do algoritmo se baseia nos seguintes pontos:

a) o uso de um método local para adaptar seus pesos sinápticos;
b) o uso de um método eficiente para calcular as derivadas parciais das funções de custo em relação aos pesos sinápticos.

2.3 REDES NEURAIS ARTIFICIAIS E O PROBLEMA DA COMPREENSIBILIDADE

A habilidade para aprender a partir de exemplos é uma importante faceta da inteligência e uma promissora área de estudos para pesquisadores em inteligência artificial, estatística, ciência cognitiva e demais campos relacionados. Algoritmos que conseguem aprender de maneira indutiva a partir de exemplos têm sido aplicados a numerosos problemas do mundo real (LEHR; RUMELHART; WIDROW, 1994, p. 93; LANGLEY; SIMON, 1995, p. 54).

Segundo Craven (1996, p. 1), métodos de aprendizado indutivo podem aprender e executar tarefas que não podem ser implementadas explicitamente, ou tal tarefa é complexa para implementar e despende muito tempo. Outra razão para se utilizar de mecanismos de

aprendizado indutivo é o propósito de adquirir conhecimento sobre determinado conjunto de dados, pelo fato de tais algoritmos montarem um modelo descritivo desse conjunto de dados. Em muitos casos esses modelos construídos são humanamente compreensíveis e oferecem um melhor conhecimento do domínio do problema. Aprendizado indutivo com foco na compreensibilidade é a atividade principal do campo da descoberta de conhecimento em bancos de dados e *data mining* (FAYYAD et al., 1996, p. 5). Naturalmente, são freqüentes os casos em que se aplicam métodos de aprendizado em um dado domínio para atingir dois objetivos: construir um sistema para executar determinada tarefa e obter um entendimento melhor dos dados disponíveis.

Um dos diversos paradigmas do aprendizado indutivo são as redes neurais artificiais (LANGLEY e SIMON, 1995, p. 55), que têm provado ser uma ótima técnica nas aplicações que envolvem o aprendizado de máquina. Entretanto, Rezende (2003, p. 257) comenta que as redes neurais têm certas deficiências, sendo a mais significante delas a de que uma rede neural artificial é essencialmente uma caixa preta. Devido a isto, determinar exatamente como uma rede neural artificial toma uma decisão particular é uma tarefa complexa. Como a rede neural não tem a habilidade de gerar explicações sobre suas decisões, torna-se difícil aceitá-las.

Rezende (2003, p. 257), afirma ainda que, para várias das aplicações que envolvem redes neurais artificiais, é importante não apenas o desempenho obtido, mas também a facilidade para o usuário compreender como a rede alcança suas decisões. Mais especificamente, quando um método é utilizado para a classificação, pode ser importante para o usuário entender como e porque um dado padrão de entrada foi classificado em uma determinada classe.

Conforme Craven (1996, p. 4), a compreensibilidade de modelos de inferência baseados em técnicas de aprendizado de máquina (na qual se encaixam as redes neurais artificiais) é um critério muito importante. Isto é, um algoritmo de aprendizado por indução deve codificar o seu modelo de tal maneira que possa ser inspecionado e compreendido por seres humanos.

> Os resultados da indução devem ser descrições simbólicas de entidades dadas, semântica e estruturalmente similares a aquelas que um especialista humano pode produzir observando as mesmas entidades. Os componentes destas descrições devem ser compreensíveis como os únicos pedaços da informação, diretamente interpretados em linguagem natural, e devem relacionar conceitos quantitativos e qualitativos de uma forma integrada. (MICHALSKI, 1983, p. 122).

Darbari (2001, p. 992) explica que uma hipótese aprendida por uma rede neural artificial é definida pela topologia da rede, pelas funções de transferência usadas pelas unidades ocultas e de saída e pelos parâmetros numéricos (pesos) associados com as

conexões. Tais hipóteses são difíceis de entender por diversas razões. Primeiramente, redes neurais tipicamente possuem muitas conexões. Os pesos dessas conexões codificam o relacionamento entre os dados de entrada e o resultado da saída. Embora um único parâmetro dessa codificação não seja difícil de entender, o conjunto completo das ligações e seus parâmetros tornam a tarefa de entendimento muito complexa. Além disso, em redes multicamadas, os parâmetros das ligações podem representar relações não-lineares e não-monotômicas entre os valores de entrada e de saída. Esse efeito ocorre nos neurônios ocultos e é causado pela combinação dos diversos valores nas conexões de entradas dos mesmos, durante o treinamento. Essa é uma vantagem do modelo neural, que assim aprende as dependências entre esses valores, porém, o conhecimento fica distribuído por toda a rede.

Se o objetivo de uma determinada tarefa é obter compreensão em um determinado domínio, então a aplicação das redes neurais a esse problema não seria propícia. Ao invés disso, o uso de métodos que produzem modelos que são mais fáceis para o entendimento humano seria a melhor escolha. Porém, para determinados problemas, as redes neurais artificiais são um melhor método para aprendizado do que os demais métodos (CRAVEN, 1996, p. 8).

Em alguns casos, redes neurais restringem melhor o espaço de uma hipótese do que outros algoritmos. Por exemplo, o algoritmo *Q-Learning* para aprendizado com reforço requer que o método de aprendizado represente hipóteses sobre funções de valores contínuos, e requer que esta hipótese seja atualizada para cada exemplo de treinamento. Poucos algoritmos de aprendizado simbólico atendem a esses requisitos. Tarefas de predições seqüenciais e temporais são outros tipos de problemas onde as redes neurais artificiais são mais apropriadas, pois fornecem a melhor hipótese sobre o espaço do problema (CRAVEN, 1996, p. 8).

Em outros casos, as redes neurais artificiais são o método preferido de aprendizado não devido às classes de hipóteses que elas podem representar, mas simplesmente por que elas induzem hipóteses cuja generalização é melhor do que a generalização obtida outros algoritmos. Diversos estudos empíricos têm demonstrado que existem alguns tipos de problemas onde as redes neurais artificiais obtêm um nível superior de acurácia preditiva em relação aos mais comuns métodos de aprendizado simbólico (MOONEY; SHAVLIK; TOWEL, 1991, p. 112).

Por outro lado, Thrun (1993, p. 2), afirma que muitos sistemas de aprendizado simbólico baseados em regras, oferecem um alto nível de compreensibilidade, devido à natureza simples das regras simbólicas. Procedimentos de aprendizado simbólico procuram gerar conjuntos de regras pequenos e simples, extraídas do conjunto de treinamento. Tais

regras são interpretadas mais facilmente por humanos. Além disso, as regras simbólicas permitem uma fácil interação com vários sistemas baseados em conhecimento, como sistemas especialistas. Se as redes neurais artificiais são o método escolhido para uma determinada tarefa, mecanismos que expliquem as hipóteses aprendidas pela rede oferecem uma promissora perspectiva para superar a óbvia deficiência das mesmas em oferecer explanação sobre o que elas generalizaram.

Tais mecanismos que expliquem as hipóteses aprendidas pela rede, que são as técnicas de extração de regras de redes neurais, são o tema da próxima seção, que procura abordar de forma mais aprofundada esse assunto.

2.4 EXTRAÇÃO DE REGRAS DE REDES NEURAIS ARTIFICIAIS

Nesta seção, é apresentado o tema da extração de regras de redes neurais artificiais. É feito um destaque sobre a definição do termo extração de regras de redes neurais. Destacam-se em seqüência as vantagens de se utilizar a extração de regras de redes neurais, uma taxonomia para a classificação das técnicas, e uma breve informação sobre algumas técnicas existentes. Também apresenta uma descrição das técnicas a serem analisadas e implementadas neste trabalho.

2.4.1 Definições para extração de regras de redes neurais artificiais

Um meio de entender as hipóteses geradas por uma rede neural artificial treinada é a tradução da hipótese em uma linguagem mais compreensiva. Vários métodos que utilizam esta estratégia têm sido estudados sobre o nome de extração de regras. Nesse contexto, uma definição de extração de regras de redes neurais artificiais é de que, dada uma rede neural artificial treinada e os exemplos utilizados para treiná-la, produz-se uma descrição simbólica da hipótese gerada pela rede que seja próxima do comportamento da rede (CRAVEN, 1996, p. 9).

Conforme Rezende (2003, p. 276), é importante chamar a atenção sobre a distinção entre extração de conhecimento e extração de regras de redes neurais artificiais. Quando técnicas de extração de conhecimento são utilizadas, o conhecimento obtido não precisa

necessariamente estar na forma de regras, embora isto aconteça na maioria dos casos. Outras modalidades de conhecimento podem ser extraídas, como valores estatísticos, protótipos de agrupamento e outras. A extração de regras de redes neurais artificiais é um caso especial de extração de conhecimento, porém a maioria dos trabalhos encontrados na literatura de extração de conhecimento referencia os algoritmos de extração de regras.

A importância da extração de regras das redes neurais artificiais há tempos é reconhecida e existe uma variedade de técnicas disponíveis. Algumas são classificadas de acordo com os tipos de regras geradas, outras dependem de detalhes de construção e da arquitetura da rede neural, ou dos algoritmos de treinamento utilizados (THRUN, 1993, p. 2).

2.4.2 Vantagens da extração de regras de redes neurais artificiais

Desde que a extração de regras de redes neurais artificiais tem um custo em termos de recursos e esforço adicional, é importante que primeiramente sejam verificadas algumas razões de por que a extração de regras é uma importante extensão da técnica de redes neurais artificiais. Conforme Andrews, Diederich e Tickle (1995, p. 374), os méritos de incluir técnicas de extração de regras como suplemento para técnicas de redes neurais artificiais apresentam diversas vantagens, as quais são relatadas nos próximos parágrafos.

A primeira vantagem é a provisão da capacidade de explicação. Dentro do campo da Inteligência Artificial (IA) simbólica, o termo "explanação" se refere a uma estrutura explícita que pode ser internamente usada para o próprio aprendizado, e externamente para prover explicações para um usuário. Usuários de sistemas de IA simbólicos se beneficiam de uma representação de forma declarativa do conhecimento sobre o domínio de problema, tipicamente na forma de hierarquias de objeto, redes semânticas, blocos estruturais, etc. A capacidade de explicação de um sistema de IA simbólica também inclui os passos intermediários do processo de raciocínio (ou seja, um rastreamento dos passos executados, uma estrutura de prova, etc.), cujo objetivo é responder a questão de como se chega à hipótese gerada pelo sistema. Experiências têm mostrado que a capacidade de explicação é considerada como uma das funções mais importante oferecida por sistemas de IA simbólicos.

Em particular, a utilização de sistemas baseados em conhecimento que possuem a capacidade de gerar explicações (em termos de significância e coerência) é absolutamente crucial para a aceitação de tais sistemas pelos usuários. Em contraste com sistemas de IA simbólicos, as redes neurais artificiais não apresentam nenhuma representação de

conhecimento de forma explícita e declarativa. Há uma dificuldade considerável em gerar as estruturas de explicação exigidas. Torna-se aparente que a ausência de capacidade da explicação em sistemas de redes neurais artificiais limita a utilização completa do potencial de tais sistemas. É tal deficiência que o processo de extração de regras busca superar.

Além da provisão de capacidade da explicação para redes neurais artificiais, outra vantagem do uso de técnicas de extração de regras das mesmas é que com isso as redes neurais artificiais podem ser utilizadas em domínios de problemas de missão crítica. Como um exemplo, deve ser possível ao especialista de domínio determinar ou não se a rede neural treinada tem uma estrutura ou tamanho suficiente para o problema em questão. Então, um requisito para soluções de redes neurais é que os estados internos do sistema possam ser interpretados sem ambigüidades. A satisfação de tal requisito contribui de forma significante para a tarefa de identificar as soluções de redes que têm o potencial de oferecer resultados errôneos, permitindo o acompanhamento ou indicação de quando e por que um resultado não é apropriado. Tal capacidade é obrigatória para que as soluções baseadas em redes neurais artificiais sejam aceitas em determinadas áreas de aplicações de missão crítica, como, por exemplo, o controle de tráfego aéreo, a operação de sistemas de energia e cirurgia médica. A extração de regras oferece essa possibilidade.

A verificação e depuração de componentes de redes neurais artificiais integrados em outros sistemas de software é outra vantagem citada**.** A tarefa de verificação de software é considerada importante, mas é reconhecida como sendo difícil, particularmente para grandes sistemas. Se redes neurais artificiais são integradas dentro de outros sistemas de software que necessitam de verificações, conseqüentemente este requisito deve ser encontrado também na rede neural. Os algoritmos de extração de regras não provam que uma rede neural se comporte conforme certa especificação, mas fornecem um mecanismo para que se entenda a rede neural treinada, de forma parcial ou completa. Isto é visto como um meio promissor para, pelo menos indiretamente, alcançar a meta exigida, habilitando uma possível comparação a ser feita entre as regras extraídas e a especificação de software onde a rede neural está integrada.

A melhoria da generalização em soluções de redes neurais artificiais é também uma vantagem da utilização da extração de regras. Especialmente quando são utilizados dados limitados ou não representativos para treinar as redes neurais em um determinado problema, é difícil de determinar se o desempenho da generalização da rede atinge o ponto ideal. Sendo capaz de expressar o conhecimento embutido dentro da rede neural treinada como um conjunto de regras simbólicas, o processo de extração de regras pode fornecer ao especialista

de domínio a capacidade para antecipar ou predizer um conjunto de circunstâncias sob a qual a generalização da rede pode falhar. Alternativamente o especialista pode ser capaz de usar as regras extraídas para identificar regiões no espaço de entrada que não são suficientemente representadas nos dados de treinamento atuais para a rede neural e providenciar o suplemento desses dados.

Outra vantagem importante é a possibilidade de exploração de dados e a indução de teorias científicas. As redes neurais artificiais têm provado serem ferramentas com papel importante na exploração de dados, com a capacidade para descobrir previamente desconhecidas dependências e relações em dados. Craven e Shavlik (1994, p. 37) observam que um sistema de aprendizado pode descobrir certas características nos dados de entrada cuja importância não era previamente reconhecida. Porém, ainda que uma rede neural treinada aprenda relações interessantes e possivelmente não-lineares, estas relações são codificadas de forma incompreensível como vetores de pesos e conseqüentemente não podem facilmente servir para a geração de teorias científicas. Os algoritmos de extração de regras significativamente melhoram as capacidades das redes neurais artificiais na exploração de dados para o benefício do usuário.

A última vantagem apresentada é a aquisição de conhecimento para sistemas de IA simbólicos. Uma das principais razões da introdução de algoritmos de aprendizagem de máquina é a necessidade de superar o problema da aquisição de conhecimento para sistemas de IA simbólicos. A maior dificuldade e o maior consumo de tempo em construir um sistema especialista é construir e depurar seu conhecimento básico. As redes neurais artificiais treinadas são utilizadas então como um meio para sintetizar o conhecimento que é crucial para o sucesso de sistemas de conhecimento baseados em casos. A vantagem principal é que as técnicas extraem regras simbólicas das redes neurais treinadas e estas regras podem ser diretamente adicionadas à base de conhecimento. Alternativamente, o conhecimento de domínio que é adquirido por um processo de engenharia de conhecimento pode reduzir o tamanho do espaço de procura durante a fase de aprendizado e conseqüentemente contribuir para melhorar o desempenho nessa fase.

2.4.3 Classificação das técnicas de extração de regras

Uma das propostas mais rigorosas e sistemáticas para desenvolver uma taxonomia coerente e consistente para classificar técnicas de extração de regras de redes neurais

artificiais foi proposta por Andrews, Diederich e Tickle (1995, p. 379). Os termos dessa proposta são: (a) o formato das regras extraídas; (b) a transparência da técnica de extração de regra das unidades da RNA; (c) a portabilidade entre diversas arquiteturas ou regimes de treinamentos de redes neurais artificiais; (d) a qualidade das regras extraídas; e (e) a complexidade do algoritmo. Os próximos parágrafos descrevem de uma forma resumida cada critério proposto pelos autores da mesma.

O primeiro critério de classificação indicado é o formato em que as regras extraídas são apresentadas. Tal formato indica a força de expressão das regras extraídas. Sob esta ótica, existem três formatos mais comuns, onde o primeiro formato é o convencional (booleano ou proposicional), que apresenta regras em lógica simbólica no formato `se...então...senão...`. Outro formato utiliza os conceitos da lógica *fuzzy*. Este formato permite expressar regras também na forma `se...então...senão...`, porém tais regras usam o conceito de verdades parciais, conforme o contexto onde estão inseridas. O terceiro formato é o das regras expressas em lógica de primeira ordem, ou seja, regras com quantificadores e variáveis.

O segundo critério de classificação proposto é em relação à transparência. O termo transparência é usado para revelar qual nível da estrutura da rede neural os algoritmos de extração de regras analisam para extrair as regras. Tais níveis são encaixados em duas categorias básicas: a decomposicional e a pedagógica. A técnica decomposicional analisa a arquitetura interna da rede neural treinada, avaliando as relações entre os neurônios e os pesos das conexões. Muito comumente dependem da própria arquitetura da rede e do algoritmo de treinamento. Uma característica básica das técnicas dessa categoria é que a saída de cada unidade oculta da rede é transformada em um formato binário (sim ou não) e corresponde à noção de conseqüente de uma regra. Cada unidade oculta pode ser interpretada como uma pequena regra e as regras extraídas de cada unidade são agrupadas para compor uma ou várias regras que representam a rede como um todo. A transparência, nesse caso, é total, pois a rede é analisada e decomposta internamente, dando sentido ao termo decomposicional.

A segunda categoria dentro do critério da transparência é a pedagógica. Nessa categoria, as regras são extraídas através da análise dos relacionamentos entre os dados de entrada e a resposta fornecida pela rede a essas entradas, sem analisar as características estruturais da rede. Nesse caso, a técnica é usada juntamente com um algoritmo de aprendizado simbólico que utiliza a rede para fornecer as respostas e assim deve aprender o conhecimento adquirido pela rede, o que justifica o nome da categoria. Para este algoritmo, a rede neural continua sendo uma caixa-preta, ou seja, não há visão do interior.

As duas categorias de transparência nem sempre conseguem classificar corretamente todos os métodos de extração de regras, pois esses métodos podem se constituir de combinações das duas categorias, com participação menor ou maior de uma das categorias. Ou seja, em certa fase, a técnica pode analisar a estrutura interna da rede enquanto que em outra fase, ou mesmo paralelamente, analisa o relacionamento entrada-saída da rede, sem se preocupar com a estrutura. Para esses algoritmos, é adotada então uma terceira categoria derivada chamada eclética, que indica uma combinação das duas categorias básicas.

O terceiro critério de classificação aplica-se à questão da portabilidade de uma técnica em relação às características da rede em que ela será aplicada. Ou seja, o quanto uma técnica pode ser aplicada em diferentes arquiteturas de redes ou diferentes regimes de treinamento da rede. A principal medida nesse caso é a quantidade de adaptações que são necessárias para aplicar uma determinada técnica a uma determinada arquitetura de rede, ou mesmo se é possível aplicar a técnica na rede. Alguns algoritmos são dependentes da arquitetura ou do algoritmo de treinamento, e por isso nem sempre podem ser aplicados em outras arquiteturas ou regimes de treinamento. O critério da transparência geralmente está atrelado à portabilidade, pois quanto mais decomposicional é uma técnica, menos portável ela se torna e quanto mais pedagógica, mais portável ela se torna.

O quarto critério de classificação proposto é a qualidade das regras extraídas. Critérios para avaliar a qualidade das regras extraídas incluem a precisão, fidelidade, consistência e compreensibilidade. Neste contexto as regras são consideradas como precisas se classificam corretamente exemplos previamente não vistos. De forma semelhante, um conjunto de regras é considerado de alto nível de fidelidade se pode imitar o comportamento da RNA da qual foram extraídas. A consistência é a forma de verificar se, sob sessões de treinamento diferentes, são geradas regras que produzem as mesmas classificações de exemplos não vistos. Finalmente, a compreensibilidade de um conjunto de regras é determinada medindo o tamanho da regra em termos do número de regras e o número de antecedentes por regra.

Comumente, o âmbito na comparação das várias técnicas usando o critério de qualidade de regra é limitado. Em muitos domínios de problema do mundo real a aplicação simultânea de todos os critérios de avaliação pode não ser sempre desejável. Por exemplo, em aplicações de missão crítica é imperativo que a rede neural artificial seja validada sob todas as condições de entradas possíveis. Isto pode criar uma situação, por exemplo, onde a compreensibilidade pode não ser o requisito principal, e sim a precisão e fidelidade. O enfoque no assunto de qualidade de regra em geral é sobre a sintaxe de regra e não na semântica de regra.

O critério final no esquema de classificação é a complexidade da técnica de extração de regras. A inclusão de tal critério reflete um requisito quase universal para os algoritmos que fornecem suporte ao processo de extração de regras, que é a questão da eficiência. Um assunto crucial no desenvolvimento de uma técnica de extração de regra é o de como reduzir o tamanho do espaço de busca da solução. Porém, essa característica (a de complexidade) não é comumente reportada pelos autores dos algoritmos.

2.4.4 Algoritmos de extração de regras de redes neurais artificiais existentes

Existe na literatura uma grande quantidade de algoritmos de extração de regras. No quadro 5 são listados os mais conhecidos e algumas de suas características, conforme Santos (2001, p. 16) .

Algoritmo	Características			
KT	**Autor(es):**	Li Min Fun		
	Ano Publicação:	1991	**Abordagem:**	Decomposicional
	Formato das Regras:	Proposicional (se...então...)		
	Qualidade:	Não disponível		
	Portabilidade:	Aplicável a redes multicamadas alimentadas adiante		
	Complexidade:	Não disponível		
SUBSET	**Autor(es):**	Geoffrey Towell e Jude Shavlik		
	Ano Publicação:	1993	**Abordagem:**	Decomposicional
	Formato das Regras:	Proposicional (se...então...)		
	Qualidade:	Não disponível		
	Portabilidade:	Aplicável a redes multicamadas alimentadas adiante		
	Complexidade:	Não disponível		
RULENET	**Autor(es):**	Clayton McMillan, Michael C. Mozer e Paul Smolensky		
	Ano Publicação:	1991	**Abordagem:**	Decomposicional
	Formato das Regras:	Proposicional (se...então...)		
	Qualidade:	Não disponível		
	Portabilidade:	Aplicável a redes multicamadas alimentadas adiante, porém requerem algoritmo de treinamento específico		
	Complexidade:	Não disponível		
MOFN	**Autor(es):**	Geoffrey Towell e Jude Shavlik		
	Ano Publicação:	1993	**Abordagem:**	Decomposicional
	Formato das Regras:	M de N, ou Se (M dos seguintes N antecedentes for verdadeiro) então conseqüente		

	Qualidade:	Não disponível
	Portabilidade:	Aplicável a redes multicamadas alimentadas adiante, mas requer um algoritmo de treinamento chamado *constrained error for backpropagation*
	Complexidade:	Não disponível
RULEX	**Autor(es):**	Robert Andrews e Shlomo Geva
	Ano Publicação:	1994 **Abordagem:** Decomposicional
	Formato das Regras:	Proposicional (se...então...)
	Qualidade:	Não disponível
	Portabilidade:	Aplicável a redes multicamadas alimentadas adiante
	Complexidade:	Não disponível
COMBO	**Autor(es):**	R. Krishnan
	Ano Publicação:	1996 **Abordagem:** Decomposicional
	Formato das Regras:	Proposicional (se...então...))
	Qualidade:	As regras extraídas mostram alta fidelidade, acurácia e compreensibilidade
	Portabilidade:	Aplicável a redes multicamadas alimentadas adiante, mas requer que as entradas da rede sejam entradas binárias
	Complexidade:	Exponencial no pior caso
RX	**Autor(es):**	Hongjun Lu, Huan Liu e Rudy Setiono
	Ano Publicação:	1995 **Abordagem:** Decomposicional
	Formato das Regras:	Proposicional (se...então...senão)
	Qualidade:	Boa
	Portabilidade:	Aplicável a redes multicamadas alimentadas adiante
	Complexidade:	Não disponível
PARTIAL-RE	**Autor(es):**	Ismail A. Taha e Joydeep Ghosh
	Ano Publicação:	1999 **Abordagem:** Decomposicional
	Formato das Regras:	Proposicional (se...então...)
	Qualidade:	Não disponível
	Portabilidade:	Aplicável a redes multicamadas alimentadas adiante
	Complexidade:	Não disponível
FULL-RE	**Autor(es):**	Ismail A. Taha e Joydeep Ghosh
	Ano Publicação:	1999 **Abordagem:** Decomposicional
	Formato das Regras:	Proposicional (se...então...)
	Qualidade:	Não disponível
	Portabilidade:	Aplicável a redes multicamadas alimentadas adiante e independe do tipo de entrada da rede (nominal, binária ou contínua)
	Complexidade:	Não disponível
VIA	**Autor(es):**	Sebastian B. Thrun

	Ano Publicação:	1994	**Abordagem:**	Pedagógica
	Formato das Regras:	Proposicional (se...então...)		
	Qualidade:	Não disponível		
	Portabilidade:	Aplicada geralmente a redes multicamada alimentadas adiante, treinadas com o algoritmo *backpropagation*, mas é seu propósito é ser aplicado a qualquer modelo de rede		
	Complexidade:	Não disponível		
TREPAN	**Autor(es):**	Mark W. Kraven		
	Ano Publicação:	1996	**Abordagem:**	Pedagógica
	Formato das Regras:	Árvore de decisão, onde os nodos possuem regras do tipo M de N, ou Se (M dos seguintes N antecedentes for verdadeiro) então conseqüente		
	Qualidade:	As árvores extraídas apresentam alta acurácia, fidelidade e compreensibilidade		
	Portabilidade:	Aplicável a qualquer modelo de rede		
	Complexidade:	Polinomial no tamanho do exemplo, na dimensionalidade do espaço de entrada e também no número máximo de valores para uma característica discreta		
RULENEG	**Autor(es):**	R. Hayward, C. Ho-Stuart, Joaquim Diederich and E. Pop		
	Ano Publicação:	1994	**Abordagem:**	Pedagógica
	Formato das Regras:	Proposicional (se...então...)		
	Qualidade:	Não disponível		
	Portabilidade:	Aplicável a qualquer modelo de rede		
	Complexidade:	Não disponível		
DEDEC	**Autor(es):**	Alan B. Tickle, Joaquim Diederich e M. Orlowski		
	Ano Publicação:	1994	**Abordagem:**	Pedagógica
	Formato das Regras:	Proposicional (se...então...)		
	Qualidade:	Não disponível		
	Portabilidade:	Aplicável a qualquer modelo de rede		
	Complexidade:	Não disponível		
BIO-RE	**Autor(es):**	Ismail A. Taha e Joydeep Ghosh		
	Ano Publicação:	1999	**Abordagem:**	Pedagógica
	Formato das Regras:	Regras no formato se não...então...		
	Qualidade:	Não disponível		
	Portabilidade:	Aplicável a qualquer modelo de rede, porém requer que as entradas sejam binárias		
	Complexidade:	Não disponível		

Fonte: adaptado de Santos (2001, p. 16).

Quadro 5 – Alguns algoritmos de extração de regras existentes

2.4.5 Técnicas de extração de regras de redes neurais artificiais avaliadas

Nesta subseção são abordados com maior ênfase dois algoritmos de extração de regras de redes neurais artificiais utilizados neste trabalho. O objetivo é fornecer uma melhor compreensão dos mesmos, explicando o funcionamento e as características de cada algoritmo. Tais algoritmos são o RX e o TREPAN, descritos nas próximas subseções.

2.4.5.1 O algoritmo de extração de regras RX

O algoritmo de extração de regras RX (de *Rule eXtraction*) foi desenvolvido por Liu, Lu e Setiono (1995, p. 478) em 1995, como parte de um sistema de *data mining* denominado NEURORULE. Esse sistema foi desenvolvido para a tarefa de classificação em *data mining* e possui duas fases, sendo a primeira fase a de preparação de dados e treinamento de um rede neural artificial para classificação. A segunda fase trata da aplicação do algoritmo RX para a extração em formato de regras do conhecimento adquirido pela rede neural artificial treinada na primeira fase. A rede neural artificial utilizada em questão é uma rede MLP, treinada com o algoritmo *backpropagation*.

O algoritmo RX é um algoritmo da abordagem decomposicional e assim, necessita abordar a estrutura e dados internos da rede neural artificial para extrair as regras. As regras extraídas são regras no formato `se...então...`. Alves (2001, p. 19), descreve de forma sucinta os passos do algoritmo, enumerados abaixo:

a) processar as entradas pertencente ao conjunto de treinamento através da rede neural artificial já treinada e computando-se os valores de ativação dos neurônios da camada oculta;
b) aplicar um algoritmo de agrupamento (*clustering*) aos valores de ativação computados, a cada neurônio oculto, obtendo assim um número discreto de valores de ativação e mantendo a precisão original da rede neural dentro de um limite;
c) enumerar os valores de ativação discretos de cada neurônio oculto e combinar esses valores com os de outros neurônios ocultos da mesma camada, propagando os valores para a camada de saída e computando as respectivas saídas da rede;
d) gerar regras que descrevam as saídas da rede em função dos valores de ativação discretos de cada neurônio combinado com os valores discretos dos outros

neurônios ocultos;

e) para cada neurônio oculto, enumerar os valores de entrada que conduzem aos respectivos valores de ativação discretos, gerando regras que descrevem os valores de ativação discretos em função dos valores de entrada da rede;

f) combinar as regras obtidas nos dois passos anteriores, de forma que as regras descrevam as saídas da rede em termos dos valores de entrada da mesma.

O algoritmo de agrupamento descrito no segundo passo tem o intuito de agrupar os valores de ativação de cada neurônio em um número discreto de valores de ativação, que são mais fáceis de gerenciar. Um pequeno número de valores de ativações torna possível determinar a dependência entre estes valores discretos e os valores de saída da rede e também a relação desses valores discretos com as entradas da rede (LIU, LU E SETIONO, 1995, p. 483). Um exemplo do algoritmo de agrupamento utilizado pelos autores do algoritmo RX está disposto no quadro 7.

Agrupamento dos valores de ativação(*clustering*):

1) para cada neurônio oculto

   a) seja $\varepsilon \in (0,1)$. Seja *D* o número de valores de ativação discretos no neurônio oculto. Seja δ_1 o valor de ativação no neurônio oculto para o primeiro padrão no conjunto de treinamento. Seja *H*(1) = δ_1, contador(1) = 1, soma(1) = δ_1 e atualize *D* = 1

   b) para todos os padrões *i* = 2, 3, ...*k* no conjunto de treinamento:

      seja δ o valor de ativação

      se existe um $\bar{j}$ tal que $\left|\delta - H(\bar{j})\right| = \min_{j \in \{1,2,...D)} \left|\delta - H(j)\right|$ e $\left|\delta - H(\bar{j})\right| \leq \varepsilon$ então

         contador ($\bar{j}$) = contador ($\bar{j}$) + 1

         soma(*D*) = soma(*D*) + δ

      senão

         D = *D* + *1*

         H(*D*) = δ

         contador(*D*) = 1

         soma(*D*) = δ

   c) substituir em cada agrupamento de *H* o valor de ativação pela média de todos os valores de ativação que foram agrupados no determinado agrupamento:

      para *j* = 1 até *D*

         H(*j*) = soma(*j*)/contador(*j*)

2) verificar a taxa de acerto da rede com os valores de ativação δ^i nos neurônios ocultos substituindo tais valores por δ_d, ou seja, pelos valores de ativação discretos. Se a taxa de acerto cair abaixo do nível requerido, diminuir ε e repetir o passo 1.

Fonte: adaptado de Lu, Liu e Setiono (1995, p. 484).

Quadro 7 – Exemplo de algoritmo de agrupamento utilizado no algoritmo RX

Conforme Liu, Lu e Setiono (1995, p. 484), ao ser aplicado o algoritmo de agrupamento, é necessário verificar se os valores de ativação discretos não afetam demasiadamente a precisão da rede neural artificial. Para isso, as respostas da rede obtidas com os valores de ativação originais são comparadas com as respostas obtidas com os valores discretos respectivos. Além disso, o número de valores discretos obtidos para cada neurônio não pode ser muito grande, sendo então usado um valor que controla o intervalo de valores originais que são agrupados em um valor discreto. Tal número se inicia com um valor que gere o menor número de agrupamentos possível. Então, se a precisão da rede estiver abaixo do tolerado após o agrupamento ser efetuado, o valor pode deve ser ajustado para aumentar o número de agrupamentos e o processo é repetido.

Dessa forma pode ser encontrado um número de valores de ativação discretos para cada neurônio que seja grande o suficiente para não afetar a precisão da rede, mas pequeno o bastante para não gerar muitas combinações entre esses valores de ativação, o que poderia gerar um número muito grande regras.

Os próximos passos do algoritmo, conforme Alves (2001, p. 19), são os que geram as regras propriamente ditas. Inicialmente são gerados dois conjuntos de regras intermediários, sendo que o primeiro conjunto de regras é gerado tendo como conseqüentes os valores de saída da rede e como antecedentes os valores discretos das ativações que geraram tais saídas. O segundo conjunto é gerado tendo como antecedentes os valores de entrada da rede neural e como conseqüentes os valores de ativação discretos de cada neurônio da camada oculta, respectivos aos valores de entrada que fazem parte do intervalo de agrupamento daquele valor discreto. Ao combinar posteriormente o conjunto de regras acima, as regras são obtidas no formato indicado no quadro 8.

Formato das regras geradas pelo algoritmo RX:

"se $(a_1 \theta v_1)$ e $(x_2 \theta v_2)$ e ... e $(x_n \theta v_n)$ então C_j"

onde

a_i são os atributos de um padrão de entrada

v_i são constantes

θ são operadores relacionais ($=, \geq, \leq, <>$)

C_j é um dos rótulos de classe

Fonte: adaptado de Alves (2001, p. 20).

Quadro 8 – Formato das regras geradas pelo algoritmo RX

Em relação às características de qualidade e acurácia das regras extraídas, Andrews et al. (1998, p. 1060) comenta que as mesmas são equivalentes ou até melhores em comparação com outros algoritmos de indução de regras, como o C4.5 (que gera árvores de decisão diretamente a partir dos dados) ou o MOFN, que também é um algoritmo de extração de regras de redes neurais artificiais. A complexidade do algoritmo não é comentada, e sobre a generalidade, o algoritmo é aplicado geralmente sobre redes MLP, com algoritmos de aprendizagem supervisionados.

2.4.5.2 O algoritmo de extração de regras TREPAN

O algoritmo TREPAN foi desenvolvido por Mark Craven em 1996 como parte de sua tese de doutorado. O principal propósito do algoritmo TREPAN é suprir os problemas de escalabilidade e generalidade das outras abordagens em extração de regras. O algoritmo construído não necessita conhecer a arquitetura da rede neural e nem a forma como a rede foi treinada. Dessa forma, o algoritmo pode ser aplicado a uma grande variedade de modelos de aprendizagem, não somente a redes neurais, e faz parte da abordagem pedagógica, dentro da classificação dos tipos de técnicas de extração de regras (CRAVEN, 1996, p. 1).

Conforme Craven (1996, p. 42), o algoritmo obtêm uma representação simbólica aproximada em formato de uma árvore de decisão dos conceitos aprendidos por uma rede neural treinada. O algoritmo se utiliza de um oráculo, ou seja, um sistema ao qual é possível dirigir questionamentos e que seja capaz de responder a esses questionamentos. No caso, a rede neural artificial treinada é utilizada como o oráculo e a função alvo do problema deve gerar questionamentos sobre o conhecimento que a rede representa, e usando, se possível, os dados de treinamento da rede. O quadro 9 fornece uma descrição em pseudocódigo dos passos do algoritmo.

Pseudocódigo do algoritmo TREPAN

Entradas: Oráculo(), conjunto de treinamento *S*, conjunto de características *F*, parâmetro *mínimo_exemplos*, critério de parada

1. **para cada** exemplo *x* em *S*
2. rótulo de classe para *x* = Oráculo(*x*)
3. inicializar a raiz da árvore, *R*, como um nodo folha
4. construir modelo *M* de distribuição de instâncias cobertas pelo nodo *R*
5. $instancias\_consulta_R$ = DESENHAR_EXEMPLOS({}, *mínimo_exemplos*-|*S*|, *M*)
6. usar *S* e $instancias\_consulta_R$ para determinar o rótulo de classe de *R*
7. inicializar *Fila* com o conjunto <*R*, *S*, $instancias\_consulta_R$,{}>
8. **enquanto** *Fila* não está vazia e critério de parada global não satisfeito
9. remover <nodo *N*, S_N, $instancias\_consulta_N$, $restrições_N$> do topo de *Fila*
10. *T* = CONSTUIR_TESTE(*F*, $S_N \cup instancias\_consulta_N$)
11. faça *N* um nodo interno com teste *T*
12. **para cada** resultado, *t*, do teste *T*
13. faça *C*, um novo filho de *N*
14. $restrições_C = restrições_N \cup \{T = t\}$
15. S_C = membros de S_N com o resultado *t* no teste *T*
16. construir modelo *M* de distribuição de instâncias cobertas pelo nodo C
17. $instancias\_consulta_C$ = DESENHAR_EXEMPLOS($restrições_C$, *mínimo_exemplos*-|S_C|, *M*)
18. usar S_C e $instancias\_consulta_C$ para determinar o rótulo de classe para C
19. **se** critério de parada local não satisfeito então
20. colocar (*C*, S_C, $instancias\_consulta_C$, $restrições_C$) em *Fila*

Retorno: árvore com raiz *R*

Fonte: adaptado de Craven (1996, p. 43).

Quadro 9 – Pseudocódigo do algoritmo de extração TREPAN

Conforme Núñes, Rincón e Rozo (2004, p. 822), existem diversos algoritmos de indução de árvores de decisão a partir de um conjunto de exemplos, como o ID3 e C4.5. Basicamente o TREPAN é semelhante ao algoritmo ID3. Porém, existem algumas diferenças, onde as principais são descritas nos parágrafos que seguem.

O oráculo, no caso de que é necessário representar o conhecimento embutido na rede neural artificial treinada, será a própria rede. A rede receberá do algoritmo um conjunto de dados para classificação e irá responder ao algoritmo qual é a classe dessa instância. Não há necessidade de se explorar internamente a rede, pois ela apenas irá classificar os dados recebidos.

Outra característica do algoritmo TREPAN é a geração de novas instâncias, ou seja, novos dados para classificação. Essas novas instâncias são enviadas ao oráculo para classificação. Desta forma, o algoritmo exige que exista uma quantidade mínima de dados em um determinado nodo da árvore antes de se obter a identificação da classe ou para selecionar um teste de partição, típico em algoritmos de árvores de decisão.

A técnica de expansão da árvore de decisão apresenta uma característica diferenciada da maioria dos algoritmos de indução de árvores de decisão. O critério usado pelo algoritmo TREPAN é de escolher primeiramente o melhor nodo para expansão, que no caso é o que possui maior probabilidade de aumentar a fidelidade da árvore que é extraída da rede. Isto é feito com base numa equação que considera a fração estimada de exemplos de treinamento que atingem o nodo e a fração de instâncias aonde a rede e a árvore chegaram à mesma conclusão, ou seja, classificaram corretamente os exemplos.

A seleção dos testes de partição é a outra característica diferenciada do algoritmo TREPAN face às demais abordagens de algoritmos de indução de árvores de decisão. O TREPAN utiliza apenas testes binários para realizar a partição da árvore. Tais testes podem ser disjuntivos ou expressões do tipo `se M dos seguintes N antecedentes são verdadeiros então conseqüente`. Para construir esse teste são considerados todos os possíveis valores assumidos pelas variáveis, oriundos tanto dos dados de treinamento como dos novos dados gerados pela geração de novas instâncias do algoritmo e classificadas pelo oráculo. A seleção dos testes e dos critérios de divisão é feita por uma avaliação heurística chamada de critério da taxa de ganho (*gain criterion rate)*, que avalia as divisões candidatas.

O critério de parada é utilizado para definir quando deve ser terminada a expansão da árvore. O algoritmo TREPAN utiliza dois critérios, sendo um local e outro global. O critério local avalia a pureza dos dados que chegam a um nodo, ou seja, se os dados dos exemplos pertencem sempre a uma mesma classe. Já o critério global se refere ao tamanho máximo da árvore gerada, que pode ser determinado pelo usuário em termos do número de nodos internos ou o próprio algoritmo pode calcular o tamanho através do conjunto de dados de teste.

Conforme Andrews et al. (1998, p. 1061), o algoritmo gera regras em formato de árvores de decisão com testes do tipo se `M dos seguintes N antecedentes são verdadeiros` em cada nodo. As árvores de decisões geradas são consideradas de alta acurácia, fidelidade e compreensibilidade. A complexidade é polinomial em relação à dimensionalidade do espaço de entrada (quantidade de exemplos do conjunto de treinamentos) e do número máximo de valores de uma determinada característica.

3 DESENVOLVIMENTO DO TRABALHO

As próximas seções descrevem a especificação do protótipo onde serão executados os algoritmos de extração de regras, utilizando os conceitos apresentados na fundamentação teórica.

3.1 VISÃO GERAL DO PROTÓTIPO

O protótipo tem o intuito de executar dois algoritmos de extração de regras de redes neurais artificiais, no caso o RX e TREPAN. Para a execução dos algoritmos também é necessário que o protótipo possa construir, treinar e testar uma rede neural artificial para a tarefa de classificação de dados, e que também permita selecionar e tratar os dados a serem classificados, para treinamento e posterior extração de regras da rede neural artificial.

Para facilitar os passos acima descritos, o protótipo foi construído na forma de um ambiente que permita executar e configurar os componentes de cada passo. Cada passo será considerado parte de um experimento, e um experimento pode ser persistido e recuperado com algumas informações textuais. Os passos ou componentes de um experimento estão assim divididos:

a) configurar uma fonte de dados, onde é possível informar a fonte de dados (arquivo texto, base de dados, etc.), configurar as informações acerca dos dados (atributos), além de buscar os dados para serem utilizados na rede neural artificial;
b) criar, configurar e treinar uma rede neural artificial e aplicar os algoritmos de extração de regras sobre a rede já treinada.

Um experimento possui duas fases distintas, sendo a primeira a fase de configuração e a segunda a fase de execução. A fase de configuração permite criar um experimento, informar detalhes textuais acerca do mesmo e adicionar os componentes, indicando a forma de busca de dados pelo componente de fonte de dados e ajustando os atributos desses dados. O componente de rede neural artificial e extração de regras também deve ser incluído e configurado pelo usuário.

A fase de execução analisa os requisitos mínimos para execução e chama os métodos dos operadores para buscar os dados e para executar a rede neural. Na execução do

componente de rede, deve ser escolhido um dos passos que este componente deve efetuar. Esses passos podem ser ou o passo de treinamento, ou de teste, ou de extração de regras ou de validação das regras extraídas. Para se extrair efetivamente as regas da rede neural, cada passo descrito deve ser configurado na seqüência descrita acima e para cada um, o experimento deve ser executado novamente.

O ambiente de configuração e execução de um experimento é independente da implementação dos algoritmos de busca de dados que serão utilizados e dos algoritmos de redes neurais e extração de regras. Ou seja, o ambiente não apresenta componentes próprios implementados internamente. Para permitir a execução e configuração de componentes que possuem esses algoritmos, existe um padrão sobre o qual os algoritmos deverão ser desenvolvidos. Para isso é utilizado o conceito de herança do paradigma da programação orientação a objeto. O ambiente disponibiliza publicamente classes de objetos com *interfaces* pré-definidas, ou seja, com métodos e atributos padrões. O ambiente utiliza esses métodos padrões e os atributos para as fases de configuração e execução dos componentes desenvolvidos com base nos componentes disponibilizados. Tais componentes encapsulam a implementação dos algoritmos aos quais se destinam, como a busca de dados ou manipulação e extração de regras de uma rede neural artificial.

Os componentes são denominados de operadores, No caso, esses operadores devem atender aos objetivos do protótipo, direta ou indiretamente, e seriam os seguintes:

a) operadores capazes de ler os dados de entrada para a rede neural de alguma fonte;
b) operadores capazes de instanciar e executar uma rede neural, procedendo ao seu treinamento, teste e extração de regras.

Os operadores possuem formas de entrada e saída de dados, os quais podem ser transferidos a outro operador, visualizados no próprio protótipo, ou ser apenas o resultado do experimento. Cada classe base de operador possui seu gerenciador específico e sua execução é coordenada com os demais operadores, sempre em seqüência.

Os operadores utilizados como extensões dos operadores de base também possuem uma forma de registro, o que permite identificar o tipo de operador e um nome. Esses operadores, implementados externamente como pacotes de aplicativo, podem então ser adicionados ao sistema e disponibilizados para uso.

Com base nos operadores base de manipulação de redes neurais artificiais e extração de regras, deverão ser feitas extensões de dois novos operadores, encapsulando em uma das extensões o algoritmo RX e em outra extensão o algoritmo TREPAN, cada qual com sua respectiva rede neural MLP.

3.2 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Com o levantamento dos requisitos é possível que usuários e desenvolvedores tenham a mesma visão do problema a ser resolvido. Essa etapa é muito importante para saber a dimensão do problema e aplicar no desenvolvimento do protótipo. O problema tratado neste trabalho será dividido em duas partes, sendo uma para o desenvolvimento do protótipo que permite gerar os experimentos e fornecer as classes base para extensão e a outra será a implementação dos algoritmos de extração de regras e manipulação das redes neurais, a partir das classes base de operadores.

A seguir são relacionados, no quadro 10, os requisitos funcionais e no quadro 11, os requisitos não funcionais do protótipo:

REQUISITOS FUNCIONAIS	CASO DE USO
RF01: Possuir e permitir configurar e manipular um operador como extensão da classe base de redes neurais artificiais para permitir a configuração, construção e treinamento de uma rede neural artificial MLP e a extração de regras da rede neural artificial através do algoritmo RX	UC02 UC03 UC04
RF02: Possuir e permitir configurar e manipular um operador como extensão da classe base de operador de entrada e manipulação de dados, para a devida manipulação e busca dos dados de um banco de dados e provendo os dados para o treinamento e extração de regras de uma rede neural artificial	UC02 UC03 UC04
RF03: Permitir registrar os operadores que sofreram extensão a partir das classes base de operadores para uso no protótipo.	UC01
RF04: Permitir criar, persistir, recuperar e manipular um experimento, montando o mesmo com os operadores que estão registrados no protótipo.	UC02
RF05: Configurar os operadores utilizados no experimento, avaliando possíveis inconsistências ou falta de utilização de parâmetros essenciais na execução.	UC03 UC02 UC04
RF06: Executar um experimento, executando os operadores para que seja efetuada a busca de dados, o treinamento ou extração de regras de redes neurais artificiais.	UC04

Quadro 10 – Requisitos funcionais

REQUISITOS NÃO FUNCIONAIS
RNF01: O protótipo deverá ser implementado no ambiente Delphi 7.
RNF02: Os operadores a sofrerem extensão também serão implementados em Delphi 7 e gerados como bibliotecas de ligação dinâmica (*dinamic link library* – DLL).
RNF03: Utilizar o padrão XML para a persistência do experimento. O experimento possui seu próprio XML e cada operador a sofrer extensão deverá fornecer o seu XML com seus dados de configuração.
RNF04: Utilizar o banco de dados Microsoft SQL Server Express para persistência dos dados, configurações e resultados do experimento e também das informações sobre os registros dos operadores, pelo protótipo.

Quadro 11 – Requisitos não funcionais

3.3 ESPECIFICAÇÃO

O protótipo foi especificado através da ferramenta Entreprise Architect (SPARX SYSTEMS, 2007), utilizando os conceitos de orientação a objetos e baseando-se nos diagramas em UML, gerando como produtos os diagramas de caso de uso, de atividades, de classes e de dados.

3.3.1 Diagrama de casos de uso

O diagrama de casos de uso indica os passos necessários para o funcionamento da ferramenta. A figura 5 representa os quatro casos de usos levantados. Cada caso é descrito na seqüência.

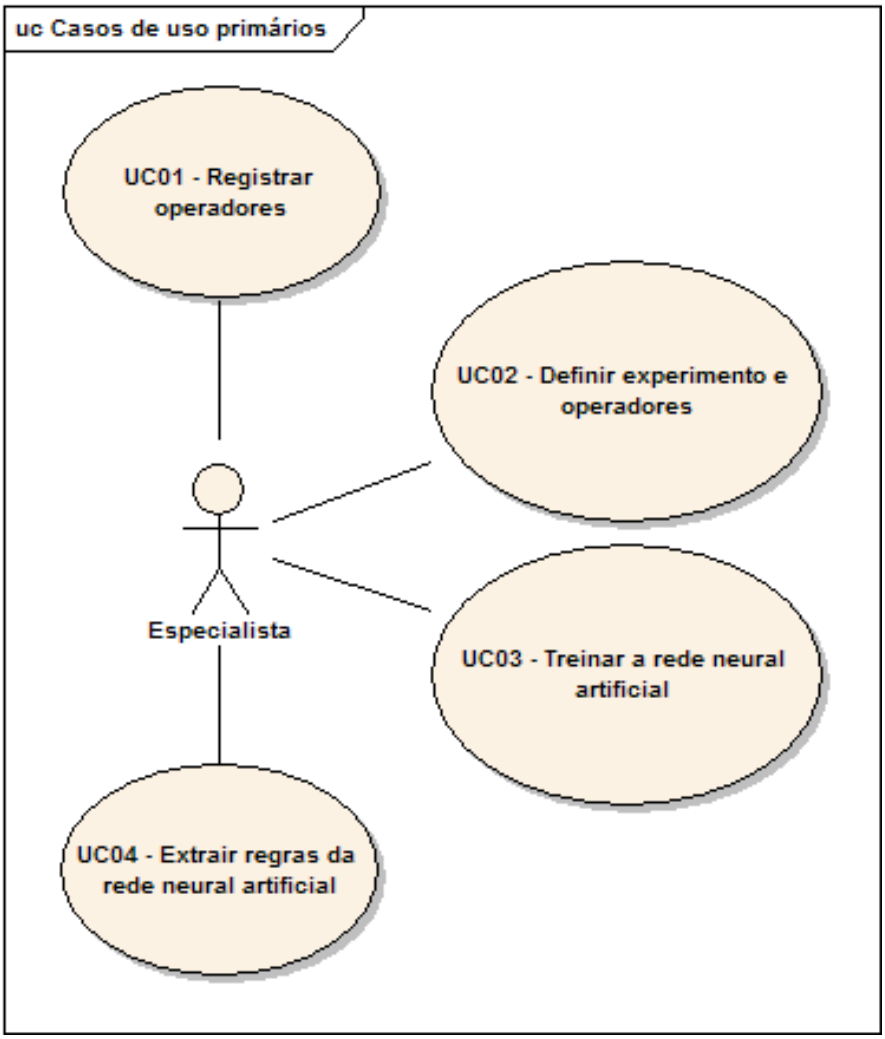


Figura 5 – Diagramas de casos de uso

O primeiro caso de uso é denominado de `Registrar operadores` (quadro 12), e descreve o procedimento para escolher e registrar no protótipo um operador desenvolvido externamente a partir de uma das classes base disponibilizadas. Esse operador deve ser uma classe disponibilizada em um arquivo no formato de biblioteca de ligação dinâmica. Caso seja validado pelo protótipo como descendente das suas classes base, será disponibilizado para uso em experimentos.

UC01 - Registrar operadores	
Requisitos atendidos	RF03
Pré-condições	Não possui.
Cenário principal	1) O especialista seleciona um operador em forma de arquivo DLL (*Dynamic* Link Library) existente externamente ao protótipo 2) O sistema carrega o operador e avalia o tipo do mesmo,registrando no sistema conforme esse tipo
Exceção 01	Operador inválido: Se no passo 2 o sistema detectar que o operador não é válido, no caso não foi feita a extensão desse operador a partir de um operador base, é gerada uma exceção
Exceção 02	Operador já cadastrado: Se no passo 2 o sistema detectar que o operador já está registrado, é gerada uma exceção
Pós-condições	Operadores registrados

Quadro 12 – Caso de uso `UC01`

O segundo caso de uso (quadro 13) é designado `Definir experimento e operadores`, e é um dos casos de uso mais importantes do protótipo, visto que é nele que o especialista define o experimento completo, em sua fase de configuração. Ele possui um cenário principal, onde é feita a definição do experimento e seus operadores e a configurações possíveis. Mais dois fluxos alternativos demonstram a possibilidade de recuperação de um experimento e a remoção e adição de operadores. Existem também dois cenários de exceção, onde os possíveis erros encontrados durante o processo são levantados. O diagrama de atividades apresentado na figura 6 demonstra os passos da execução desse caso de uso.

UC02 - Definir experimento e operadores	
Requisitos atendidos	RF01, RF02, RF04, RF05
Pré-condições	Os operadores devem estar registrados
Cenário principal	1) O especialista informa um experimento, definindo uma identificação textual para o mesmo. A qualquer momento após isso o especialista pode salvar o experimento 2) O especialista seleciona e adiciona um operador de entrada de dados e um operador de redes neurais artificiais e extração de regras ao experimento 3) O especialista informa ao operador de entrada de dados os parâmetros exigidos por este para obter os dados a serem lidos para o uso no experimento 5) O especialista solicita a busca dos atributos dos dados da fonte de dados pelo operador de busca de dados 4) O protótipo chama o método de validação do operador de entrada de dados para verificar se os parâmetros estão corretos e a conexão é válida e funcional 5) O protótipo chama o método de execução do operador de entrada de dados, para extrair os atributos da fonte de dados e exibir as informações no editor de atributos 6) O especialista analisa, classifica ou omite atributos no editor de atributos, sendo essas informações mantidas pelo operador 7) O especialista especifica a estrutura da rede neural artificial, indicando o número de camadas e os números de neurônios por camada 8) O especialista mapeia os valores possíveis dos atributos disponibilizados pelo operador de entrada de dados para os neurônios de entrada da rede neural artificial 9) O especialista mapeia os valores possíveis do atributo que possui o rótulo de classe dos dados de entrada para os neurônios de saída da rede neural artificial 10) O especialista salva o experimento
Fluxo alternativo 01	1) O especialista seleciona um experimento existente da base de dados 2) O protótipo recupera o experimento da base de dados, juntamente com os operadores e suas configurações e monta os operadores para permitir edição 2) O especialista após isso pode adicionar ou remover operadores (fluxo alternativo 02) ou configurar e editar os operadores existentes, partindo do passo 3 ou 6 do cenário principal
Fluxo alternativo 02	Em qualquer passo ou momento após o passo 2 do cenário principal, é possível remover e adicionar novos operadores ao experimento. Se o operador adicionado for um operador de entrada e saída de dados, é possível retornar passo 3 do cenário principal. Se for um operador de rede neural artificial e extração de regras, pode ser retornado ao passo 6 do cenário principal
Exceção 01	Operador de busca de dados inválido: No passo 4, é chamada a validação do operador pelo gerenciador do operador de busca de dados. Nesse caso, o operador deve validar a conexão e gerar uma exceção a ser devolvida e emitida pelo protótipo, voltando ao passo 3
Pós-condições	Experimento definido

Quadro 13 – Caso de uso UC02

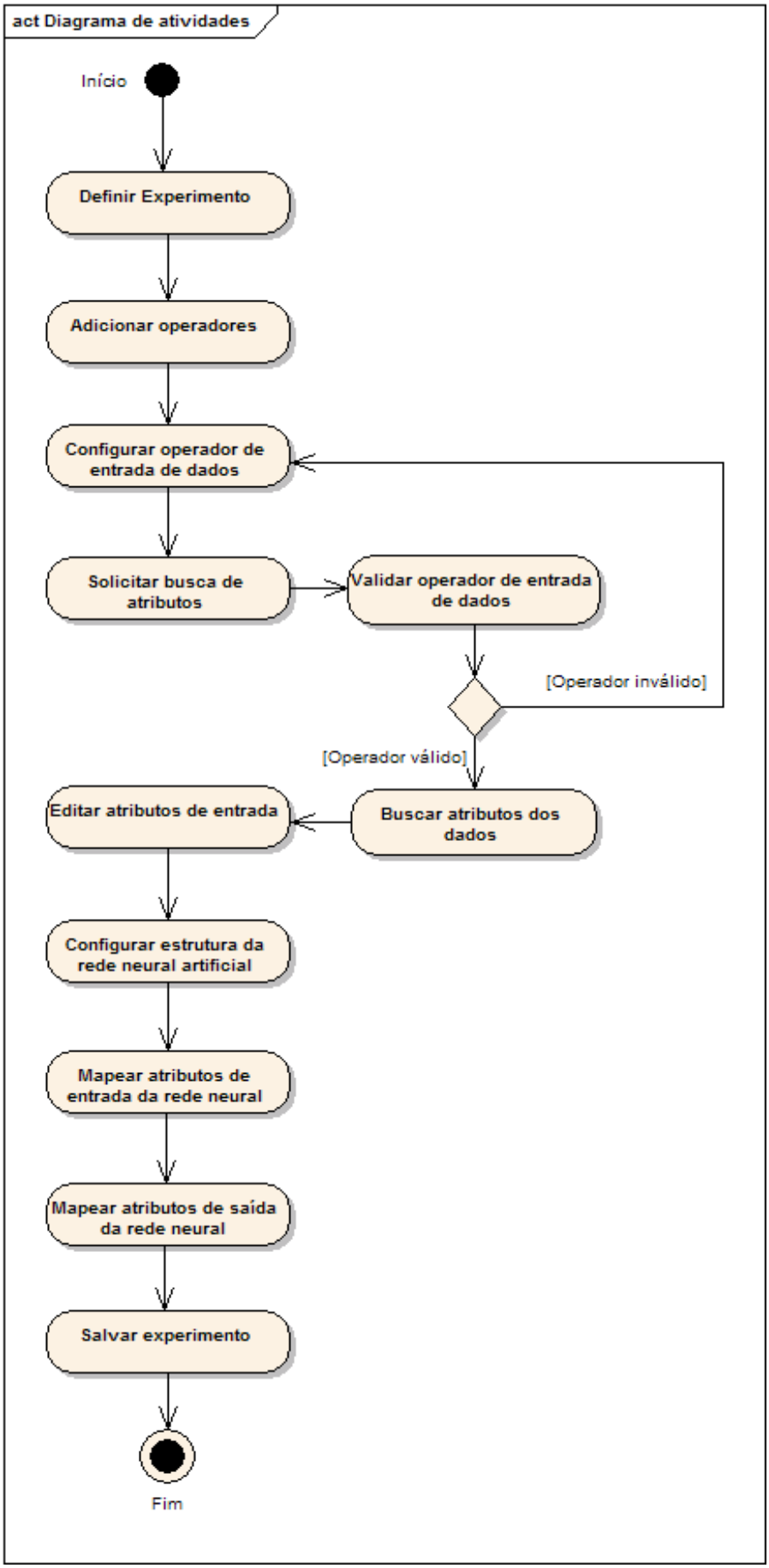


Figura 6 – Diagramas de atividades do caso de uso UC02

O terceiro caso de uso (quadro 14) é designado `Treinar a rede neural artificial`. Neste caso de uso são descritos os passos onde o especialista define os parâmetros exigidos para o treinamento da rede neural artificial, executa o experimento para proceder ao treinamento da rede e efetua os testes para validar a taxa de acerto da rede após o treinamento. Esse processo é aplicado ao operador de redes neurais e extração de regras, e pode ser repetido diversas vezes, já que podem ocorrer ajustes nos parâmetros da rede neural artificial. Isso dependerá do fato da taxa de acerto que da rede ser aceita ou não pelo especialista. O produto desse caso de uso é a rede neural artificial treinada.

UC03 - Treinar a rede neural artificial	
Requisitos atendidos	RF01, RF02, RF05 e RF06
Pré-condições	O operador de entrada de dados deve estar configurado; Rede neural com configurações e estrutura corretamente parametrizadas.
Cenário principal	1) O especialista define parâmetros de treinamento para a rede neural artificial, conforme exigido pelo operador da rede neural artificial 2) O especialista informa ao operador da rede que este deve entrar em modo de treinamento e executar o experimento 3) O protótipo chama o método para validar a estrutura e configuração do operador de busca de dados 4) O protótipo chama o método para validar a estrutura e configuração da rede neural artificial 5) O protótipo chama o método de execução do operador de busca de dados, repassando os dados obtidos por esse operador e ao operador da rede neural artificial 6) O protótipo chama o método de execução do operador de rede neural artificial 7) O operador da rede neural artificial deverá executar o treinamento da rede, conforme os parâmetros pré-definidos 8) O especialista modifica o modo de operação da rede para executar os testes sobre um conjunto de dados diferentes do conjunto de dados do treinamento 9) O especialista modifica o operador de entrada de dados para obter um conjunto de dados diferente do conjunto de dados de treinamento da rede 10) O especialista executa o experimento novamente 11) O protótipo executa os passos 3 a 6 novamente, efetuando a passagem dos dados pela rede e registrando o resultado de cada exemplo de treinamento 12) Ao final da passagem do conjunto de dados de teste, o operador apresenta a taxa de acerto da rede em relação ao conjunto de dados de teste 13) O especialista analisa a taxa de acerto e pode decidir por voltar ao passo 1, redefinindo os parâmetros de treinamento caso a taxa não seja a desejada ou passar ao passo 14 14) Após o treinamento, o operador da rede neural artificial deverá manter a estrutura da rede, ou seja o conjunto de pesos sinápticos da rede
Exceção 01	Configuração da rede neural incorreta: O protótipo chama o método de validação do operador da rede neural artificial, que deve verificar se as estruturas e configurações exigidas para o operador estão corretas. Se não estiverem corretas, o problema deve ser devolvido ao gerenciador do operador e será gerada uma exceção
Pós-condições	Rede neural artificial treinada

Quadro 14 – Caso de uso `UC03`

O quarto caso de uso, apresentado no quadro 15 com a denominação `Extrair regras da rede neural artificial`, é também o caso de uso mais relevante do protótipo. Neste caso de uso, são descritos os passos do especialista para proceder a extração das regras da rede neural artificial já treinada. Esse processo envolve também todo o experimento, pois são necessários os dados de entrada e saída do conjunto de treinamento da rede, bem com a execução da rede para a extração de regras. Um cenário de exceção existe para informar quando ocorrerem problemas de configuração dos parâmetros de configuração do algoritmo de extração de regras implementado no operador. O produto desse caso de uso são as regras extraídas da neural artificial sobre do conjunto de treinamento da rede.

UC04 - Extrair regras da rede neural artificial	
Requisitos atendidos	RF01, RF02 e RF05, RF06
Pré-condições	O operador de entrada de dados deve estar configurado; Rede neural artificial treinada.
Cenário principal	1) O especialista informa os parâmetros exigidos pelo operador que possui a implementação da rede neural artificial e do algoritmo de extração de regras, e indica para o operador que deve entra no modo de extração das regras 3) O especialista configura o operador de entrada de dados para obter novamente os dados de treinamento da rede neural 2) O especialista executa o experimento. 3) O protótipo valida a estrutura e configuração do operador de busca de dados 4) O protótipo valida a estrutura e configuração da rede neural artificial para execução 5) O protótipo chama o método de execução do operador de busca de dados, repassando os dados obtidos por esse operador ao operador da rede neural artificial 6) O protótipo chama o método de execução do operador de rede neural artificial 7) O operador da rede neural artificial deverá executar o algoritmo de extração de regras da rede, conforme os parâmetros pré-definidos 8) Após a extração de regras da rede neural artificial, o operador deve exibir e manter as regras extraídas, no formato pré-definido pelo algoritmo.
Exceção 01	Configuração dos parâmetros do algoritmo de extração de regras inválidos: Se no passo 1 o especialista não informar os parâmetros exigidos pelo algoritmo extrator de regras de redes neurais artificiais, o operador deve devolver o problema ao gerenciador do operador e será então levantada uma exceção
Pós-condições	Regras das redes neurais artificiais extraídas

Quadro 15 – Caso de uso `UC04`

3.3.2 Diagrama de classes

O diagrama de classes apresenta uma visão de como as classes estão estruturadas e relacionadas. Devido ao número de classes necessárias para a implementação do protótipo,

elas estão reunidas em pacotes e representadas com suas ligações lógicas pela figura 7. Serão descritas as funcionalidades de cada pacote e também as descrições das funcionalidades das classes mais importantes de cada pacote.

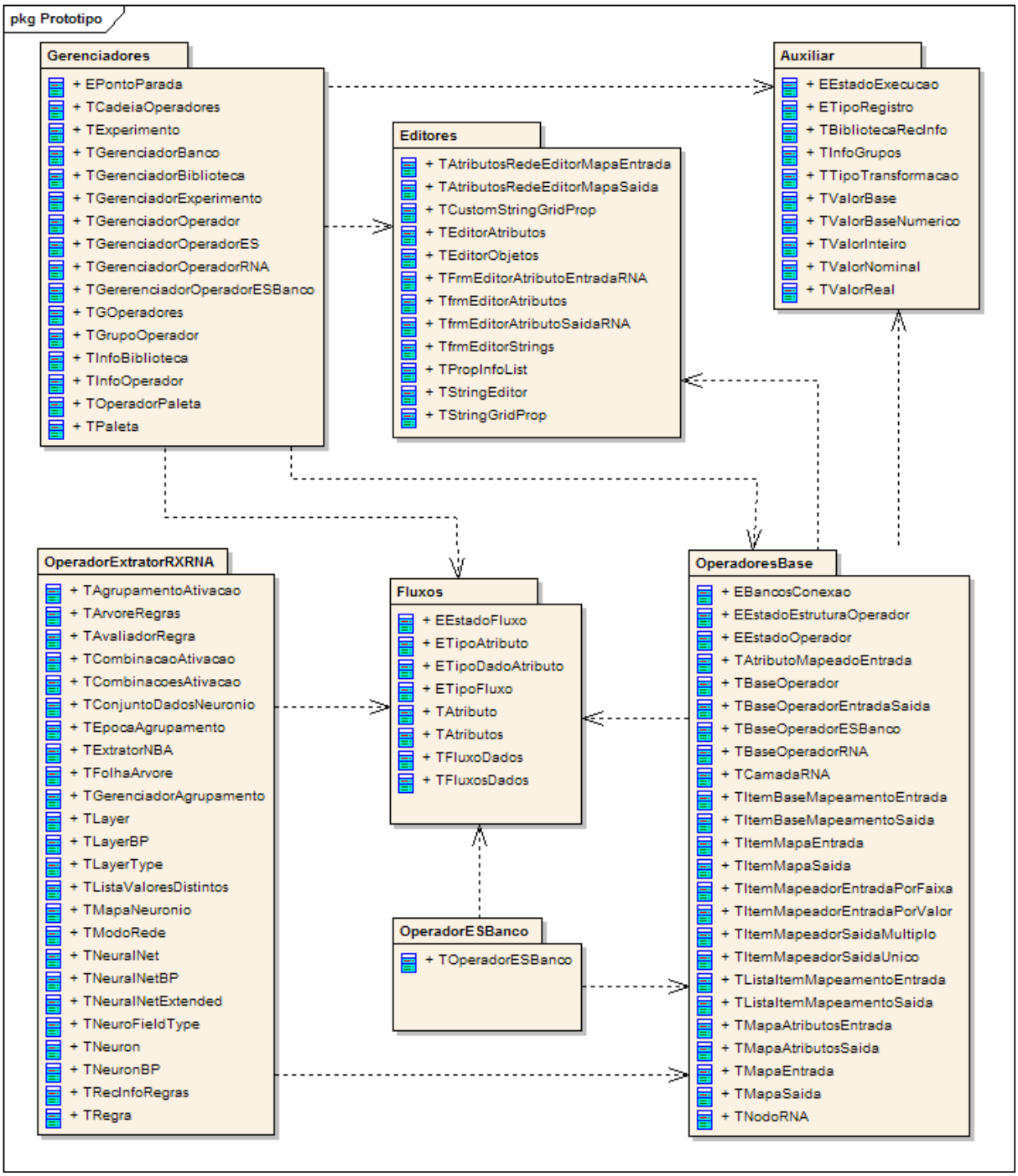


Figura 7 – Pacotes do protótipo

3.3.2.1 Pacote `Auxiliar`

O pacote `Auxiliar` possui artefatos que são comuns aos demais pacotes do protótipo. A figura 8 exibe os artefatos desse pacote.

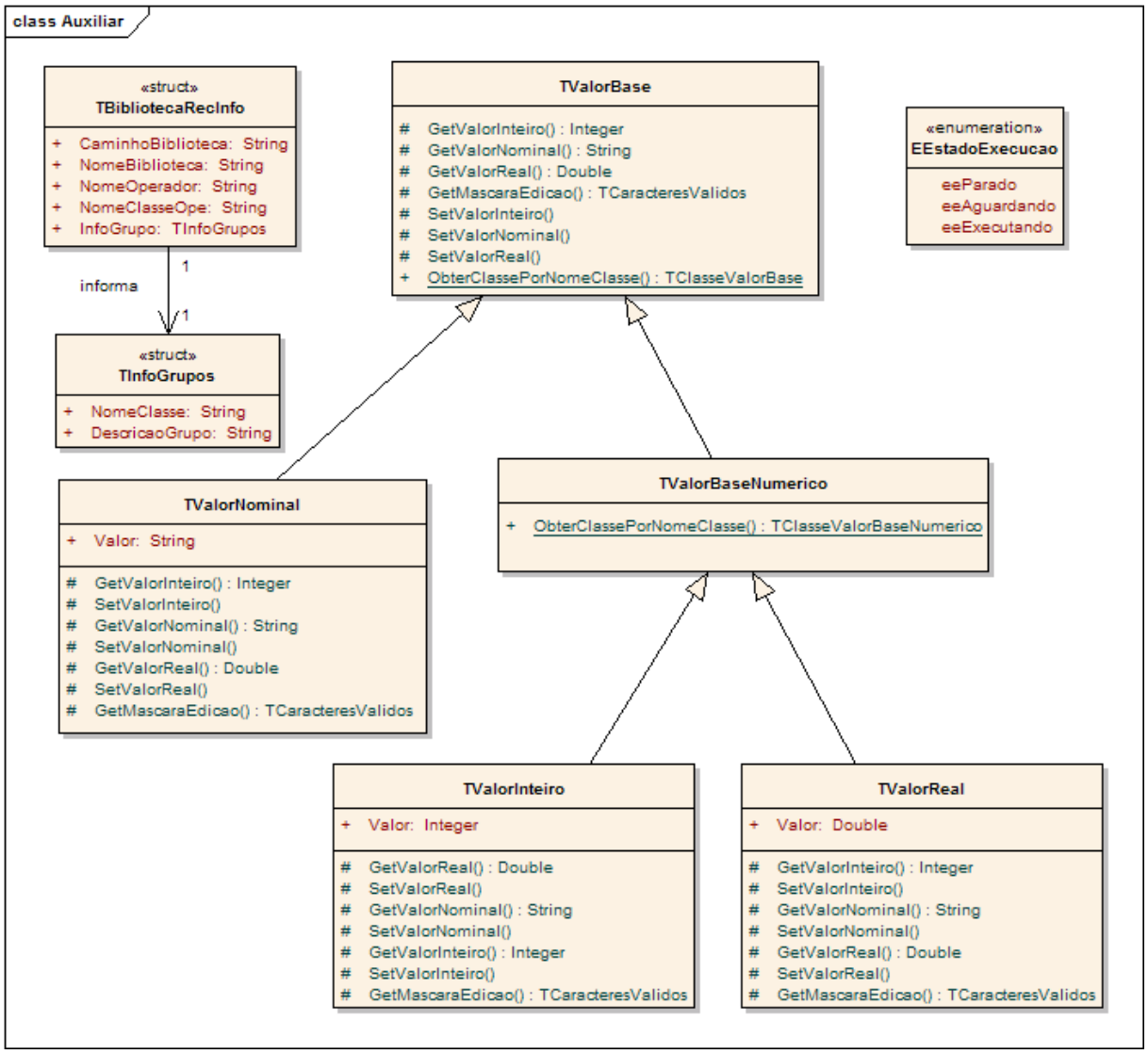


Figura 8 – Diagrama de classes e outros artefatos do pacote `Auxiliar`

A funcionalidade de cada classe, enumeração e estrutura do pacote `Auxiliar` são a seguir descritas:

a) `TBibliotecaRecInfo` é um registro com informações a respeito das bibliotecas de ligação dinâmica registradas no protótipo que possuem os operadores estendidos;

b) `TInfoGrupos` é um registro com o nome de uma classe de operador registrado no protótipo e o nome do grupo de operadores ao qual essa classe pertence;

c) `EEstadoExecução` é uma enumeração que indica o estado de execução de um operador;

d) `TValorBase` é uma classe abstrata utilizada para indicar um tipo determinado de valor de atributo;
e) `TValorNominal` é uma classe descendente de `TValorBase` que representa um valor alfanumérico de um atributo e as conversões para outros tipos de valores;
f) `TValorBaseNumerico` é uma classe abstrata descendente de `TValorBase` utilizada para representar um valor numérico;
g) `TValorInteiro` é uma classe especializada de `TValorBaseNumerico` que representa um valor numérico inteiro e as conversões para outros tipos de valores;
h) `TValorReal` é uma classe especializada de `TValorBaseNumerico` que representa um valor numérico de ponto flutuante e as conversões para outros tipos de valores.

3.3.2.2 Pacote `Fluxos`

O pacote `Fluxos`, cujo diagrama de classes é apresentado na figura 9, disponibiliza classes para manipulação de atributos e seus valores e de conjuntos de dados. É utilizada pelo protótipo e pelos operadores base, e a breve descrição de seus artefatos segue abaixo:

a) `EtipoAtributo` é uma enumeração que indica se um determinado atributo contém a informação de um simples dado, de um rótulo de classe ou da predição de saída de uma rede neural;
b) `EtipoDadoAtributo` é uma enumeração que indica o tipo de valor mantido por uma determinado atributo;
c) `TAtributo` é uma classe que define um atributo real de dado e suas informações;
d) `TAtributos` é uma classe que mantém uma lista de atributos (`TAtributo`);
e) `TFluxoDados` é uma classe de manipulação de um conjunto de dados e seus atributos (`TAtributos`);
f) `TFluxosDados` é uma classe que mantêm e gerencia uma lista de fluxos de dados (`TFluxoDados`);
g) `EEstadoFluxo` é uma enumeração que indica o estado de um fluxo de dados;
h) `ETipoFluxo` é uma enumeração que indica se um fluxo de dados é de entrada, saída ou temporário.

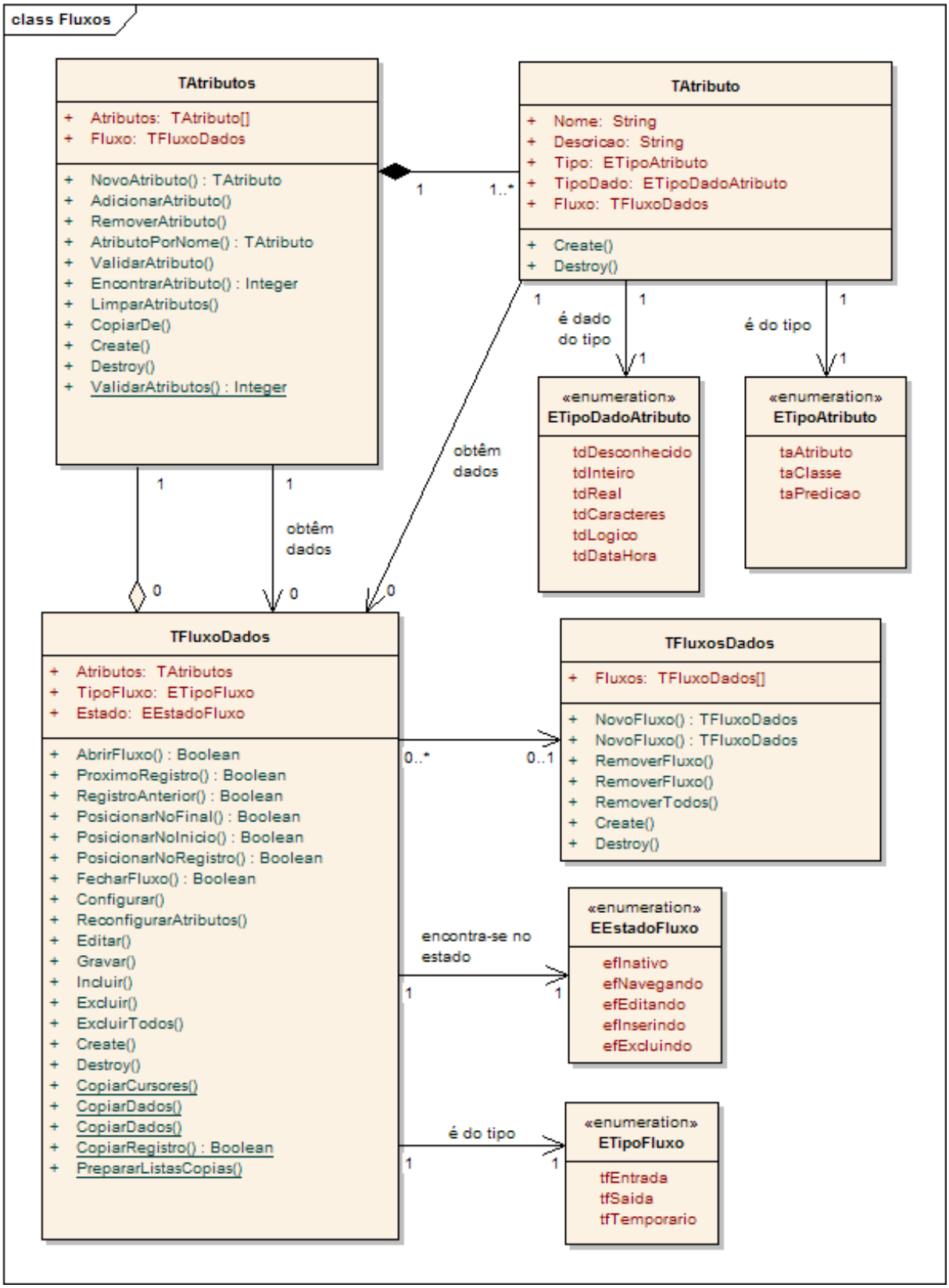


Figura 9 – Diagrama de classes e outros artefatos do pacote `Fluxos`

3.3.2.3 Pacote `OperadoresBase`

O pacote `OperadoresBase` contém as classes base para os operadores de entrada e saída e manipulação de dados e atributos, bem como as classes base para operadores de redes neurais artificiais e extração de regras. Também existem classes para mapeamento de atributos de dados para os neurônios de entrada e saída de uma rede, além de outras estruturas e enumerações necessárias para os operadores. A partir destas classes são construídas as extensões para os operadores que encapsulam os algoritmos de busca e manipulação de dados e de redes neurais artificiais e extração de regras. Os diagramas de classes estão divididos em três, sendo que o primeiro (figura 10) apresenta as classes base de operadores, o segundo (figura 11) apresenta as classes para mapeamento de valores de atributos em valores de entrada da rede e o terceiro (figura 12) apresenta as classes para mapeamento dos valores de saída da rede em valores de atributos. As classes para o diagrama da figura 10 são descritas a seguir:

a) `EEstadoOperador` é uma enumeração indicando o estado de execução ou de configuração de um operador em uso em um experimento;
b) `EEstadoEstruturaOperador` é uma enumeração indicando se o estado da estrutura ou da configuração de um operador em uso em um experimento é válido ou inválido para execução;
c) `TBaseOperador` é a classe base de todos os operadores;
d) `TBaseOperadorEntradaSaída` é a classe base dos operadores para entrada, saída e manipulação de dados, descendente de `TBaseOperador`;
e) `TBaseOperadorESBanco` é uma classe especializada de `TBaseOperadorEntradaSaída`, utilizada para operações de entrada e saída de dados envolvendo conexão com banco de dados;
f) `TBaseOperadorRNA` é a classe base dos operadores para manipulação de redes neurais artificiais e algoritmos de extração de regras;
g) `EBancosConexao` é uma enumeração que indica os tipos de gerenciadores de bancos possíveis para as conexões externas com bancos de dados.

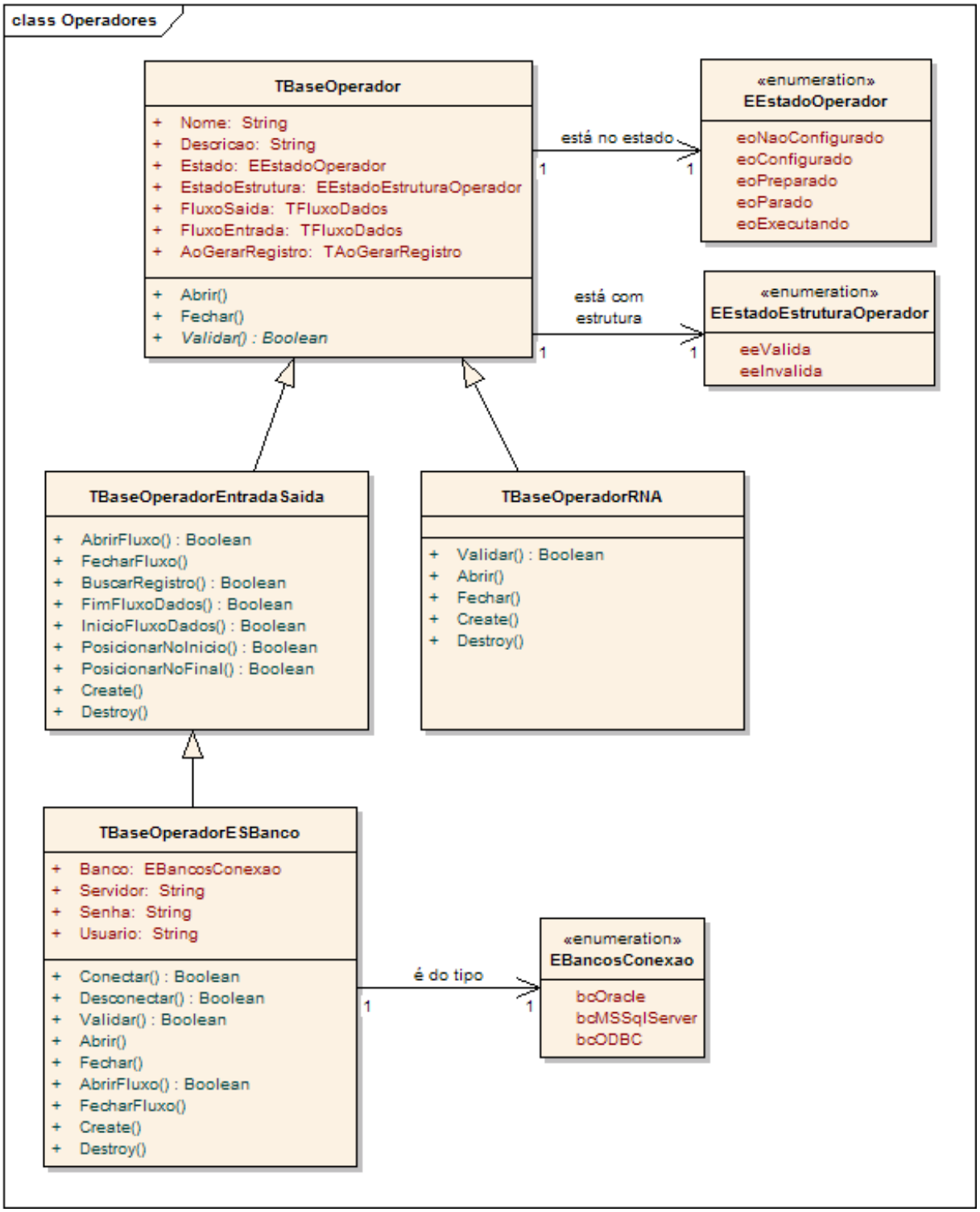


Figura 10 – Diagrama das classes de operadores base do pacote `OperadoresBase`

Os artefatos do diagrama da figura 11 tratam da interoperabilidade entre os atributos de dados de um fluxo de dados e os neurônios da camada de entrada da rede neural artificial. Estes artefatos permitem indicar que atributos de dados são ligados a que neurônios de entrada da rede, além de configurar e efetuar transformações ou conversões dos valores dos atributos para os neurônios de entrada da rede. As descrições simplificadas de cada classe estão a seguir:

a) `TNodoRNA é` uma classe que representa um neurônio de saída ou de entrada de uma rede neural artificial e seu valor numérico;

b) `TCamadaRNA` é a classe que representa uma camada de saída ou de entrada de uma rede neural artificial, com sua lista de neurônios;

c) `TItemBaseMapeamentoEntrada` é a classe cujo objetivo é converter diretamente um valor de entrada qualquer em um valor numérico para entrada de um neurônio específico da rede neural;

d) `TItemMapeadorEntradaPorValor` é uma especialização de `TItemBaseMapeamentoEntrada` que converte um valor qualquer para um valor numérico já pré-definido para o neurônio de entrada da rede neural;

e) `TItemMapeadorEntradaPorFaixa` é uma especialização de `TItemBaseMapeamentoEntrada` que verifica se um determinado valor pertence a uma faixa de valores pré-definidos e transforma em um único valor numérico para o neurônio de entrada da rede neural;

f) `TListaItemMapeamentoEntrada` é uma classe que gerencia uma lista de itens de mapeamento de entrada (baseados em `TItemBaseMapeamentoEntrada`);

g) `TAtributoMapeadoEntrada` é uma classe que possui um atributo de dado associado a uma lista de itens de mapeamento (`TListaItemMapeamentoEntrada`);

h) `TMapaAtributosEntrada` é uma classe que recebe a lista de atributos de dados (`TAtributos`) e a lista de neurônios da camada de entrada de uma rede neural (`TCamadaRNA`) e efetua a ligação desses atributos com os neurônios e seus mapeamentos (`TAtributoMapeadoEntrada`) , mantendo os mesmos em uma lista;

i) a estrutura `TItemMapaEntrada` é utilizada para manter em um formato de acesso direto a referência da ligação entre um nodo de entrada o seus item de mapeamento;

j) a estrutura `TMapaEntrada` referência um atributo (`TAtributo`) e seus itens de mapeamento (`TItemMapaEntrada)` para acesso direto. Essa estrutura é redundante com as classes de mapeamento e é montada apenas quando a rede é executada, por motivos de desempenho no processamento, pois não efetua iterações em listas como na estrutura das classes;

k) `TTipoTransformação` é uma enumeração indicando se é efetuada alguma transformação do valor de um atributo ao ser transferido esse valor a um neurônio de entrada.

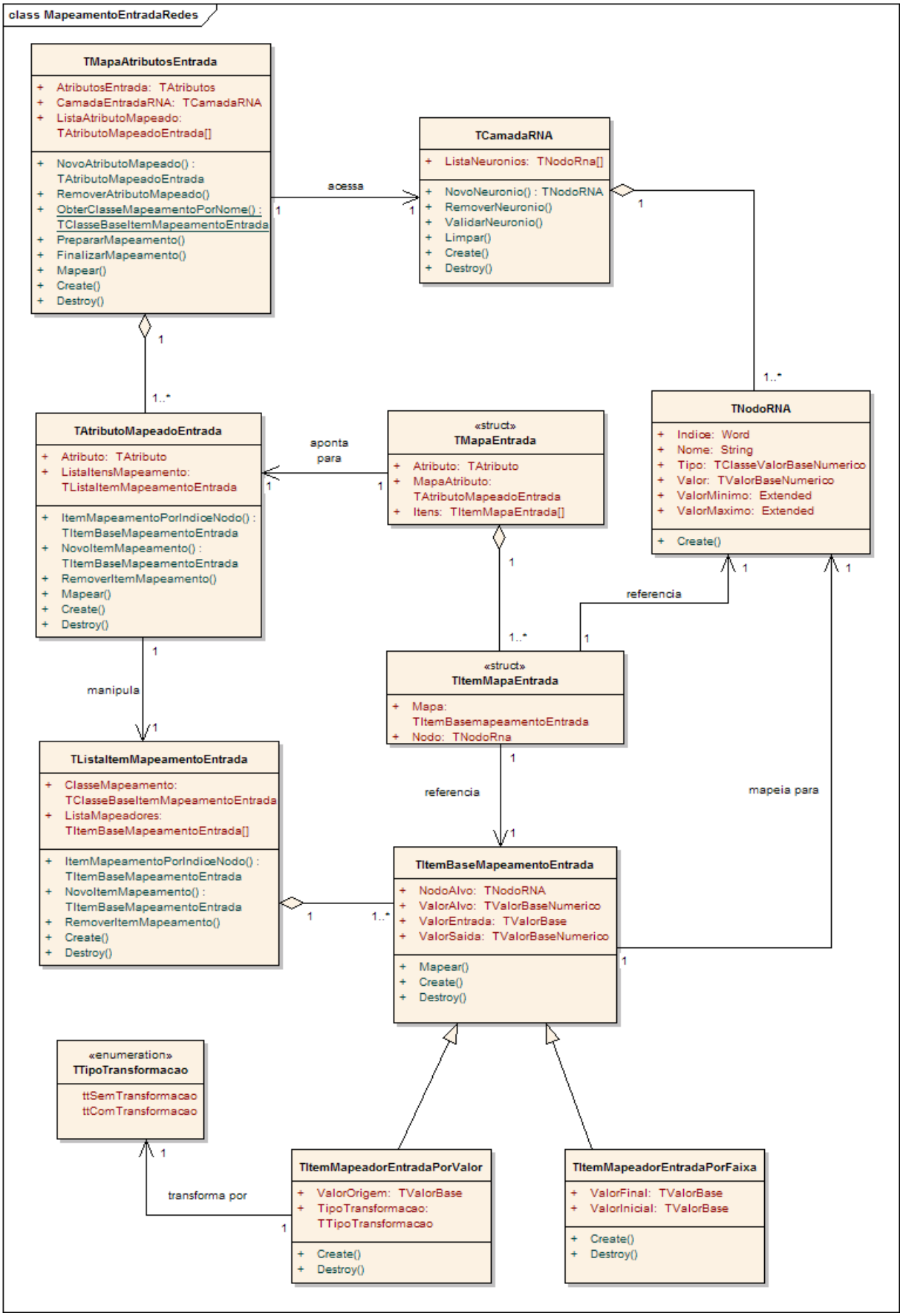


Figura 11 – Diagrama das classes de mapeamento dos valores dos atributos de dados em valores de entrada de redes neurais do pacote `OperadoresBase`

Os artefatos do diagrama da figura 12 tratam da interoperabilidade entre os neurônios da camada de saída das redes neurais e atributos de dados de um fluxo de dados. A saída ou resposta da rede precisa ser armazenada e, para isso, é necessário indicar qual atributo de um fluxo de dados de saída que irá receber o valor de saída da rede, indicando também quais são os neurônios de saída da rede. As classes disponibilizam transformações nos valores de saída da rede antes de serem atribuídos aos atributos que receberão esses valores. Seguem as descrições sucintas de cada classe:

a) `TItemBaseMapeamentoSaida` é uma classe para o mapeamento do valor de saída de um neurônio para o valor de um determinado atributo que armazena um rótulo de classe;
b) `TItemMapeadorSaidaUnico` é uma classe especializada de `TItemBaseMapeamentoSaida` que busca nos neurônios de saída o que possuir o maior valor de saída, assumindo este valor como valor de saída da rede;
c) `TItemMapeadorSaidaMultiplo` é uma classe especializada de `TItemBaseMapeamentoSaida` que analisa o valor de saída de cada neurônio de saída e determina este valor como saída da rede se o mesmo está dentro de uma determinada faixa de valores;
d) `TListaItemMapeamentoSaida` é uma classe que manipula e gerencia uma lista de itens de mapeamento de saída (`TItemBaseMapeamentoSaida`);
e) `TMapaAtributosSaida` é uma classe que recebe o atributo de dado que será atualizado com o valor real da saída da rede (`TAtributo`) e a lista de neurônios da camada de saída de uma rede neural (`TCamadaRNA`). Tal classe efetua a ligação desse atributo com os neurônios e seus mapeamentos de saída (`TListaItemMapeamentoSaida`) e computa também a taxa de acertos e erros da rede;
f) a estrutura `TItemMapaSaida` é utilizada para manter em um formato de acesso direto a referência entre um nodo de saída o seus item de mapeamento;
g) a estrutura `TMapaSaida` referencia o atributo de saída (`TAtributo`) e seus itens de mapeamento (`TItemMapaEntrada`) para acesso direto. Essa estrutura é redundante com as classes de mapeamento e é montada apenas quando a rede é executada, por motivos de desempenho no processamento, pois não efetua iterações em listas como na estrutura das classes.

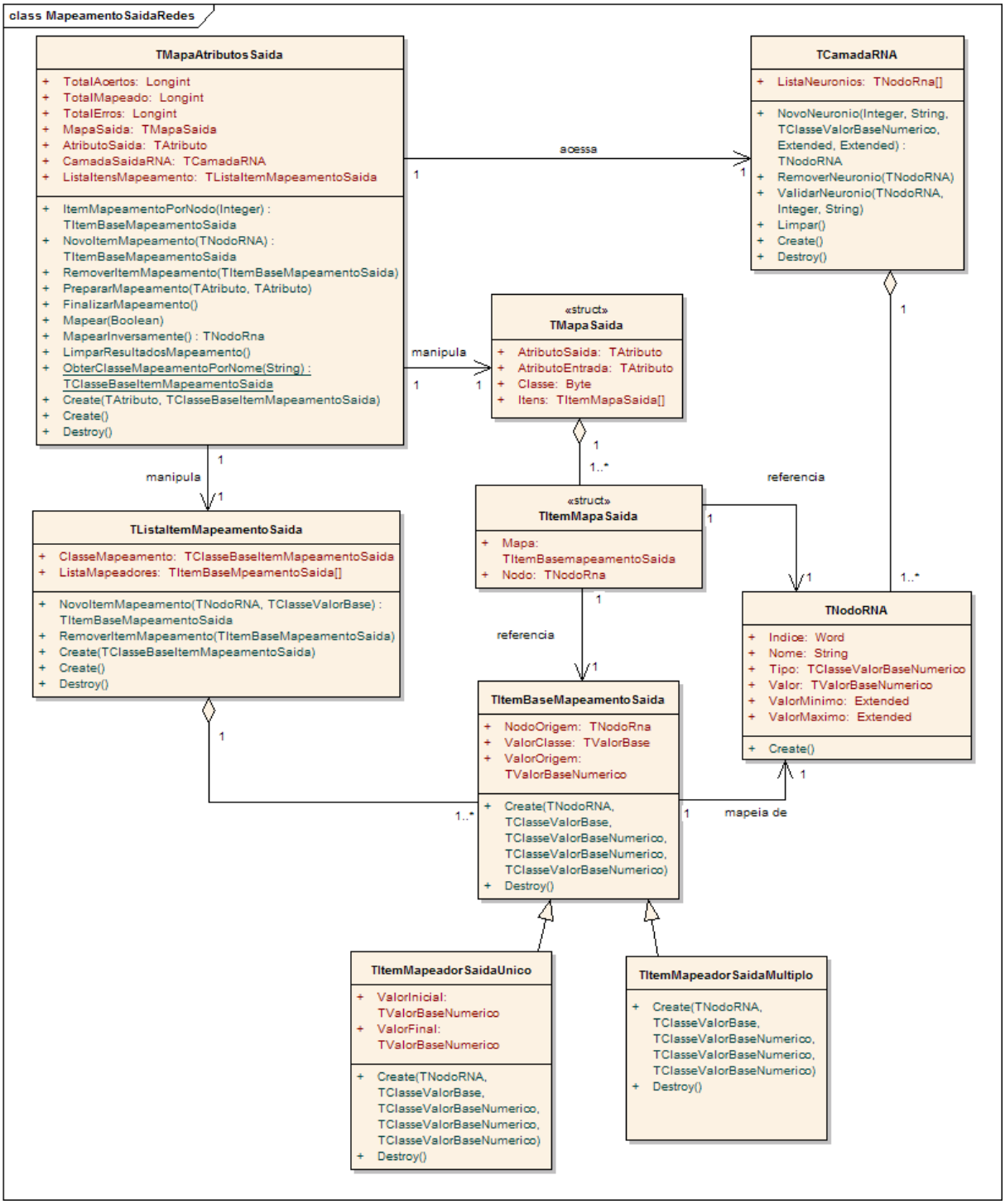


Figura 12 – Diagrama das classes para mapeamento dos valores de saída de redes neurais para valores de atributos de dados do pacote `OperadoresBase`

3.3.2.4 Pacote `Gerenciadores`

O pacote `Gerenciadores` contêm os artefatos que gerenciam os operadores base, demonstrados no diagrama da figura 13. Também estão presentes os artefatos que permitem a

criação e execução de um experimento e sua seqüência de operadores, além dos artefatos para registro, classificação e seleção para uso em um experimento dos operadores externos criados como bibliotecas. Abaixo seguem as descrições dos artefatos do diagrama da figura 13, para os gerenciadores de operadores base:

a) a enumeração `EPontoParada` indica os tipos de possibilidade de parada na execução do experimento, se é feita antes ou depois de executar um determinado operador;

b) `TGerenciadorOperador` é a classe base para gerenciamento de um operador. Essa classe invoca os métodos padrões dos operadores base da classe `TBaseOpeador` do pacote `OperadoresBase`, além de conhecer seus atributos;

c) `TGerenciadorOperadorES` é uma especialização de `TGerenciadorOperador` para gerenciamento de um operador de entrada e saída de dados, especificamente;

d) `TGerenciadorOperadorESBanco` é uma especialização de `TGerenciadorOperadorES` para gerenciamento de um operador de entrada e saída de dados que permite a conexão a um gerenciador de banco de dados;

e) `TGerenciadorOperadorRNA` é uma especialização de `TGerenciadorOperador` para gerenciamento de um operador para manipulação redes neurais artificiais e extração de regras, especificamente.

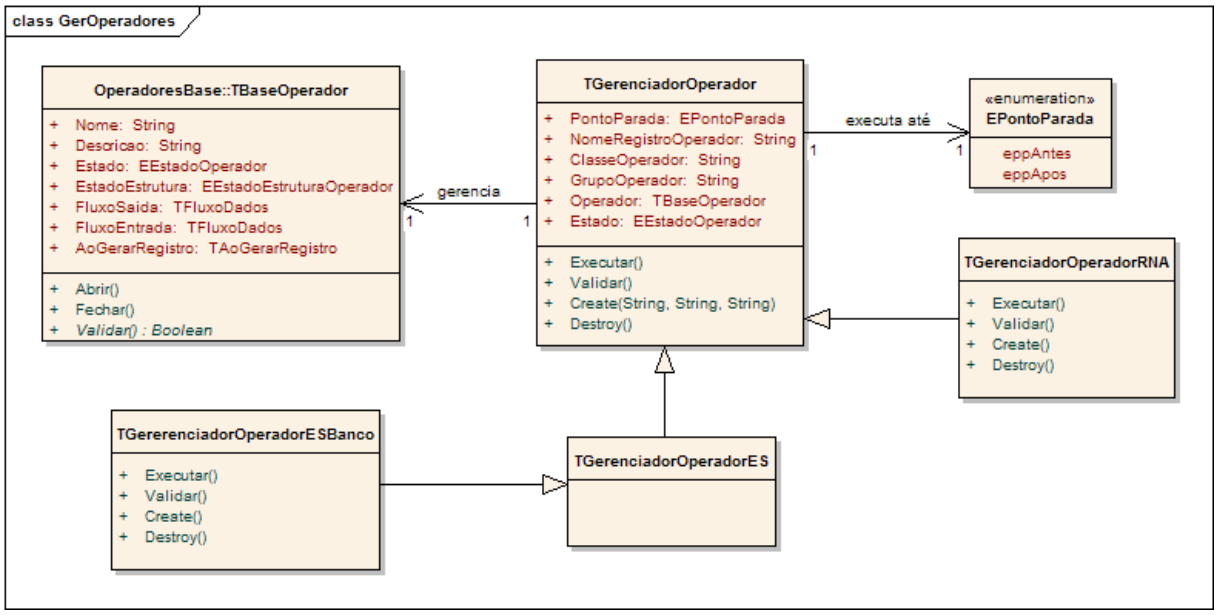


Figura 13 – Diagrama de classes dos gerenciadores de operadores base do pacote `Gerenciadores`

A seguir seguem as descrições dos artefatos do diagrama de classes da figura 14, para os gerenciadores de experimento e de controle dos operadores a serem registrados no protótipo:

a) a classe `TGerenciadorBanco` controla a conexão com o gerenciador de banco de dados e disponibiliza artefatos para persistência e recuperação de dados no banco;
b) `TInfoOperador` é uma classe abstrata que contêm a definição de métodos e atributos básicos necessários a respeito de um operador registrado no protótipo;
c) a classe `TInfoBiblioteca` é uma especialização de `TInfoOperador` usada para manter a instância da biblioteca;
d) a classe `TOperadorPaleta` é uma especialização de `TInfoOperador` usada para identificar o operador através de um grupo com denominação específica;
e) a classe `TGrupoOperador` manipula uma lista dos operadores (`TOperadorPaleta`) de um determinado grupo de nome específico;
f) `TPaleta` é a classe que apresenta ao usuário os grupos e operadores existentes, além de permitir efetuar o registro no protótipo dos operadores externos;
g) a classe `TGerenciadorBiblioteca` é responsável por manter a lista de instâncias das bibliotecas dos operadores e de instanciar e obter informações das mesmas;
h) a classe `TGerenciadorExperimento` é responsável pela persistência e recuperação de um experimento, além da criação de um novo ou exclusão do mesmo;
i) a classe `TCadeiaOperadores` é o artefato que mantêm e manipula a seqüência (ou cadeia) de operadores de um experimento, através de seus gerenciadores, gerenciando assim a inclusão e seqüenciamento dos operadores e de sua execução. Um gerenciador de operador é criado quando é adicionado um operador à cadeia e a instância do operador é manipulada pelo gerenciador, o qual é manipulado pela cadeia. Essa classe também controla a ligação e passagem de fluxos de dados entre os operadores, e a própria criação e destruição de fluxos;
j) `TGOperadores` é uma classe que mantêm a lista de gerenciadores de operadores de um cadeia (seqüência) de operadores;
k) a classe `TExperimento` é uma classe que identifica um experimento e que manipula a cadeia de operadores (`TCadeiaOperadores`).

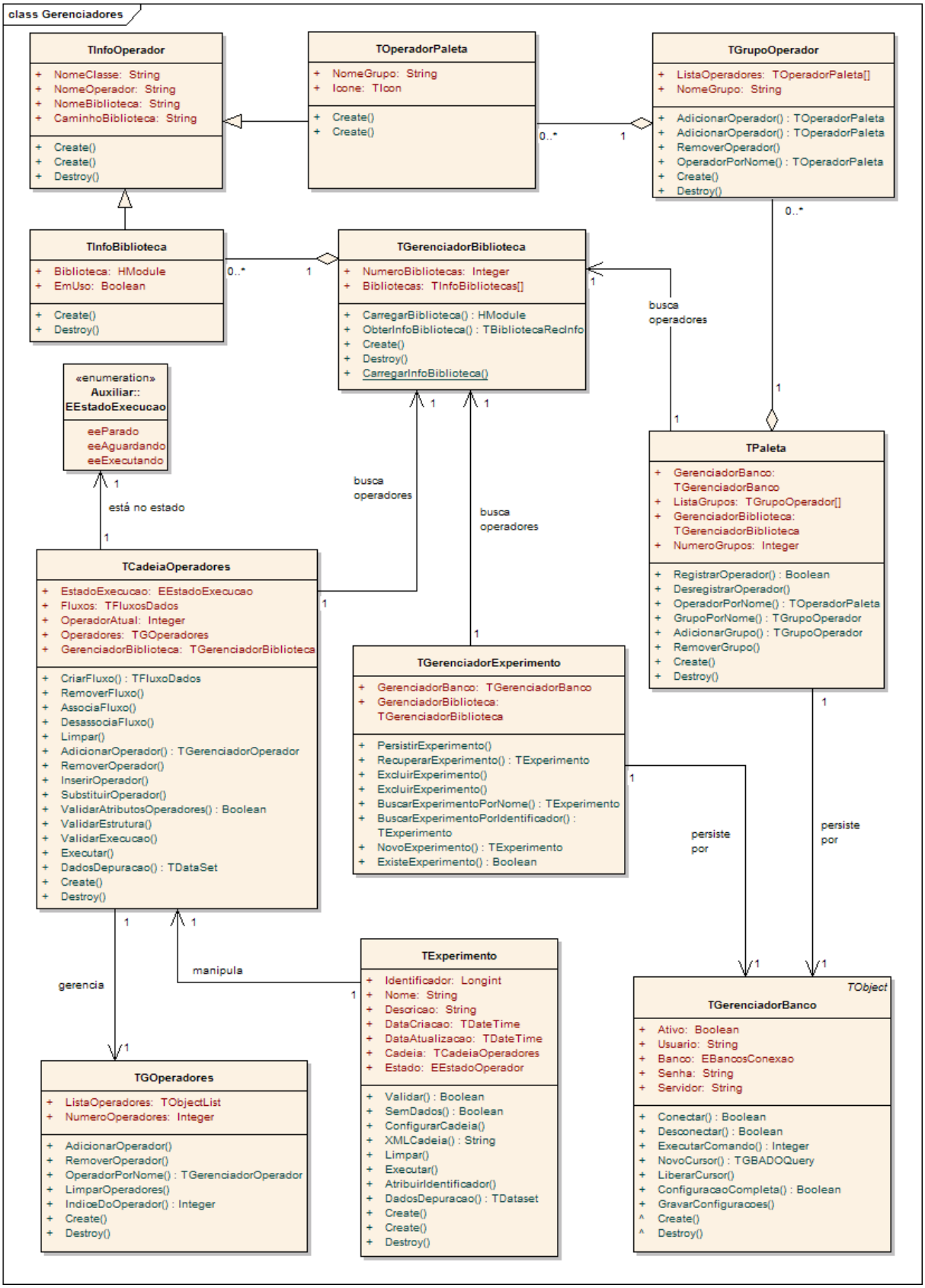


Figura 14 – Diagrama de classes dos gerenciadores de experimento e de registro de operadores do pacote `Gerenciadores`

3.3.2.5 Pacote `Editores`

O pacote `Editores` contêm classes para edição e manipulação pelo usuário dos parâmetros dos operadores de um experimento e dos atributos de dados para efetuar o mapeamento dos atributos para as entradas e saídas de uma rede neural. O pacote está dividido em dois diagramas, sendo que o primeiro (figura 15) define o editor de parâmetros dos operadores do protótipo e o segundo (figura 16) define os editores para atributos e para mapeamento de atributos para entradas e saídas da rede. Para as classes da figura 15, seguem as suas breves descrições:

a) a classe `TEditorObjetos` é a classe que permite manipular os parâmetros de um operador de maneira dinâmica, utilizando a informação de tipo em tempo de execução (*Run Time Type Information* – RTTI) das propriedades publicadas de um objeto. A RTTI é uma característica de aplicativos gerados na linguagem de implementação *Object Pascal*;

b) `TPropInfoList` é uma classe que mantêm uma lista de propriedades de um objeto;

c) `TCustomStringGridProp` é uma classe que permite a edição de valores em formato de *grid*;

d) `TStringGridProp` é uma classe especializada de `TCustomStringGridProp` que permite a edição específica de propriedades disponibilizadas por um objeto da classe `TEditorObjetos`.

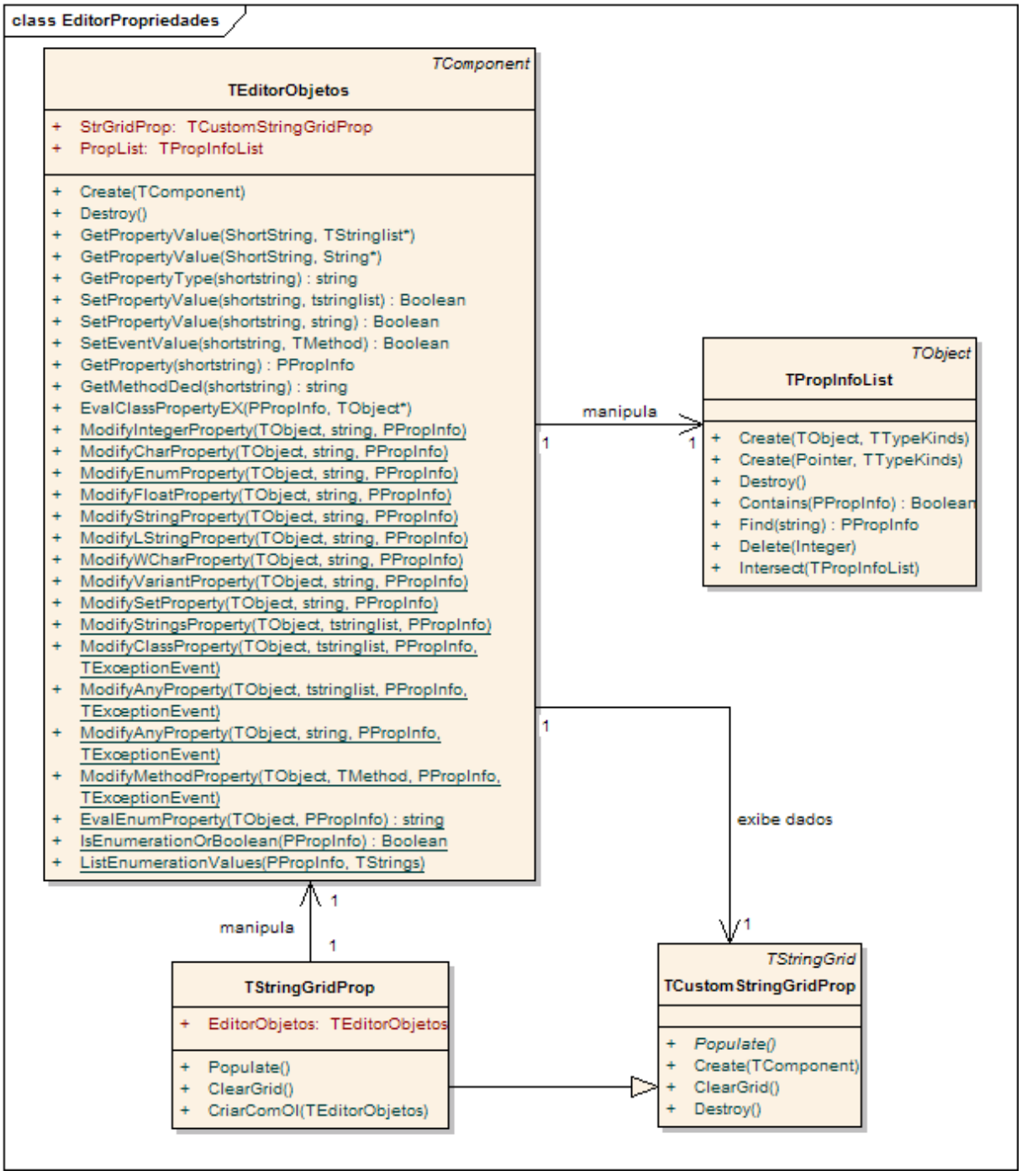


Figura 15 – Diagrama de classes dos editores de propriedades do pacote `Editores`

As classes para o diagrama de classes da figura 16 são descritas a seguir:

a) `TStringEditor` é uma classe que manipula um valor alfanumérico e disponibiliza um recurso visual para edição da mesma, que no caso é a classe `TFrmEditorStrings`;

b) `TFrmEditorStrings` é uma classe que permite a edição de um valor alfanumérico por um usuário de forma visual;

c) `TEditorAtributos` é uma classe que permite manipular um conjunto de atributos

de dados e disponibiliza um recurso visual para edição da mesma, que no caso é a classe `TFrmEditorAtributos`;

d) `TFrmEditorAtributos` é a classe que permite ao usuário a edição visual das informações dos vários atributos em formato de *grid*, como nome, descrição, tipo e tipo de dado;

e) `TAtributosRedeEditorMapaEntrada` é a classe que permite efetuar as ligações entre um conjunto de atributos e os neurônios de entrada de uma rede neural, além dos possíveis mapeamentos. Utiliza-se de um recurso visual para manipulação das ligações e mapeamentos, que no caso é a classe `TFrmEditorAtributoEntradaRNA`;

f) `TFrmEditorAtributoEntradaRNA` é a classe que permite a manipulação visual das ligações entre atributos e neurônios de entrada e seus mapeamentos de valores;

g) `TAtributosRedeEditorMapaSaida` é a classe que permite efetuar a ligação entre os neurônios de saída de uma rede neural e seus possíveis mapeamentos para o atributo que receberá a saída da rede. Disponibiliza um recurso para manipulação visual da mesma, que no caso é a classe `TFrmEditorAtributoSaidaRNA`;

h) `TFrmEditorAtributoSaidaRNA` é a classe que permite a manipulação visual dos atributos e neurônios de saída e seus mapeamentos de valores.

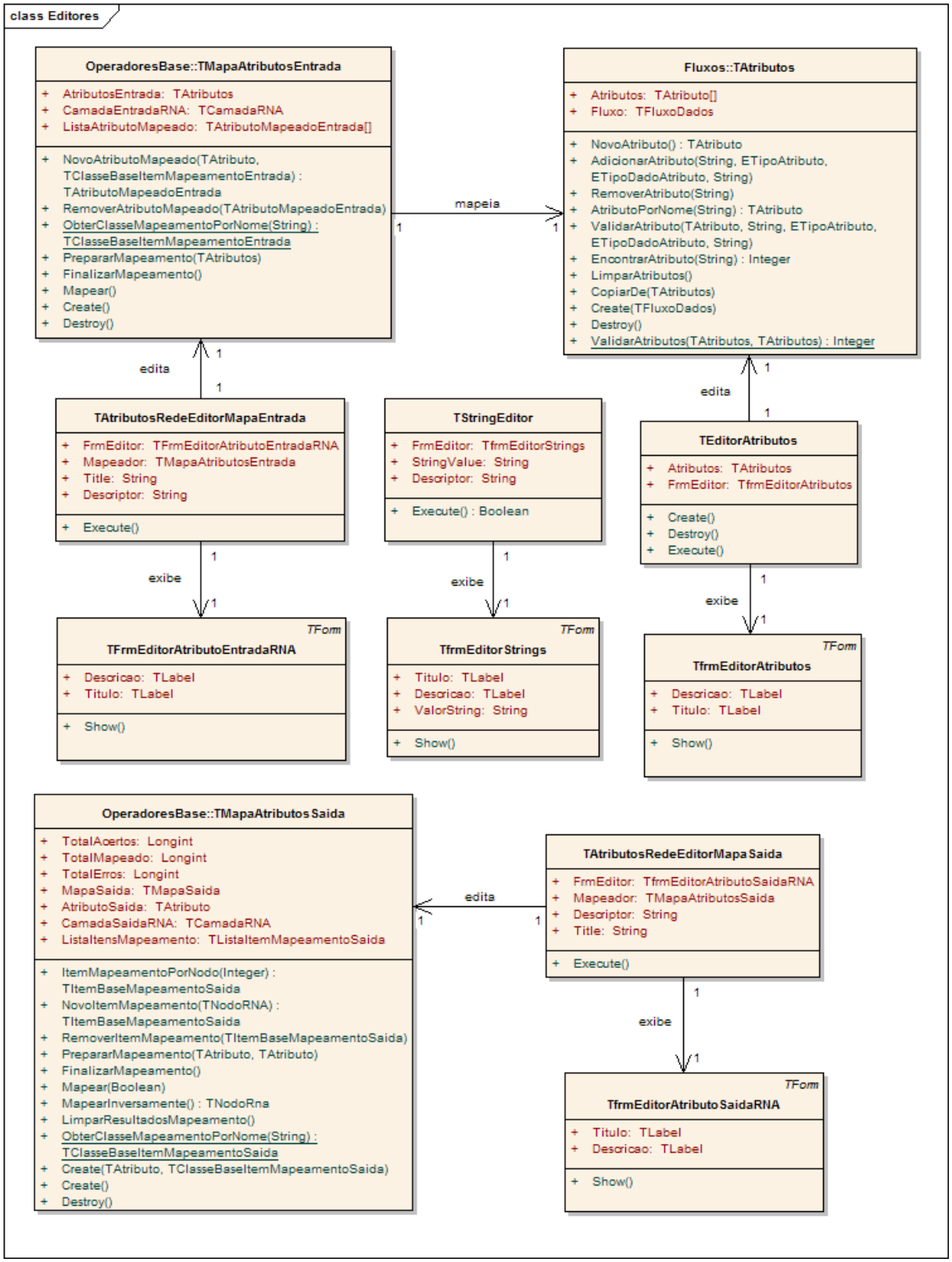


Figura 16 – Diagrama de classes dos editores de atributos e mapeamento de atributos para neurônios de entrada e saída do pacote `Editores`

3.3.2.6 Pacote `OperadoresESBanco`

O pacote `OperadoresESBanco` define apenas a classe `TOperadorESBanco`, descendente de `TBaseOperadorESBanco`. Esta classe descreve um operador de entrada e saída de dados com possibilidade de conexão a um gerenciador de banco de dados, via comandos SQL (*Structured Query Language*). A classe é simples já que a classe base `TBaseOperadorESBanco` já possui os métodos e atributos necessários para a conexão com o gerenciador de banco de dados e para a busca de atributos e de dados. Este operador será registrado no protótipo como operador de entrada e saída de dados. A figura 17 demonstra o diagrama de classes deste pacote.

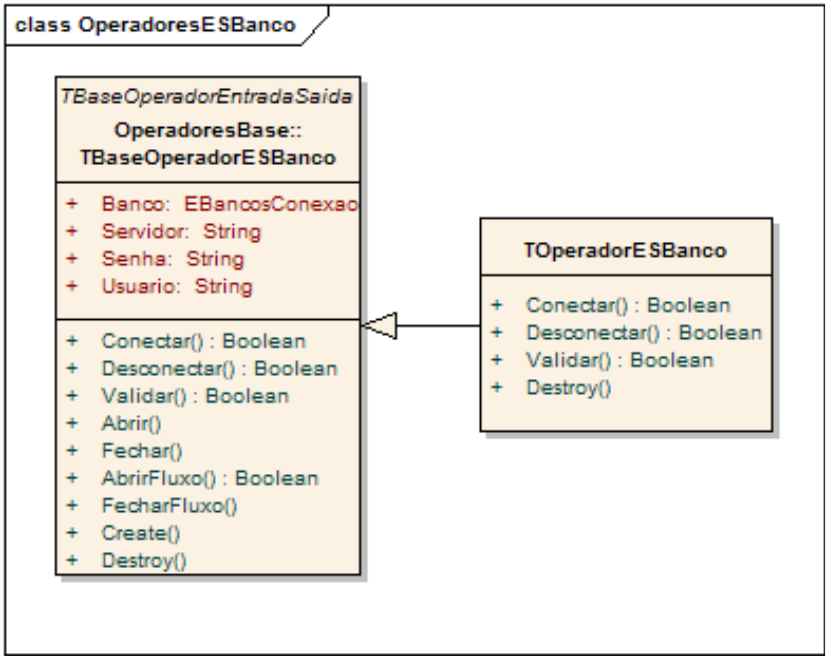


Figura 17 – Diagrama do artefato do pacote `OperadoresESBanco`

3.3.2.7 Pacote `OperadorExtratorRXRNA`

O pacote `OperadorExtratorRXRNA` possui as classes para execução e treinamento de uma rede neural artificial do tipo MLP e as classes para a extração de regras utilizando o algoritmo RX. Ainda existem nesse pacote, classes para a manipulação e avaliação das regras geradas pelo algoritmo. O pacote está divido em dois diagramas para melhor visualização.

O diagrama da figura 18 possui a classe descendente da classe do operador de rede e extração de regras, além das classes da rede neural e das classes para execução do algoritmo

de agrupamento do algoritmo RX, bem como as classes para avaliação das regras geradas. O diagrama da figura 19 demonstra as classes que geram e os as regras após os agrupamentos efetuados e que armazenam e manipulam as regras geradas, efetuando também sua validação. As classes do diagrama da figura 18 são a seguir descritas de forma breve:

a) `TExtratorNBA` é a principal classe do pacote, descendente de `TBaseOperadorRNA`, e que manipula e repassa parâmetros aos objetos das demais classes do pacote, além de invocar métodos para a execução da rede neural, do algoritmo de agrupamento e de extração de regras, e dos métodos para avaliação das regras;
b) `TGerenciadorAgrupamento` é a classe que efetua o agrupamento dos valores de ativação dos neurônios ocultos da rede, possuindo métodos para manipular e obter informações da rede para o propósito do agrupamento;
c) `TEpocaAgrupamento` é uma estrutura usada por `TGerenciadorAgrupamento` que armazena os valores de agrupamento dos neurônios para uma determinada taxa de agrupamento do algoritmo RX;
d) `TModoRede` é uma enumeração usada para indicar qual é o método que deve ser invocado por um objeto da classe `TExtratorNBA` para efetuar ou o treinamento, ou o teste da rede, ou extração de regras ou a validação das regras;
e) `TLayerType` é uma estrutura usada para indicar se uma camada de neurônios da rede neural é de entrada, saída ou oculta;
f) `TLayer` é uma classe base que representa uma camada da rede neural artificial;
g) `TLayerBP` é a especialização de `TLayer` para uma camada de uma rede MLP;
h) `TNeuron` é uma classe base que representa um neurônio da rede neural;
i) `TNeuronBP` é a especialização de `TNeuron` para um neurônio de uma rede MLP;
j) `TNeuralNet` é a classe base que representa uma rede neural artificial;
k) `TNeuralNetBP` é a especialização de `TNeuralNet` para uma rede do tipo MLP;
l) `TNeuralNetExtended` é uma classe derivada de `TNeuralNetBP` que possui métodos específicos para normalização e manipulação dos dados de entrada e saída de rede neural artificial.
m) `TRegra` é uma classe que armazena uma regra, com antecedentes e conseqüentes, permitindo avaliar a mesma e armazenar estatísticas dessas avaliações;
n) `TAvaliadorRegra` é uma classe que manipula uma lista de regras (`TRegras`), invocando métodos para avaliação destas regras e armazenando estatísticas globais das avaliações.

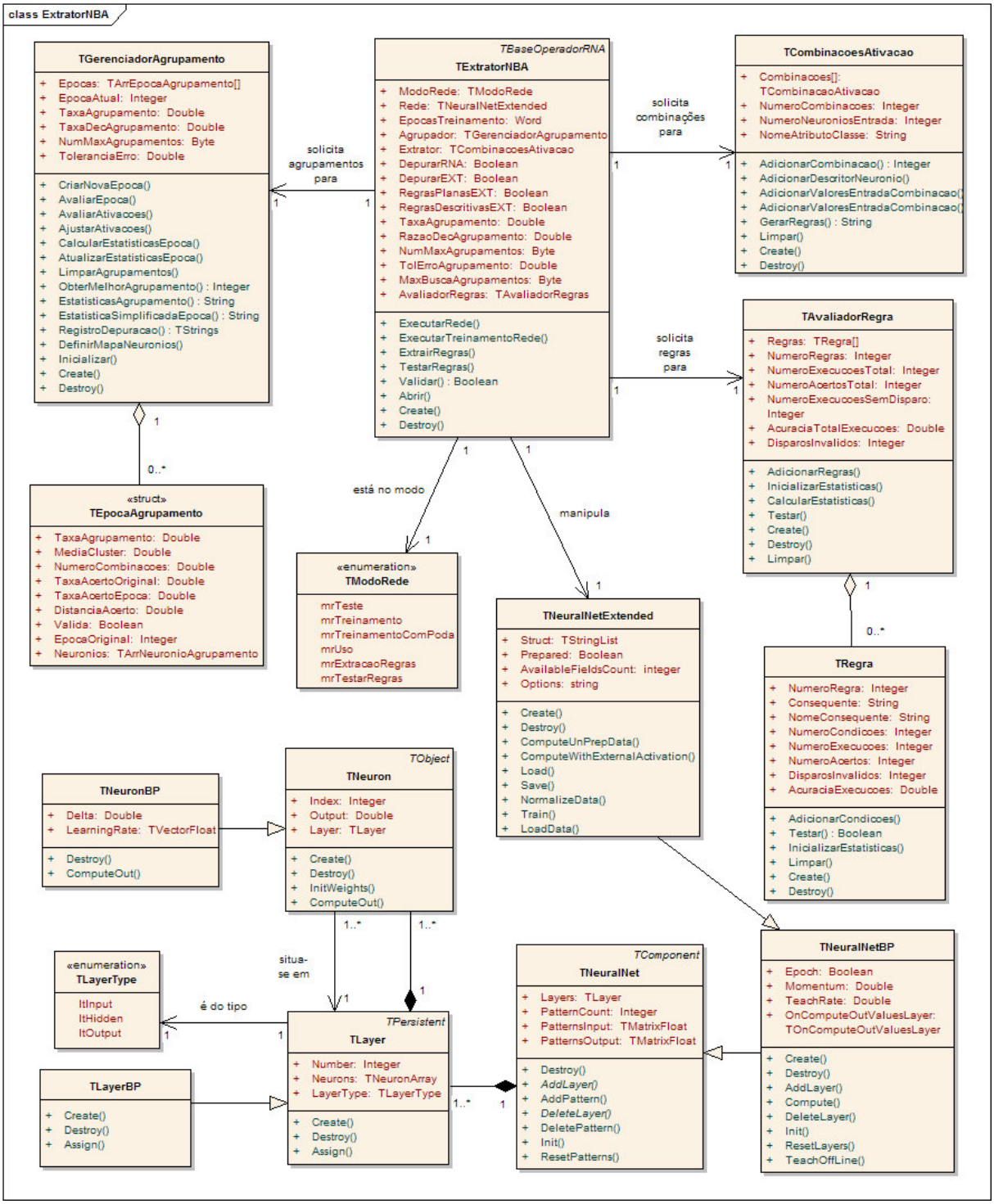


Figura 18 – Primeiro diagrama dos artefatos do pacote `OperadorExtratorRXRNA`

As classes do segundo diagrama do pacote `OperadorExtratorRXRNA` representam as classes para a segunda parte do algoritmo RX, que efetivamente gera as regras, e as classes para avaliação da acurácia das regras geradas. As descrições destas classes são feitas a seguir, de forma breve:

a) `TRecInfoRegras` é uma estrutura utilizada para armazenar informações sobre um neurônio de entrada e o atributo de dados ligado a este neurônio, além de outros

valores necessários a montagem de uma regra no formato `se...então...`;

b) `TFolhaArvore` é uma classe que representa uma condição de uma regra em uma folha de uma estrutura em árvore, que é utilizada visando otimizar o número de condições de uma regra;

c) `TArvoreRegras` é uma classe que mantêm as condições de uma regra em formato de árvore, onde cada condição representa uma folha (`TFolhaArvore`) na árvore e as próximas condições possíveis de uma regra;

d) `TCombinacaoAtivacao` é uma classe que armazena e manipula uma das combinações possíveis entre valores de neurônios agrupados pelo objeto da classe `TGerenciadorAgrupamento`. Esta classe recebe também o rótulo de classe que a combinação dos valores de ativações gera como saída da rede, além dos dados de entrada da rede que levam aos valores de ativação da combinação armazenada, gerando esta classe as regras e armazenando em `TArvoreRegras`;

e) `TConjuntoDadosNeuronios` é uma classe que armazena os valores de entrada que levam a uma determinada combinação de ativações de neurônios ocultos da rede. Os dados são armazenados para cada neurônio da camada de entrada;

f) `TListaValoresDistintos` é uma classe que armazena os valores distintos de um neurônio específico de entrada armazenado em uma instância de `TConjuntoDadosNeuronios`;

g) `TCombinacoesAtivacao` é uma classe que gerencia a lista de combinações (`TCombinacaoAtivacao`) possíveis dos valores de ativação dos neurônios de uma camada oculta da rede neural.

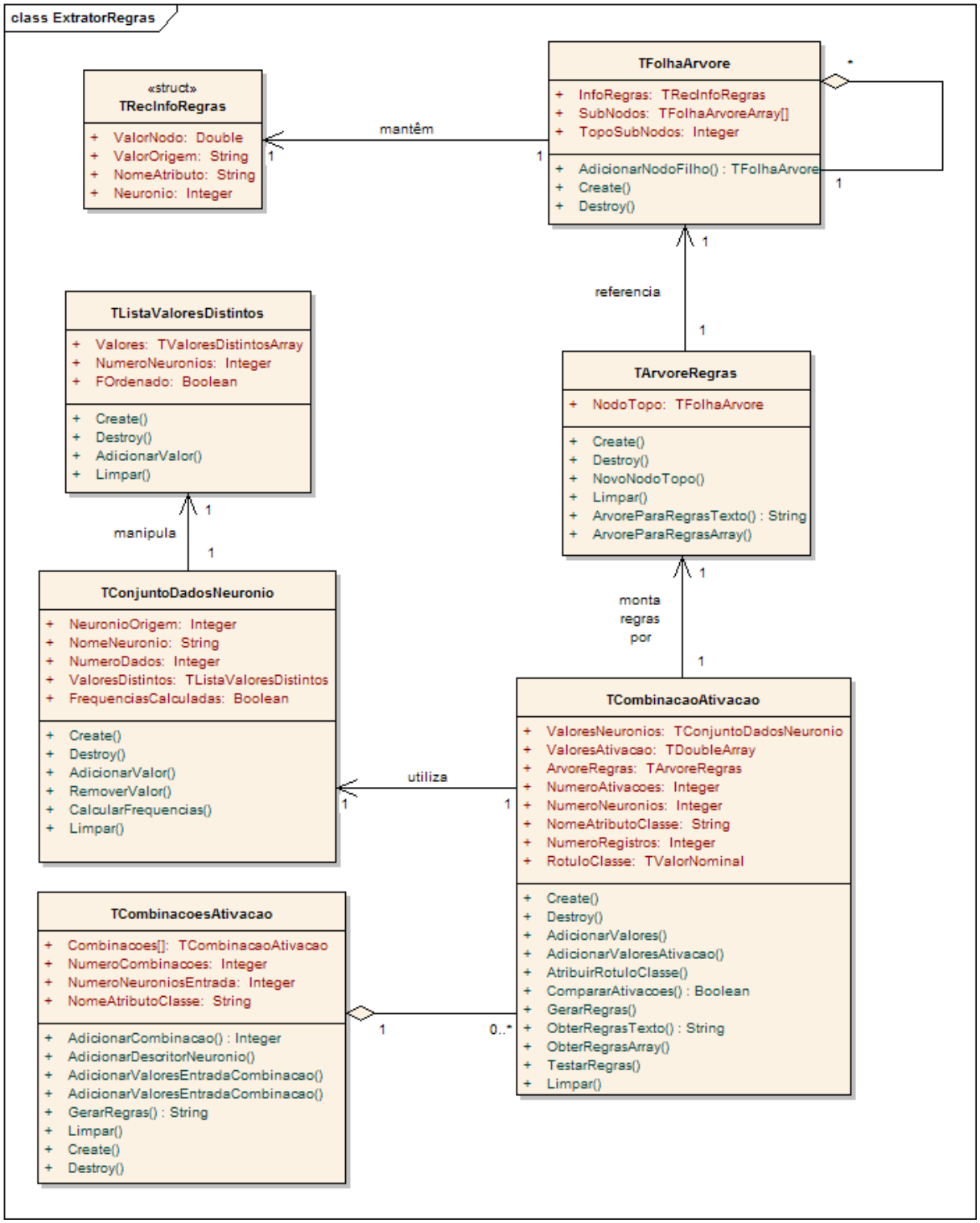


Figura 19 – Segundo diagrama dos artefatos do pacote `OperadorExtratorRXRNA`

3.4 IMPLEMENTAÇÃO

Nas seções a seguir são descritas as técnicas e ferramentas utilizadas na implementação, as principais funcionalidades do protótipo e a implementação das mesmas, além da aplicação de um estudo de caso para demonstrar a operacionalidade da ferramenta.

3.4.1 Técnicas e ferramentas utilizadas

Tanto o protótipo quanto os operadores externos foram implementados na linguagem Object Pascal, utilizando o ambiente de desenvolvimento Delphi 7. A criação da base de dados necessária para armazenamento das informações do protótipo e para os dados do estudo de caso foi feita utilizando-se a ferramenta SQL Server Management Studio Express. Os dados do estudo de caso, antes de serem armazenados no gerenciador de banco de dados, sofreram etapas de pré-processamento e normalização através da ferramenta Yale (Rapid-I, 2007).

3.4.2 Implementação do protótipo

A implementação do protótipo foi feita a partir do desenvolvimento dos artefatos definidos na especificação, com início nos pacotes com menos dependências funcionais para os que mais possuem dependências, e fazendo ajustes ao longo do tempo. A interface com o usuário foi montada posteriormente. Paralelamente ao desenvolvimento da interface, os operadores descendentes das classes base foram desenvolvidos, de forma que se pudesse testar tanto a interface quanto os operadores.

Os artefatos do pacote `Auxiliar` foram desenvolvidos inicialmente por apresentarem menos dependências. As classes principais desse pacote, descendentes de `TValorBase` tem o propósito de facilitar a atribuição ou obtenção de valores de dados de atributos entre estes e os neurônios de entrada e saída das redes neurais artificiais. Elas efetuam conversões implícitas através dos métodos específicos, transformando quando possível, o tipo de dado conforme for solicitado.

Os artefatos do pacote `OperadoresBase`, e no caso das classes de operadores base, possuem implementações simples, devido ao fato de que a maioria são classes bases e possuindo apenas métodos declarados como virtuais e poucos métodos com implementação interna. Sua finalidade é prover métodos e atributos que devem ser sobrescritos pelos operadores descendentes, de forma que o gerenciador do experimento conheça e execute os métodos apropriados, independente da implementação na classe descendente. A seção de declaração do operador `TBaseOperador`, é exibido no quadro 16.

```
TBaseOperador = class
private
  FNome : String;
  FDescricao : String;
  FAoConfigurar : TAoConfigurar;
  FAoGerarRegistro : TAoGerarRegistro;
  FFluxoEntrada : TFluxoDados;
  FFluxoSaida : TFluxoDados;

  function GetNome: String;
  procedure SetNome(const aNome: String);
  function GetEstado: EEstadoOperador;
  function GetDescricao: String;
  procedure SetDescricao(const aDescricao: String);
  function GetGerarRegistro: TAoGerarRegistro;
  procedure SetGerarRegistro(const aAoGerarRegistro: TAoGerarRegistro);

protected
  FEstado : EEstadoOperador;
  FEstadoEstrutura : EEstadoEstruturaOperador;

  function GetEstadoEstrutura: EEstadoEstruturaOperador; virtual;
  procedure SetAoConfigurar(aAoConfigurar : TAoConfigurar); virtual;
  function GetAoConfigurar : TAoConfigurar; virtual;
  procedure AoConfigurar; virtual;
  procedure SetFluxoSaida(const aFluxoSaida: TFluxoDados); virtual;
  function GetFluxoSaida : TFluxoDados; virtual;
  procedure SetFluxoEntrada(const aFluxoEntrada: TFluxoDados); virtual;
  function GetFluxoEntrada : TFluxoDados; virtual;
  procedure GerarRegistro(aTipoRegistro : ETipoRegistro; aRegistro : String); virtual;
public
  procedure Abrir; virtual;
  procedure Fechar; virtual;
  function Validar: Boolean; virtual; abstract;

  property FluxoSaida : TFluxoDados read GetFluxoSaida write SetFluxoSaida;
  property FluxoEntrada : TFluxoDados read GetFluxoEntrada write SetFluxoEntrada;
  property AoGerarRegistro: TAoGerarRegistro read GetGerarRegistro write SetGerarRegistro;

published
  property Nome : String read GetNome write SetNome;
  property Descricao : String read GetDescricao write SetDescricao;
  property Estado : EEstadoOperador read GetEstado;
  property EstadoEstrutura : EEstadoEstruturaOperador read GetEstadoEstrutura;
  property Configurar : TAoConfigurar read GetAoConfigurar write SetAoConfigurar;

  constructor Create;
  destructor Destroy; override;
end;
```

Quadro 16 – Seção declarativa da implementação da classe `TOperadorBase`

Deste parágrafo em diante, serão exibidas as telas e as funcionalidades mais relevantes do protótipo, destacando em cada ponto os artefatos utilizados em tais funcionalidades. A tela principal do protótipo com um experimento em andamento é demonstrada na figura 20.

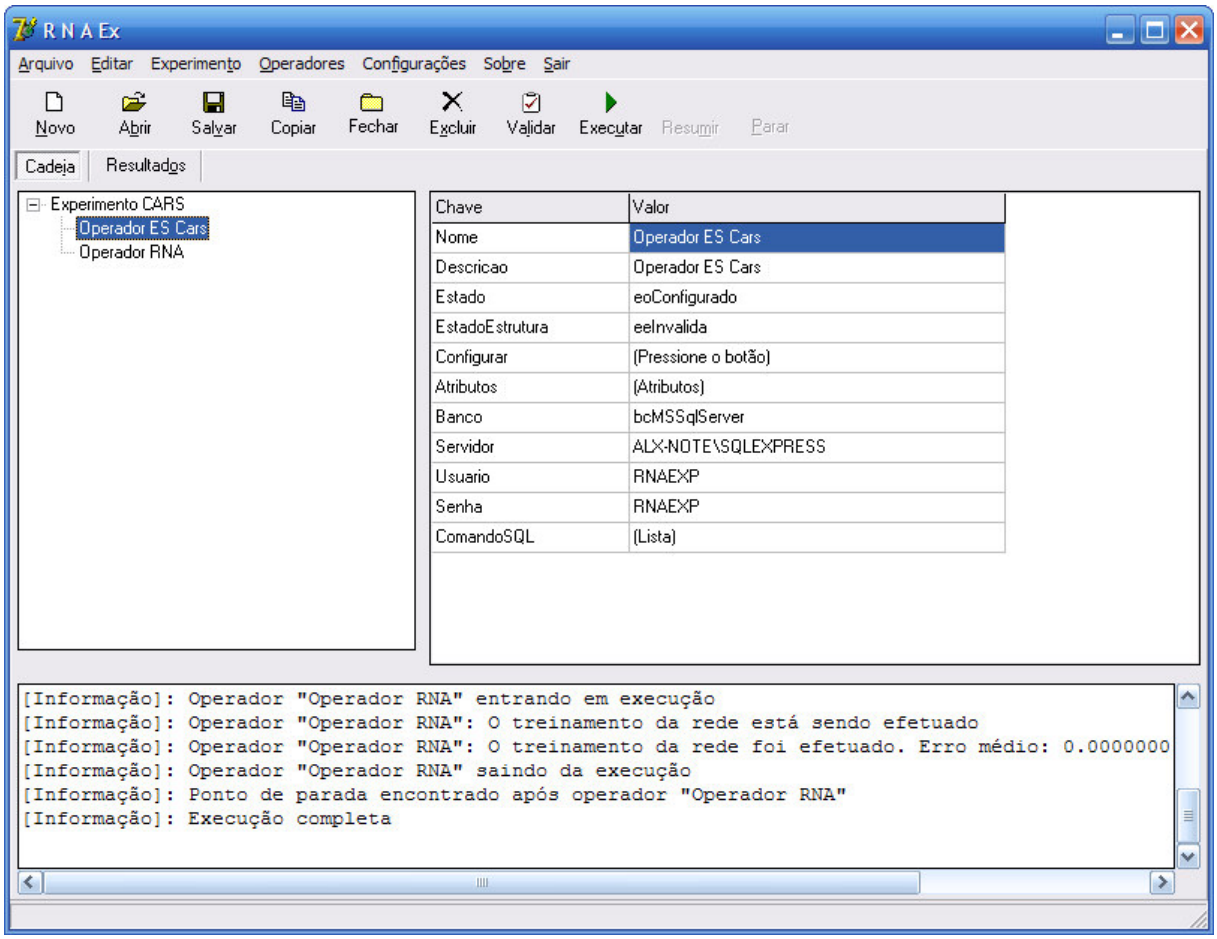


Figura 20 – Ilustração da tela principal do protótipo

A cadeia de experimentos é visualizada como uma árvore e no topo da mesma é visualizado o experimento que está aberto. Cada operador é um elemento dentro da árvore, abaixo do experimento. À direita da árvore encontra-se o editor de propriedades, onde são listadas e permitidas edições dos diversos parâmetros do operador em foco na árvore. Na parte inferior é exibida uma caixa de texto onde são exibidas informações da execução do experimento. A parte superior exibe um *menu* de opções além de uma seqüência de botões de atalho para as principais opções.

O registro dos operadores é feito pelo *menu* `Operadores`, onde se abre uma caixa de diálogo para selecionar a biblioteca de ligação dinâmica desejada, a qual deve possuir um operador que encapsula um algoritmo específico. A figura 21 exibe a seqüência de passos para registro do operador e a figura 22 exibe a adição de um operador registrado no protótipo em um experimento. Os operadores são divididos em dois grupos, conforme são as classes base, sendo o primeiro grupo o de operadores de busca e manipulação de dados e o segundo grupo o de operadores de manipulação e extração de regras de redes neurais artificiais. O quadro 17 mostra o código fonte do método `RegistrarOperador` classe `TPaleta`, que efetua o registro da biblioteca.

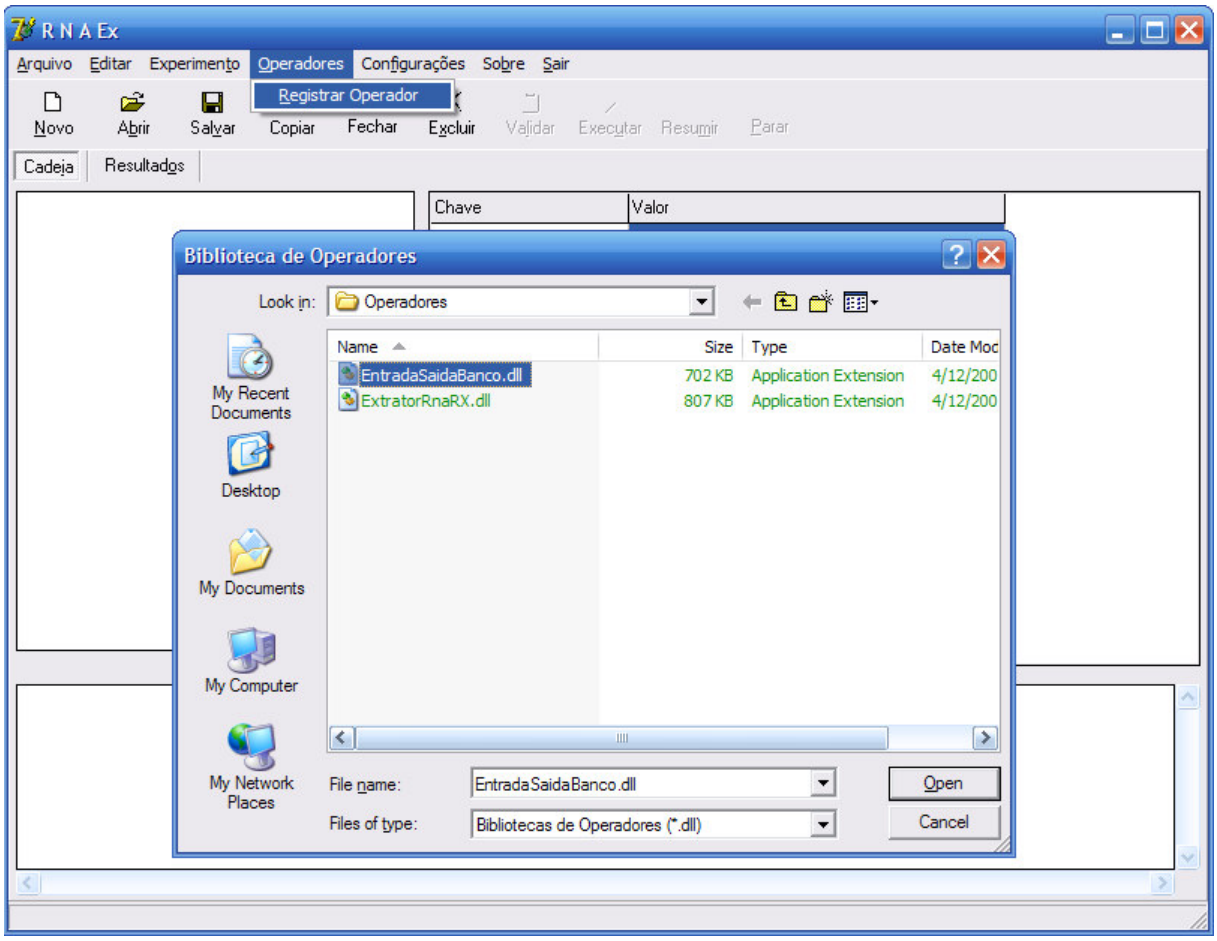


Figura 21 – Registro de um operador no protótipo

```
function TPaleta.RegistrarOperador(const aNomeBiblioteca: String;
const aCaminhoBiblioteca: String): Boolean;
var
  xRegInfo : TBibliotecaRecInfo;
  xGrupo   : TGrupoOperador;
begin
  xRegInfo := FGerenciadorBiblioteca.ObterInfoBiblioteca(aCaminhoBiblioteca,
                                                         aNomeBiblioteca);
  xGrupo := GrupoPorNome(xRegInfo.InfoGrupo.DescricaoGrupo);

  if  xGrupo = nil then
      xGrupo := AdicionarGrupo(xRegInfo.InfoGrupo.DescricaoGrupo);

  xGrupo.AdicionarOperador(xRegInfo, nil);

  Result := SalvarPaleta;
end;
```

Quadro 17 – Método `RegistrarOperador` da classe `TPaleta`

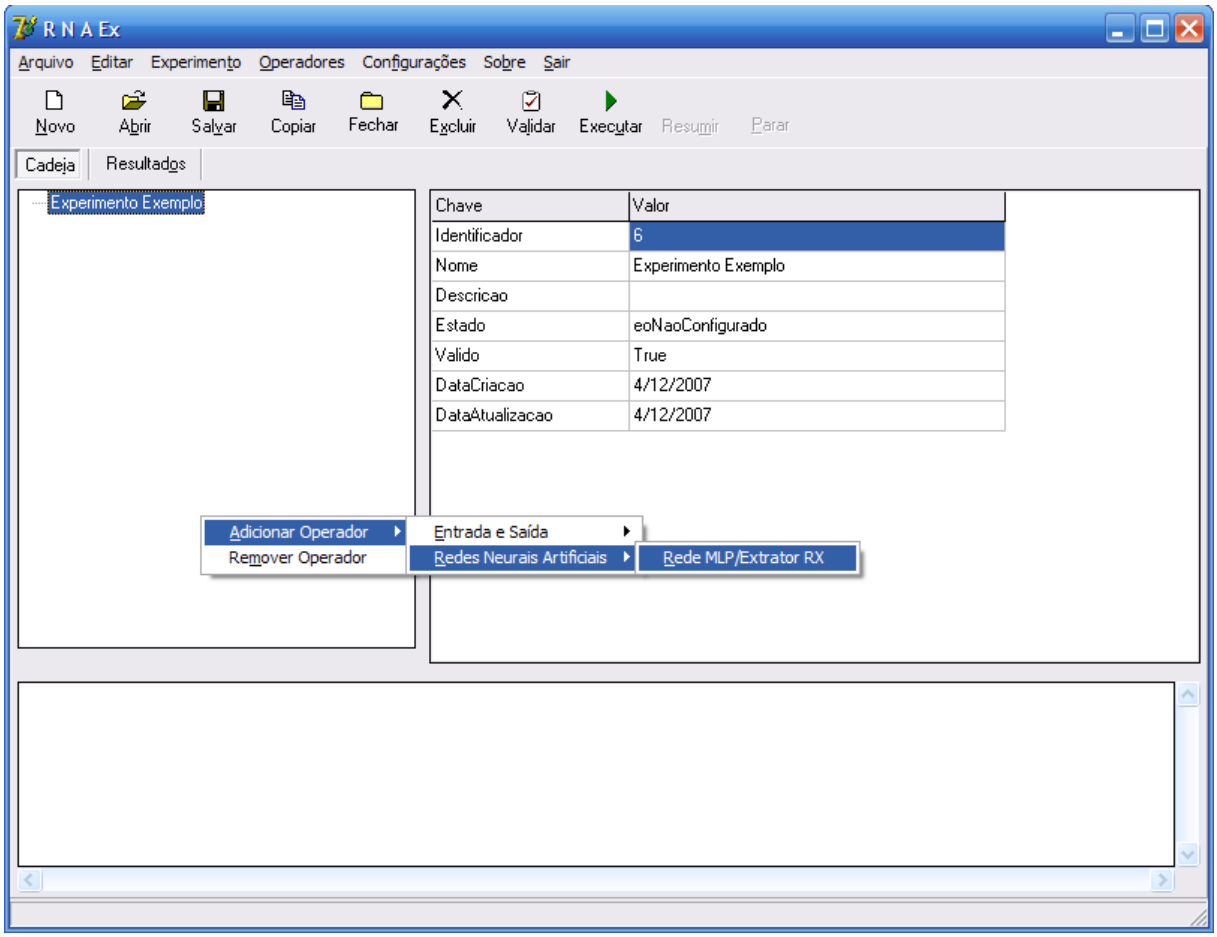


Figura 22 – Adição de um operador no protótipo

O trecho de código no quadro 18 ilustra o método `AdicionarOperador` da classe `TCadeiaOperadores`. Esse método utiliza-se do gerenciador de biblioteca (`TGerenciadorBiblioteca`) para carregar a biblioteca e instanciar o operador. Se existirem operadores anteriores à posição onde está sendo inserido o operador atual, o fluxo de dados (da classe `TFluxos`) de saída do operador de saída é ligado como fluxo de dados de entrada ao novo operador. Após isso, o método verifica qual é a classe do operador e chama um método para criar o gerenciador específico do mesmo e associar ambos. Em seguida um novo fluxo de dados é criado e associado ao operador como fluxo de saída. Finalmente, o gerenciador de operador recém-criado é adicionado na lista de gerenciadores de operadores da cadeia (da classe `TGOperadores`).

```
function TCadeiaOperadores.AdicionarOperador(aInfoOperador: TInfoOperador;
const aNomeGrupo, aNomeRegistro : String) : TGerenciadorOperador;
var
  xFluxoEnt : TFluxoDados;
  xOperador : TBaseOperador;
  xGOperadorAnterior : TGerenciadorOperador;
begin
  Result := Unassigned;

  if  (Assigned(aInfoOperador)) then
  begin
    // Carrega a biblioteca e instância o operador
    FGerenciadorBiblioteca.CarregarBiblioteca(aInfoOperador.CaminhoBiblioteca,
                                              aInfoOperador.NomeBiblioteca, xOperador);

    // obtêm o operador anterior e o fluxo de saída deste operador
    xFluxoEnt := nil;

    if  (FOperadores.NumeroOperadores > 0) then
    begin
      xGOperadorAnterior := FOperadores[FOperadores.NumeroOperadores -1];
      if  Assigned(xGOperadorAnterior) and (Assigned(xGOperadorAnterior.Operador)) then
         xFluxoEnt := xGOperadorAnterior.Operador.FluxoSaida;
    end;

    if  Assigned(xOperador) then
    begin
      // atribui função para gerar registro externamente
      xOperador.AoGerarRegistro := FAoGerarRegistro;

      if  Utils.ClasseSuportada(xOperador, TBaseOperadorESBanco) then
      begin
        Result := CriarGerenciadorOperador(TGererenciadorOperadorESBanco,
                                           TBaseOperadorESBanco, xOperador,
                                           aNomeGrupo, aNomeRegistro);

        TBaseOperadorESBanco(Result.Operador).FluxoEntrada:= xFluxoEnt;
        Result.Operador.FluxoSaida := FFluxos.NovoFluxo;
      end
      else
        if  Utils.ClasseSuportada(xOperador, TBaseOperadorRNA) then
        begin
          Result := CriarGerenciadorOperador(TGerenciadorOperadorRNA, TBaseOperadorRNA,
                                             xOperador, aNomeGrupo, aNomeRegistro);
          TBaseOperadorRNA(Result.Operador).FluxoEntrada:= xFluxoEnt;

          Result.Operador.FluxoSaida := FFluxos.NovoFluxo;
        end;
    end;
    if  Assigned(Result) then
      FOperadores.AdicionarOperador(Result);
  end;
end;
```

Quadro 18 – Método `AdicionarOperador` da classe `TCadeiaOperadores`

A tela da figura 23 exibe um exemplo da lista de parâmetros dos operadores que podem ser editados no protótipo. A lista de parâmetros de um operador é exibida quando o operador recebe foco na árvore de operadores. Os parâmetros são extraídos através da utilização da informação de tipo em tempo de execução (RTTI), técnica da linguagem Object Pascal para listar, obter e atribuir dados dinamicamente em tempo de execução para propriedades publicadas (atributos ou métodos) de objetos. Assim, para que um parâmetro de um objeto operador esteja visível para o usuário editar no protótipo, é necessário utilizar na classe do operador a diretiva `published` (publicada) antes das propriedades que representam métodos ou atributos da linguagem Object Pascal.

A classe `TEditorObjetos` é responsável por manipular as propriedades publicadas de um objeto e exibir as mesmas em formato de *grid* ao usuário para manipulação dos valores. Um trecho de código exibindo as declaração das propriedades publicadas da classe do operador `TOperadorExBase` é exibido no quadro 19. Já no quadro 20 é visualizado o método `EvalAnyProperty` da classe `TEditorObjetos`, utilizado para obter as informações de uma propriedade de um objeto. Como exemplo, são visíveis na *grid* da figura 23 a propriedade `EstadoEstrutura`, que é uma variável, e a propriedade `Configurar`, que é um método. Ambas as propriedades são publicadas na classe base de qualquer operador.

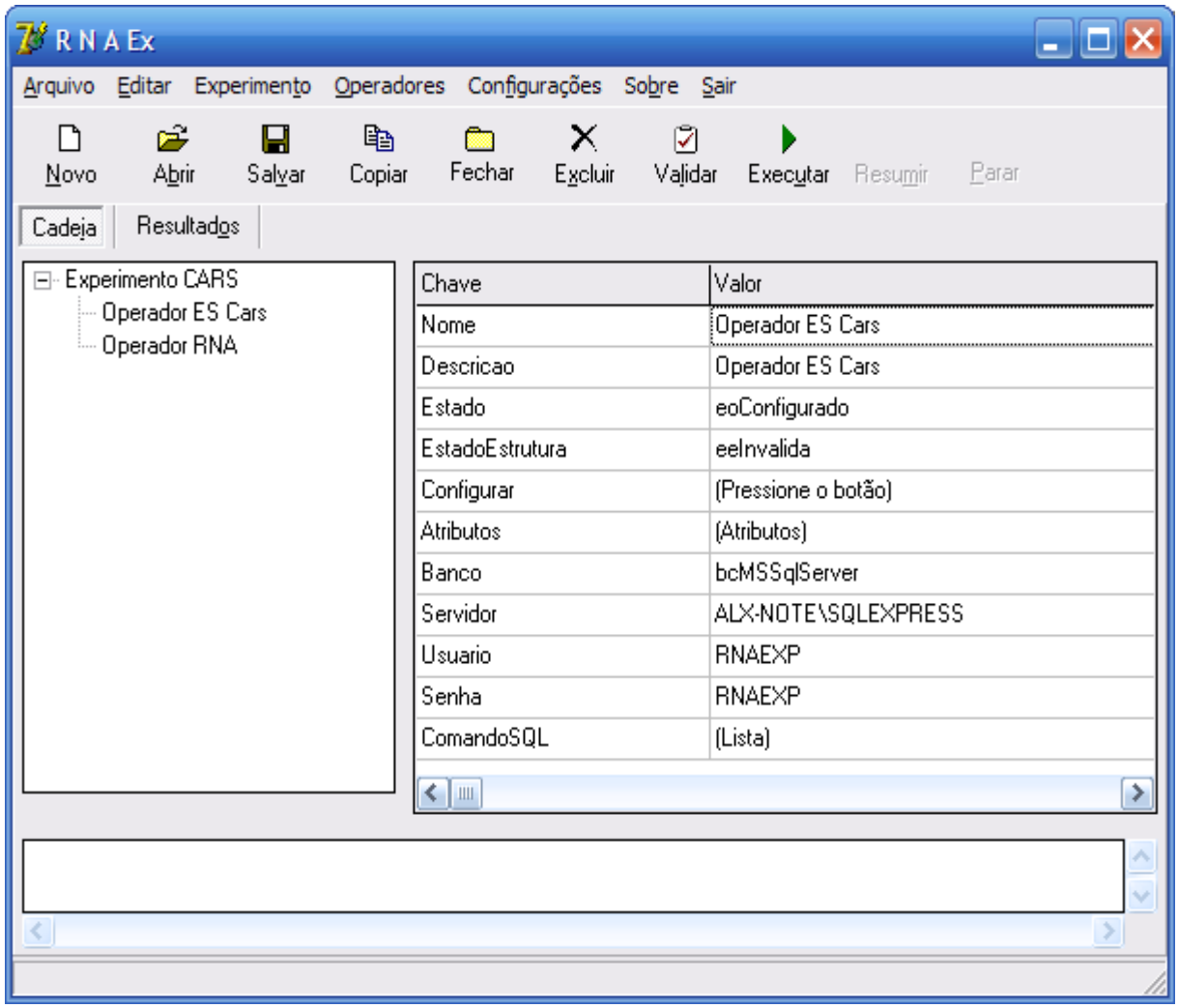


Figura 23 – *Grid* de visualização e edição dos parâmetros disponíveis de um operador

```
TOperadorExBase = class(TBaseOperadorESExemploBanco)
private
  .
  .
public
  .
  .
published
  property EstadoEstrutura : EEstadoEstruturaOperador read GetEstadoEstrutura;
  property Configurar : TAoConfigurar read GetAoConfigurar write SetAoConfigurar;
end;
```

Quadro 19 – Propriedades publicadas na classe `TOperadorExBase`

```
procedure TEditorObjetos.EvalAnyProperty(PropInfo: PPropInfo; var S: TStringList);
begin
    S.Clear;
    if PropInfo <> nil then begin
       case PropInfo^.PropType^.Kind of
         tkInteger: s.add(EvalIntegerProperty(PropInfo));
         tkChar: s.add(EvalCharProperty(PropInfo));
         tkEnumeration: s.add(EvalEnumProperty(FObject, PropInfo));
         tkFloat: s.add(EvalFloatProperty(PropInfo));
         tkLString: s.add(EvalLStringProperty(PropInfo));
         tkWChar: s.add(EvalWCharProperty(PropInfo));
         tkVariant: s.add(EvalVariantProperty(PropInfo));
         tkString: s.add(EvalStringProperty(PropInfo));
         tkSet: s.add(EvalSetProperty(PropInfo));
         tkMethod: s.add(EvalMethodProperty(PropInfo));
         tkclass: EvalclassProperty(PropInfo,s);
      end;
    end;
  end;
end;
```

Quadro 20 – Método `EvalAnyProperty` da classe `TEditorObjetos`

A *grid* da figura 23 exibe ainda o parâmetro `Atributos`, que é um objeto da classe `TAtributos` que mantém a lista de atributos do fluxo de dados do operador. No caso da figura 23, é apresentado o operador de entrada e saída de dados com conexão a um banco dados, implementado para o protótipo. Este operador possui um fluxo de dados e a edição de seus atributos é feita pela classe `TEditorAtributos` e `TfrmEditorAtributos`, que exibe a tela da figura 24, quando o parâmetro é editado.

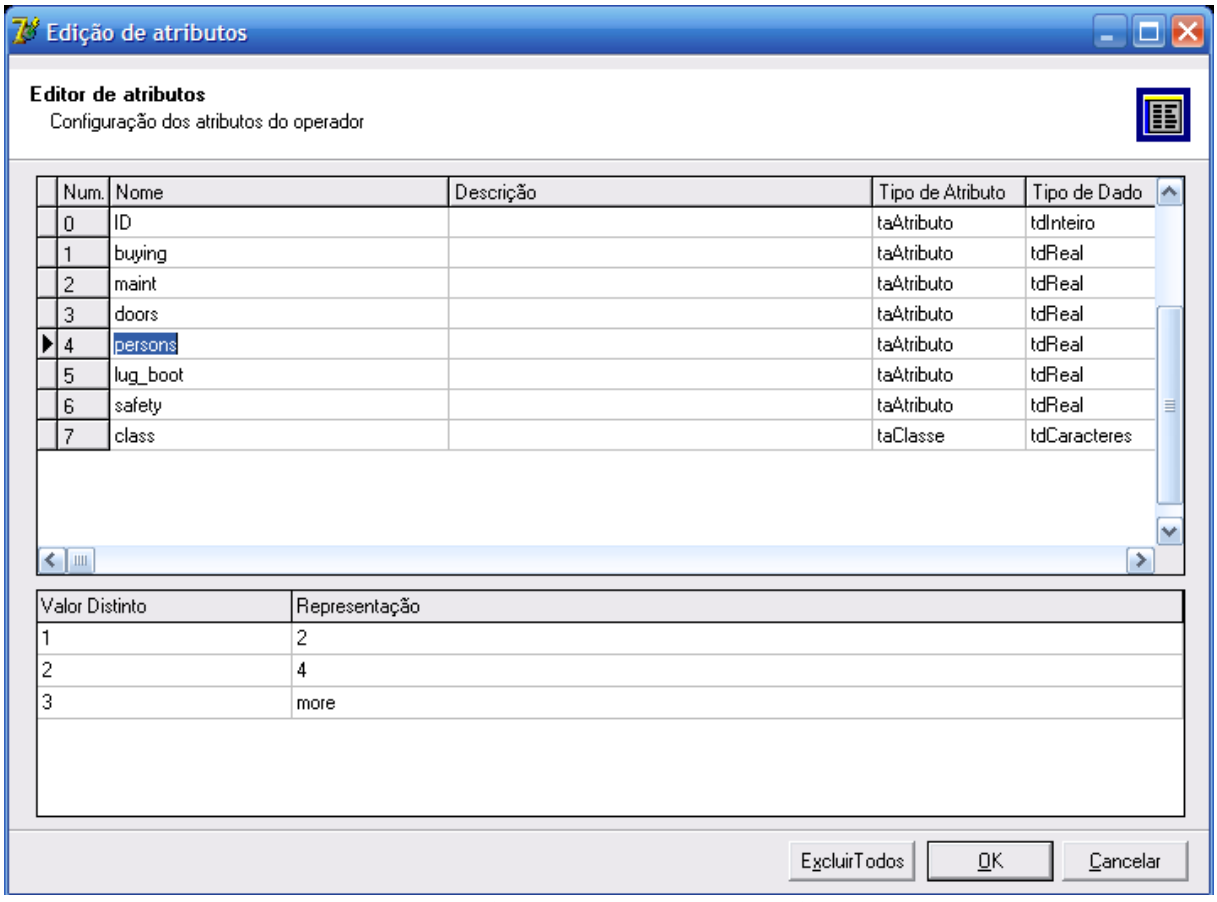


Figura 24 – *Grid* de visualização e edição de atributos de dados

Os atributos possuem uma identificação numérica seqüencial, o seu nome, uma breve descrição, o tipo de atributo e o tipo de dado. O tipo de atributo identifica se o mesmo é um atributo de dado ou um atributo que é rótulo de classe (para os dados de exemplo na tarefa de classificação em *data mining*) e para alimentar o algoritmo de treinamento da rede neural.

Um atributo pode também ser indicado como atributo de predição. Tal atributo seria um atributo extra não oriundo da fonte de dados, utilizado para armazenar o resultado da saída de rede neural artificial. Somente seria utilizado no fluxo de saída da rede neural. O tipo de dado identifica o tipo de valor que o atributo assume, podendo ser alfanumérico, numérico inteiro, numérico de ponto flutuante, de data e hora ou de tipo de dado desconhecido.

O operador de entrada e saída implementados possui um parâmetro essencial chamado `ComandoSql`, que permite incluir um comando em linguagem SQL que será usado para extrair os dados da base de dados, em tempo de configuração para a busca das informações dos atributos e em tempo de execução para a busca dos dados.

A figura 25 exibe na *grid* os parâmetros do operador de manipulação de rede neural e extração de regras. O operador gerado a partir da classe base de operadores de redes neurais (`TBaseOperadorRNA`) possui pelo menos dois fluxos de dados, um para entrada da rede e outro para a saída, exibindo os atributos dos dois como parâmetros. Dois outros parâmetros são importantes, sendo eles o `MapeamentoEntrada` e o `MapeamentoSaida`, respectivamente das classes `TMapaAtributosEntrada` e `TMapaAtributosSaida`. Os dois parâmetros especificam as formas de ligações e conversões entre os atributos do fluxo de dados entrada e os neurônios de entrada e entre os neurônios de saída e os atributos do fluxo de dados de saída.

A edição do parâmetro `MapeamentoEntrada` é feito pelas classes `TAtributoRedeEditorMapaEntrada`, que define e mantêm as regras de mapeamento, e `TFrmEditorAtributoEntradaRNA` que é classe de formulário que exibe a tela da figura 26 para edição do mapeamento pelo usuário. Já a edição do parâmetro `MapeamentoSaida` é feito pelas classes `TAtributosRedeEditorMapaSaida`, que é a classe que define e mantêm as regras de mapeamento, e `TFrmEditorAtributoSaidaRNA` que é classe de formulário que exibe a tela para edição do mapeamento pelo usuário da figura 27.

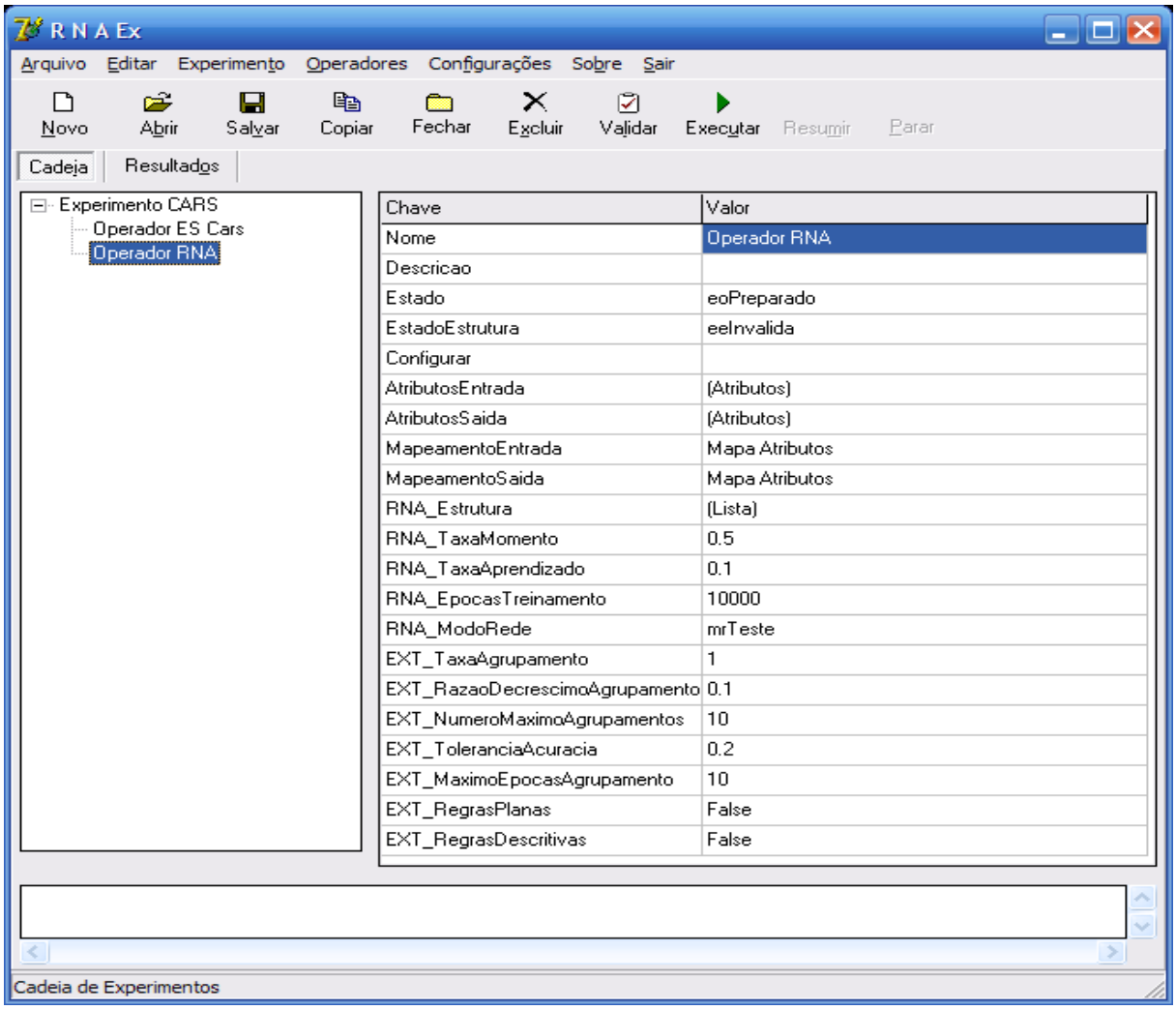


Figura 25 – Operador de rede neural artificial e seus parâmetros

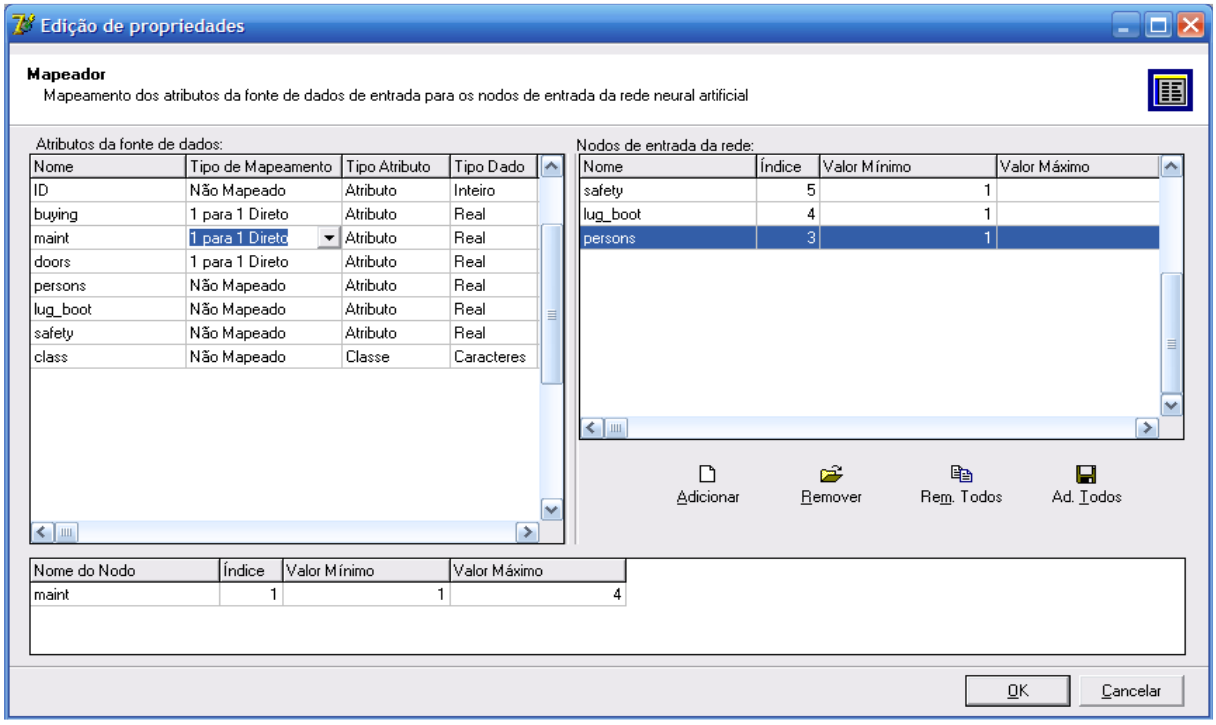


Figura 26 – Editor de mapeamento de atributos para neurônios de entrada da rede neural

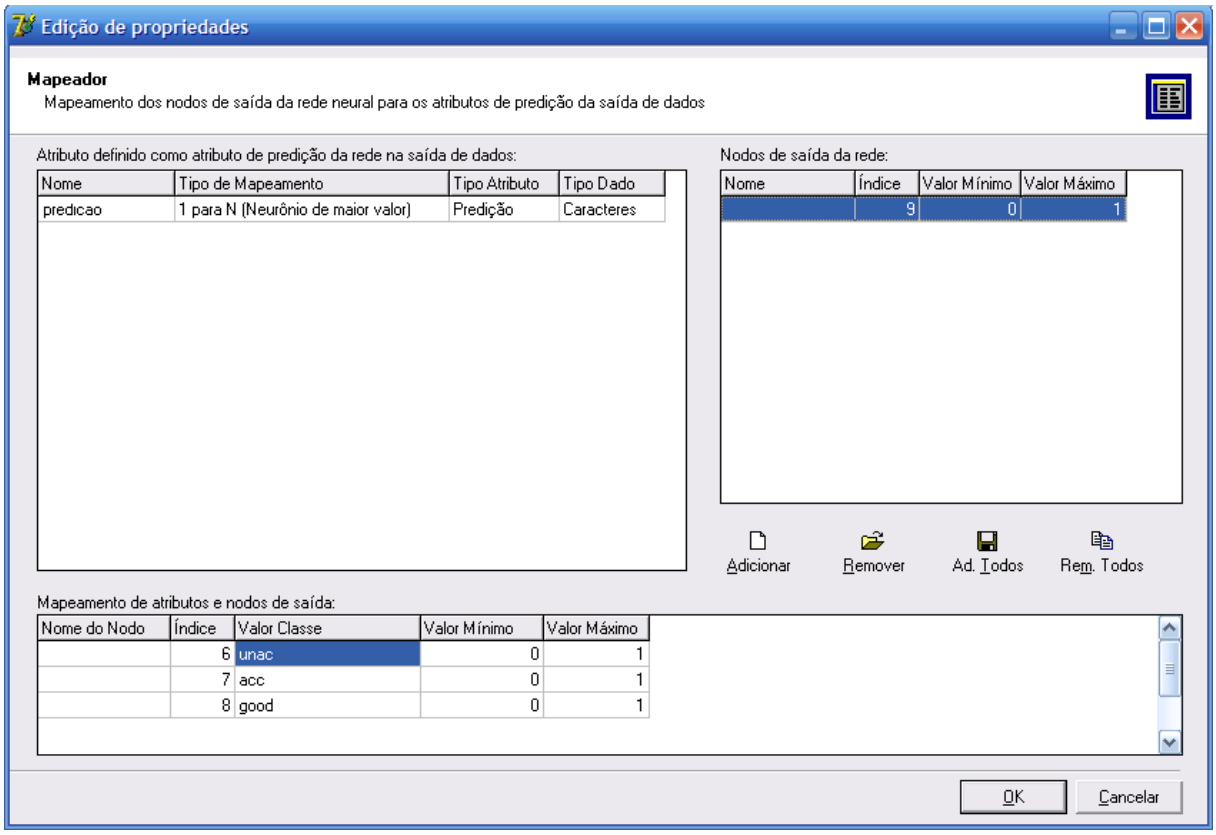


Figura 27 – Editor de mapeamento de neurônios de saída da rede neural para atributos

Os mapeadores são algumas das classes de maior complexidade do protótipo, devido à ligação entre diferentes classes e inúmeras listas internas de objetos e estruturas necessárias para seu funcionamento. A complexidade advém também do fato da possibilidade de diversas formas de ligação entre atributos e neurônios, visando flexibilidade e abstração de conversões entre os dados. No mapeamento de entrada, as ligações podem ser de quatro tipos:

a) o atributo pode não ser mapeado, ou seja não será considerado como entrada para a rede neural;

b) o atributo pode ser diretamente mapeado apenas sendo feito a conversão do valor do atributo para um valor numérico de entrada da rede. Sua designação é dita `1 para 1 diretamente`;

c) o atributo pode ser ligado a vários neurônios, onde cada valor distinto do neurônio pode ser mapeado para ativar um neurônio específico com valor específico (transformado) ou com o próprio valor do atributo (sem transformação). Esse mapeamento é dito `1 para N por valor`;

d) o atributo pode ser ligado a vários neurônios, e o valor do atributo é distribuído por faixas de valores e cada faixa é ligada a um neurônio. Estando o valor do atributo dentro de uma das faixas, o neurônio ligado a sua faixa recebe um valor de entrada especificado no mapeamento. Esse mapeamento é dito `1 para N por faixa`.

O quadro 21 mostra as implementações dos métodos de mapeamento de entrada para cada forma descrita, cada um gerido por classes específicas, que são as classes `TItemMapeadorEntradaDireto`, `TItemMapeadorEntradaPorFaixa` e `TItemMapeadorEntradaPorFaixa`, que possuem o método `GetValorSaida` que é sobrescrito da classe ascendente `TItemBaseMapeamentoEntrada`.

```
// mapeamento direto de valor
function TItemMapeadorEntradaDireto.GetValorSaida: TValorBaseNumerico;
Begin
  // tenta a conversão, porém se não conseguir será levantado erro implicitamente
  FValorSaida.ValorReal := FValorEntrada.ValorReal;
  Result := FValorSaida;
end;

// mapeamento por valor, com e sem transformação
function TItemMapeadorEntradaPorValor.GetValorSaida: TValorBaseNumerico;
Begin
  // sempre assumirá zero
  FValorSaida.ValorReal := 0;

  case  FTipoTransformacao of
    ttSemTransformacao:
      begin
        if  FValorOrigem is TValorBaseNumerico then
        begin
          if  (FValorEntrada.ValorReal = FValorAlvo.ValorReal) then
            FValorSaida.ValorReal := FValorAlvo.ValorReal
        end
        else
          if  (FValorEntrada.ValorNominal = FValorAlvo.ValorNominal) then
              FValorSaida.ValorReal := FValorAlvo.ValorReal;
      end;

    ttComTransformacao:
      begin
        if  FValorOrigem is TValorBaseNumerico then
        begin
          if  (FValorOrigem.ValorReal = FValorEntrada.ValorReal) then
            FValorSaida.ValorReal := FValorAlvo.ValorReal
        end
        else
          if  (FValorOrigem.ValorNominal = FValorEntrada.ValorNominal) then
              FValorSaida.ValorReal := FValorAlvo.ValorReal;
      end;
    end;

  Result := FValorSaida;
end;

// mapeamento por faixa de valores
function TItemMapeadorEntradaPorFaixa.GetValorSaida: TValorBaseNumerico;
Begin
  // sempre assumirá zero
  FValorSaida.ValorReal := 0;

  if  FValorFinal is TValorBaseNumerico then
  begin
    if  (FValorEntrada.ValorReal >= FValorInicial.ValorReal) and
        (FValorEntrada.ValorReal < FValorFinal.ValorReal) then
      FValorSaida.ValorReal := FValorAlvo.ValorReal
  end
  else
    if  (FValorEntrada.ValorNominal >= FValorInicial.ValorNominal) and
        (FValorEntrada.ValorNominal < FValorFinal.ValorNominal) then
        FValorSaida.ValorReal := FValorAlvo.ValorReal;

  Result := FValorSaida;
end;
```

Quadro 21 – Implementação dos métodos de mapeamento das classes de mapeamento de entrada

As classes descendentes de `TItemBaseMapeamentoEntrada` invocam o método `Mapear` para cada atributo no momento de execução do operador, quando a rede é alimentada com um registro do fluxo de dados. O método `Mapear` invoca o método `GetValorSaida`, que é executado conforme a classe que invocou o método `Mapear`. No quadro 22 é exibido o código fonte do método `Mapear`.

```
procedure TItemBaseMapeamentoEntrada.Mapear(aAtributo: TAtributo);
begin
  if   FValorEntrada is TValorBaseNumerico then
    FValorEntrada.ValorReal := aAtributo.ComoReal
  else
    FValorEntrada.ValorNominal := aAtributo.ComoCaracteres;

  FNodoAlvo.Valor := GetValorSaida.ValorReal;
end;
```

Quadro 22 – Implementação do método `Mapear` das classes base de mapeamento `TItemBaseMapeamentoEntrada`

No mapeamento de saída, as classes efetuam a ligação dos neurônios de saída com um único atributo de uma fonte de dados. Os mapeamentos de saída podem ser de dois tipos, descritos a seguir:

a) para cada neurônio de saída é atribuído um valor de rótulo de classe e o atributo destino recebe este valor do neurônio de saída que tiver maior valor dentre os demais neurônios de saída. Esse mapeamento é designado `1 para N por neurônio de maior valor` ou mapeamento de único valor;

b) para cada neurônio de saída é atribuído um valor de rótulo de classe e também uma faixa de valores. Se o valor de saída de um neurônio estiver dentro da faixa de valores atribuída a ele, o atributo recebe o valor do rótulo da classe. Este mapeamento é designado `1 para N por neurônio com faixa de valores`, ou mapeamento por faixas de valores.

No mapeamento de saída, as classes que representam o mapeamento de saída são as classes `TItemBaseMapeamentoSaida`, que é a classe base das classes de mapeamento, `TItemMapeadorSaidaUnico`, descendente de `TItemBaseMapeamentoSaida` e que gerencia o mapeamento de único valor, e por fim a classe `TItemMapeadorSaidaMultiplo`, que gerencia o mapeamento por faixas de valores.

As classes de itens de mapeamento de saída não efetuam propriamente o mapeamento, pois elas apenas armazenam o valor único ou de faixas e o valor de classe de saída. Isso é devido à necessidade de efetuar a varredura de todos os neurônios de saída para determinar o neurônio com maior valor ou cujo valor está dentro de uma faixa de valores.

Além disso, a necessidade de desempenho nesse ponto do processamento levou a construção de estruturas de matrizes de neurônios que armazenam os mapeadores e nodos, ao

invés de interagir em lista. Tais estruturas são montadas no momento do mapeamento e assim evitam a necessidade de efetuar iterações (laços) nas listas de itens de mapeamento. O quadro 23 exibe o método `Mapear` da classe `TMapaAtributosSaida`, que efetua o mapeamento de saída. Esta classe é responsável por armazenar as listas de itens de mapeamento e manipular as estruturas de matrizes, além de efetuar o mapeamento conforme as classes dos itens de mapeamento. No trecho de código fonte é possível observar também a computação do total de acertos e erros da rede, pois os mapeadores de saída conhecem o rótulo de classe real indicado no conjunto de exemplos.

```
procedure TMapaAtributosSaida.Mapear;
var
  i, j, k : Integer;
  xMaior : Extended;
  xValor : Variant;
begin
  if  FMapaSaida.AtributoSaida <> nil then
  begin
    j := Length(FMapaSaida.Itens);
    if  j > 0 then
    begin
      k := -1;
      if  FMapaSaida.Classe = TClasseItemMapeadorSaidaUnico
      begin // mapeamento único, ou seja um único valor, sem faixas de valor
        xMaior := -9999;
        for i := 0 to j - 1 do
        begin
          if  FMapaSaida.Itens[i].Nodo.Valor > xMaior then
          begin
            xMaior := FMapaSaida.Itens[i].Nodo.Valor;
            k := i;
          end;
        end;
      end
      else
        if  FMapaSaida.Classe = TClasseItemMapeadorSaidaUnico
        begin // mapeamento múltiplo, ou seja, uma faixa de valores
          i := 0;
          while (i < j) and (k = -1) do
          begin
            xMaior := FMapaSaida.Itens[i].Nodo.Valor;
            if  (xMaior >= FMapaSaida.Itens[i].Mapa.ValorInicial.ValorReal) and
                (xMaior <  FMapaSaida.Itens[i].Mapa.ValorFinal.ValorReal) then
               k := i
            else
              Inc(i);
          end;
        end;

      if  k > -1 then
      begin
        Inc(FTotalMapeado);

        if  FMapaSaida.AtributoSaida.TipoDado in [tdInteiro, tdReal, tdLogico] then
          xValor := FMapaSaida.Itens[k].Mapa.ValorClasse.ValorReal
        else
          xValor := FMapaSaida.Itens[k].Mapa.ValorClasse.ValorNominal;
        if  xValor = FMapaSaida.AtributoEntrada.Valor then
          Inc(FTotalAcertos)
        else
          Inc(FTotalErros);
      end;
    end;
  end;
end;
```

Quadro 23 – Implementação do método `Mapear` da classe `TMapaAtributosSaida`

A implementação do operador de rede neural artificial e extração de regras, descendente da classe `TBaseOperadorRNA`, é feita na classe `TExtratorNBA`. Tal classe utiliza uma classe que representa uma rede neural MLP com algoritmo de aprendizado *backpropagation*, denominada `TNeuralNetExtended`. Essa classe e suas antecessoras, `TNeuralNetBP` e `TNeuralNet`, são classes disponibilizadas livremente com código fonte.

A classe `TExtratorNBA` fornece os parâmetros e métodos para efetuar a configuração e a manipulação da rede neural, além dos parâmetros para o algoritmo de extração de regras RX e métodos para sua execução. Nos parágrafos seguintes são comentados os pontos de maior destaque nas duas implementações, primeiramente em relação à rede neural artificial e posteriormente sobre a implementação do algoritmo RX.

Dentro da implementação da classe de redes neurais, destacam-se nas classes originais a complexidade e modularidade das mesmas. Para o propósito do protótipo, algumas funcionalidades foram removidas, como por exemplo, a extensão da rede para redes recorrentes e os diversos critérios de parada do algoritmo de treinamento, e para incluir métodos que permitem obter ou alterar os valores de ativação dos neurônios artificiais dos neurônios ocultos, funcionalidade essencial para a aplicação do algoritmo RX.

No quadro 24 é exibido o método `Propagate` da classe base `TNeuralNetBP`, onde se visualiza a propagação dos sinais da rede através das camadas e a interceptação provida pelo método `DoOnComputeOutValuesLayer`, que é o método implementado especificamente para externar para o algoritmo de agrupamento os valores de entrada e de ativação dos neurônios de uma determinada camada, para proceder ao agrupamento e ajuste do valor de saída.

```
procedure TNeuralNetBP.Propagate;
var
  i, j, xIndex: integer;
  xArray: TVectorFloat;
begin
  for i := 1 to LayerCount - 1 do
  begin
    SetLength(xArray, LayersBP[i-1].NeuronCount);

    for xIndex := 0 to LayersBP[i-1].NeuronCount - 1 do
       xArray[xIndex] := LayersBP[i-1].NeuronsBP[xIndex].Output;

    for j := 0 to LayersBP[i].NeuronCount - 1 do
      LayersBP[i].NeuronsBP[j].ComputeOut(xArray);

    DoOnComputeOutValuesLayer(LayersBP[i].LayerType, LayersBP[i].NeuronArray);

    for xIndex := 0 to LayersBP[i-1].NeuronCount - 1 do
       xArray[xIndex] := 0;
  end;
  SetLength(xArray, 0);
  xArray := nil;
end;
```

Quadro 24 – Implementação do método `Propagate` da classe `TNeuralNetBP`

Os parâmetros específicos da rede neural artificial também foram simplificados e os

que necessitam ser informados pelo usuário são a taxa de aprendizado, a taxa de momento e o número de épocas de treinamento. Uma estrutura textual permite informar a estrutura da rede e também mantém as configurações da mesma, inclusive os pesos dos neurônios após a rede ser treinada.

Dentro do algoritmo RX, embutido na classe `TExtratorNBA`, destacam-se duas implementações importantes, que são o algoritmo de agrupamento de valores de ativação dos neurônios e o algoritmo de geração das regras baseados nesses valores de agrupamento.

O algoritmo de agrupamento do algoritmo RX foi desenvolvido conforme o algoritmo original. Uma classe exclusiva para a execução desse agrupamento, denominada `TGerenciadorAgrupamento`, foi construída. Esta classe solicita a execução da rede neural artificial com os dados de treinamento, para efetuar o agrupamento e testar a acurácia da rede com os valores agrupados dos neurônios.

O algoritmo de agrupamento necessita de alguns parâmetros externos, que são disponibilizados para informação pelo usuário. Tais parâmetros são a taxa inicial de agrupamento, a taxa de decréscimo de agrupamento, o número máximo de agrupamentos e a tolerância mínima de erro de erro da acurácia da rede.

O algoritmo de agrupamento executa a rede com todos os dados de treinamento, iniciando com o valor informado da taxa de agrupamento. Os valores de ativação dos neurônios são computados a cada passagem de um padrão de entrada na rede. Ao final do agrupamento, os mesmos padrões de entrada são passados novamente à rede e esta passa a operar utilizando os valores de ativação agrupados. São então computadas as saídas da rede e comparadas com as saídas esperadas, para avaliar a acurácia da rede. Após isso, a taxa de agrupamento é decrescida pelo valor da taxa de decréscimo de agrupamento e o processo é executado novamente com a nova taxa de agrupamento.

O processo de agrupamento com uma determinada taxa de agrupamento é chamado de época de agrupamento. A cada época de agrupamento, são computados a acurácia da rede com o agrupamento obtido, o número de agrupamentos gerados e número de combinações possíveis entre esses agrupamentos. Esta informação é armazenada com as informações do agrupamento. A cada época, o algoritmo de agrupamento verifica se atingiu um dos critérios de parada, que são a taxa de agrupamento chegar até zero ou o número de agrupamentos for maior que o permitido.

Ao atingir o critério de parada, o algoritmo avalia a melhor época de agrupamento, tendo como primeiro critério o menor número de combinações e como segundo critério a maior acurácia possível, dentre as que estão dentro da tolerância de erro informada pelo

usuário. Esta época e seus valores de agrupamento serão utilizados para gerar as regras de saída.

O quadro 25 mostra um trecho resumido do código do método `AgruparAtivacoes` da classe `TExtratorNBA` que efetua o processo total de agrupamento, com a manipulação de métodos da rede e dos métodos da classe do algoritmo de agrupamento. Destaca-se a atribuição dos métodos `AvaliarAtivacoes` e `AjustarAtivacoes` da classe `TGerenciadorAgrupamentos` para o método `OnComputeOutValuesLayer` de `TNeuralNetBP`. O método `AvaliarAtivacoes` efetua o agrupamento dos valores de ativação e o método `AjustarAtivacoes` ajusta o valor de ativação normal mapeando corretamente para o valor agrupado e alterando a saída do neurônio. Evidencia-se também a chamada aos métodos `CriarNovaEpoca` para geração das épocas de agrupamento e `ObterMelhorAgrupamento` para a busca do melhor agrupamento, do objeto da classe `TGerenciadorAgrupamentos`.

```
procedure TExtratorNBA.AgruparAtivacoes;
var
  xParar : Boolean;
  xAgUso : Integer;
begin
  // Cria nova época de agrupamento
  FAgrupador.CriarNovaEpoca;

  repeat
      // Executa uma época de agrupamento

      // passagem de todo o conjunto de treinamento para agrupar os valores de ativação
      FRede.OnComputeOutValuesLayer := FAgrupador.AvaliarAtivacoes;
      AvaliarRede;

      // Computa as estatísticas do vetor de agrupamento da época atual
      FAgrupador.CalcularEstatisticasEpoca(MapeamentoSaida.TaxaAcerto);

      // nova passagem de todo o conjunto de treinamento para avaliar a rede agora
      // com os novos valores de ativação agrupados
      FRede.OnComputeOutValuesLayer := FAgrupador.AjustarAtivacoes;
      AvaliarRede;

      // verifica a acurácia da rede e decide se descarta ou não o agrupamento
      FAgrupador.AtualizarEstatisticasEpoca(MapeamentoSaida.TaxaAcerto);

      // se atingiu os critérios de parada, termina o agrupamento
      xParar := (RoundTo(FAgrupador.TaxaAgrupamento,-13) <= 0) or
                (FAgrupador.TotalEpocas >= FMaxBuscaAgrupamentos);

      // Cria nova época de agrupamento
      if  (not xParar) then
          FAgrupador.CriarNovaEpoca;

      FluxoInternoSaida.ExcluirTodos;
  until xParar;

  // procura o melhor agrupamento e retorna o mesmo
  xAgUso := FAgrupador.ObterMelhorAgrupamento;
end;
```

Quadro 25 – Implementação do método `AgruparAtivacoes` da classe `TExtratorNBA`

A implementação do método `AvaliarAtivacoes` da classe `TGerenciadorAgrupamento`, método responsável por efetuar os agrupamentos dos valores de ativação de uma época segundo o método original do algoritmo RX, é exibida no quadro 26.

```
procedure TGerenciadorAgrupamento.AvaliarAtivacoes(aLayerType : TLayerType;
  const aNeuroniosRede: TNeuronArray);
var
  xTaxaEpoca : Double;

  procedure AvaliarAtivacao(var aNeuroAgp: TNeuronioAgrupamento; const aNeuroRede: TNeuron);
  var
    i, j, xInd : Integer;
    xMin : Double;
  begin
    xMin := 2;
    xInd := -1;
    i := High(aNeuroAgp.Clusters);

    if  i <> -1 then
    begin
      for j := 0 to i do
      begin
        if  Abs(aNeuroRede.Output - aNeuroAgp.Clusters[j].Ativacao) < xMin then
        begin
          xMin := Abs(aNeuroRede.Output - aNeuroAgp.Clusters[j].Ativacao);
          xInd := j;
        end;
      end;
    end;

    if  (xInd <> -1) and
        (Abs(aNeuroRede.Output - aNeuroAgp.Clusters[xInd].Ativacao) <= xTaxaEpoca) then
    begin
      aNeuroAgp.Clusters[xInd].Soma := aNeuroAgp.Clusters[xInd].Soma + aNeuroRede.Output;
      aNeuroAgp.Clusters[xInd].Numero := aNeuroAgp.Clusters[xInd].Numero + 1;
    end
    else
    begin
      xInd := Length(aNeuroAgp.Clusters) + 1;
      SetLength(aNeuroAgp.Clusters, xInd);
      aNeuroAgp.Indice := aNeuroRede.Index;
      aNeuroAgp.Clusters[xInd -1].Ativacao := aNeuroRede.Output;
      aNeuroAgp.Clusters[xInd -1].Soma := aNeuroRede.Output;
      aNeuroAgp.Clusters[xInd -1].Numero := 1;
    end;
  end;

var
  i, j, k : Integer;
  xAchou : Boolean;
begin
  if  aLayerType = ltHidden then
  begin
    xTaxaEpoca := FTaxaAgrupamento;

    k := High(aNeuroniosRede);
    for i := 0 to High(FEpocas[FEpocaAtual].Neuronios) do
    begin
      j := 0;
      xAchou := False;
      while (j <= k) and (not xAchou) do
      begin
        xAchou := (FEpocas[FEpocaAtual].Neuronios[i].Indice = aNeuroniosRede[j].Index);
        if  xAchou then
           AvaliarAtivacao(FEpocas[FEpocaAtual].Neuronios[i], aNeuroniosRede[j])
        else
          Inc(j);
      end;
    end;
  end;
end;
```

Quadro 26 – Implementação do método `AvaliarAtivacoes` da classe `TGerenciadorAgrupamento`

Após o processo de agrupamento das ativações e a escolha da melhor época, é feita a combinação dos valores de ativação dos neurônios. A classe `TCombinaçoesAtivacao` é

responsável por criar as listas de combinações, criando para cada combinação uma instância de `TCombinacaoAtivacao`. Efetuado esse processo, é feita a busca das classes que cada combinação de ativações gera, efetuando para isso a execução da rede apenas a partir da camada oculta com os valores de ativação para cada neurônio. O valor da classe resultante de uma combinação também é armazenada pela instância de `TCombinacaoAtivacao` que mantêm a combinação respectiva.

Em seqüência, executada a rede com todos os dados do conjunto de treinamento. A cada padrão de entrada, é verificado quais valores de ativações agrupados dos neurônios de camadas ocultas são gerados pelo padrão de entrada. Os valores de ativação agrupados são analisados e verificados se encaixam em alguma das combinações existentes. Se isso ocorrer, os dados do padrão de entrada são também armazenados no na instância de `TCombinacaoAtivacao` que mantêm a combinação.

Após a passagem de todo o conjunto dos padrões de entrada, cada combinação de valores de ativação agrupados possuem os valores de entrada que levam a esses valores de ativação. A partir desse momento, são geradas as regras para cada combinação de valores conforme esses dados de entrada, para a respectiva classe. Pode ocorrer nesse processo que algumas combinações não sejam ativadas, o que é normal.

O método de inferência das regras com base nos valores de entrada que levam aos valores de ativação agrupados é exibido no quadro 27. O método `GerarRegras` de `TCombinacaoAtivacao` escolhe o neurônio com o atributo de maior freqüência e utiliza este neurônio e o valor mais freqüente como condição inicial de uma regra, armazenando essa condição no topo da árvore de condições (armazenado em um nodo da classe `TFolhaArvore`).

O conjunto de dados de entrada é ordenado pelo neurônio com o atributo mais freqüente, e então criado um novo conjunto de dados removendo do conjunto original o neurônio que possui o valor mais freqüente. Esse novo conjunto de dados então passa novamente pelo processo, gerando assim o segundo nodo de condições da árvore, e assim sucessivamente até não existirem mais dados para gerar um novo conjunto de dados.

Uma árvore de regras é criada para cada combinação de ativação e nessa árvore uma condição pode ser origem para outras condições e assim sucessivamente. Assim, são criados ramos dentro da árvore, de forma a minimizar o número de condições de uma regra quando uma condição se repete no conjunto de entrada da combinação. As regras são extraídas da árvore de forma textual no formato `se condição`$_1$ `e condição`$_2$ `e...e condição`$_N$ `então conseqüente` se o parâmetro `EXT_RegrasPlanas` for verdadeiro. Ou podem ser extraídas no próprio formato de árvore onde o atributo que de maior freqüência é colocado no topo da

árvore e as demais condições são aninhadas abaixo dele, sendo que as condições no mesmo nível são condições "OU" entre elas. As regras também podem ser transferidas para uma instância da classe `TAvaliadorRegra` e testadas com um conjunto de dados de entrada, sendo que nessa situação são testadas no formato plano.

```
function InferirRegras(aCnjDados : TConjuntoDadosNeuronioArray;
  aFolhaArvore : TFolhaArvore) : TFolhaArvore;
var
  xTmpCnj : TConjuntoDadosNeuronioArray;
  xTmpCdn : TConjuntoDadosNeuronio;
  i, xMaiorFrequencia : Integer;
begin
  Result := nil;
  while (Length(aCnjDados) > 0) and (aCnjDados[0].Tamanho > 0) do
  begin
    if  Length(aCnjDados) = 1 then // só tem uma coluna
    begin
      if  High(aCnjDados[0].ValoresDistintos) > -1 then
      begin
        xTmpCdn := aCnjDados[0];
        for i := 0 to High(xTmpCdn.ValoresDistintos) do
          Result := aFolhaArvore.AdicionarNodoFilho(
                                                xTmpCdn.ValoresDistintos[i].Valor,
                                                xTmpCdn.NeuronioOrigem,
                                                xTmpCdn.ValoresDistintos[i].ValorOriginal,
                                                xTmpCdn.NomeNeuronioOrigem);
          while xTmpCdn.Tamanho > 0 do
            xTmpCdn.RemoverValor(0);
        end;
      end
      else
      begin
        for i := 0 to High(aCnjDados) do // buscar o mais freqüênte
          aCnjDados[i].CalcularFrequencias;

        xMaiorFrequencia := 0;
        for i := 0 to High(aCnjDados) do
          if  aCnjDados[i].ValorMaisFrequente.Frequencia >
              aCnjDados[xMaiorFrequencia].ValorMaisFrequente.Frequencia then
            xMaiorFrequencia := i;

        OrdenarPorFrequencia(aCnjDados, xMaiorFrequencia);
        xTmpCdn := aCnjDados[xMaiorFrequencia];
        Result := aFolhaArvore.AdicionarNodoFilho(xTmpCdn.ValorMaisFrequente.Valor,
                                                xTmpCdn.NeuronioOrigem,
                                                xTmpCdn.ValorMaisFrequente.ValorOriginal,
                                                xTmpCdn.NomeNeuronioOrigem);
        try
          SetLength(xTmpCnj, 0);
          CopiarConjuntoDados(aCnjDados, xTmpCnj,
                              aCnjDados[xMaiorFrequencia].NeuronioOrigem,
                              aCnjDados[xMaiorFrequencia].ValorMaisFrequente.Valor);
          InferirRegras(xTmpCnj, Result);
        finally
          FinalizarArrayNeuronios(xTmpCnj);
        end;
      end;
    end;
  end;
end;
```

Quadro 27 – Implementação do método `InferirRegras` da classe `TCombinacaoAtivacao`

A avaliação das regras para verificar sua acurácia é feita pela passagem de um conjunto de dados de entrada (de teste ou de treinamento) estando o operador da rede neural no modo de teste de regras. A rede é executada passando os valores do conjunto de entrada aos neurônios de entrada e estes são passados para o avaliador das regras. O valor de cada

neurônio de entrada é comparado com o valor do neurônio na condição da regra. Se todos os valores de entrada coincidem com as condições da regra e também o conseqüente do conjunto de dados coincide com o conseqüente da regra, são incrementados o número de execuções e o número de acertos da regra.

Se a regra for disparada e o conseqüente da regra não é o mesmo do conjunto de entrada, o número de execuções e o número de disparos inválidos são incrementados em uma unidade. Cada padrão de entrada é testado contra todas as regras e ao final é feito a computação da estatística global de acertos, execuções e disparos inválidos, bem como a acurácia de todas as regras. No quadro 28 é demonstrado o método `Testar` da classe `TRegra`, responsável pelo teste de uma regra e computação das estatísticas de disparos inválidos, execuções e acertos de uma das regras da árvore de regras.

```
function TRegra.Testar(const aArrayTeste : array of Double; const aConsequente : String) :
Boolean;.

  function Testar : Boolean;
  var
    i, j  : Integer;
    xAchou : Boolean;
  begin
    i := 0;
    Result := True;
    // procura o neurônio correspondente dentro do array
    while (i <= FTopoCondicoes) and (Result) do
    begin
      j := 0;
      xAchou := False;
      while (j <= FTopoCondicoes) and (not xAchou) do
      begin
        xAchou := i = FCondicoes[j].Neuronio;
        if  (not xAchou) then
          Inc(j);
      end;
      Result := xAchou;
      if  Result then
        Result := FCondicoes[j].ValorNodo = aArrayTeste[i];

      Inc(i);
    end;
  end;

begin
  Result := False;
  // se o array possui o mesmo tamanho do conjunto de condições da regra
  if  High(aArrayTeste) = FTopoCondicoes then
  begin
    Result := Testar;
    if  Result then // a regra foi disparada
    begin
      // acertou, então incrementa o número de acertos
      if  aConsequente = FConsequente then
        Inc(FNumeroAcertos)
      else
       // errou, então incrementa o número de disparos inválidos
        Inc(FDisparosInvalidos);

      Inc(FNumeroExecucoes);

      FAcuraciaExecucoes := FNumeroAcertos/FNumeroExecucoes * 100;
    end;
  end;
end;
```

Quadro 28 – Implementação do método `Testar` da classe `TRegra`

A execução do experimento, com a execução seqüencial dos operadores, é gerida pela classe `TCadeiaOperadores`, a principal classe gerenciadora do protótipo. Essa classe manipula a adição e remoção dos operadores e gerenciadores de operadores e controla a criação e remoção dos fluxos de entrada e saída dos operadores. A função de execução seqüencial dos operadores do experimento é feita através da chamada de métodos conhecidos de validação e execução dos gerenciadores dos operadores adicionados na lista de operadores da cadeia, que por sua vez chamam os métodos específicos dos operadores. Ou seja, a cadeia conhece os métodos e atributos dos gerenciadores que conhecem os métodos e atributos dos operadores.

Antes da execução de um operador a cadeia questiona o mesmo para que este informe que está corretamente configurado e pronto para execução, com todas as estruturas necessárias completas. As classes base de operadores já possuem o método de validação básico, mas os operadores descendentes dessas classes devem sobrescrever esses métodos caso possuam mais configurações ou estruturas a validar. Estando o operador validado, o mesmo pode ser executado.

O quadro 29 demonstra um trecho resumido do código fonte do método `Executar` da classe `TCadeiaOperadores`. Nele estão evidenciadas os controles durante a execução seqüencial dos gerenciadores de operadores na lista da cadeia de operadores, e conseqüentemente dos operadores vinculados aos gerenciadores. Na execução, um gerenciador de operador (`TGerenciadorOperador`) é extraído da lista de gerenciadores e em seguida é chamado o método `Validar`. O método `Validar` atua sobre o atributo `Estado` do gerenciador de operador. Em seguida esse estado é verificado e é gerada exceção caso não esteja indicado que o operador está válido.

Estando válido o operador, é chamado o método `Executar` do gerenciador deste operador, que chama respectivamente o método `Executar` do operador, que deve proceder ao seu objetivo. Assim, o fluxo de dados de saída de um operador executado deve ser alimentado com os resultados de sua execução, e o próximo operador a ser executado irá utilizar desse fluxo como fluxo de entrada. Esse processo deve ocorrer sucessivamente, item por item da lista de gerenciadores de operadores até o fim da lista ou até ocorrer alguma exceção.

No código fonte também existem trechos responsáveis pela notificação dos eventos que ocorrem na execução de um operador. Isso possibilita que os passos da execução do experimento sejam visualizados e as informações acerca desses passos sejam apresentadas na caixa de texto do protótipo. Os dados de saída de um operador também podem ser visualizados na guia `Resultados`, antes de ser executado o próximo operador. O controle de

execução possui atributos internos que permitem que se retorne o fluxo de execução após uma parada para visualização dos dados, executando então o próximo operador na fila.

```
procedure TCadeiaOperadores.Executar;
var
  xGOperador : TGerenciadorOperador;
  xContinuar, xFimLista : Boolean;
  i, j : Integer;
begin
  if  (FEstadoExecucao <> eeExecutando) then
  begin
    if (FOperadores.NumeroOperadores > 0) then
    begin
      xContinuar := True;

      if  FEstadoExecucao = eeAguardando then // ao resumir o operador atual já foi
        i := FOperadorAtual + 1               // executado, passando então ao próximo
      else
        i := 0;
      if  (i >= 0) then
      begin
        j := FOperadores.NumeroOperadores;
        xFimLista := (i >= j);
        xGOperador := nil;
        try
          while (xContinuar) and (not xFimLista) do // laço de execução
          begin
            FOperadorAtual := i;                    // armazena o estado e operador atual
            FEstadoExecucao := eeExecutando;

            // obtêm operador, revalida e executa. Se não validou, gera exceção
            xGOperador := FOperadores[i];
            xGOperador.Validar;
            if  (not (xGOperador.Estado in [eoPreparado, eoConfigurado])) then
            begin
              FEstadoExecucao := eeParado;
              raise Exception.Create(cOPEINVALIDO);
            end;

            GerarRegistro(trInformacao,Format(cENTEXECUCAO,[xGOperador.Nome]));
            xGOperador.Executar;
            GerarRegistro(trInformacao,Format(cSAIEXECUCAO,[xGOperador.Nome]));

            if  eppApos in xGOperador.PontoParada then //tem ponto de parada, muda o estado
            begin
              xContinuar := False;
              FEstadoExecucao := eeAguardando;
              GerarRegistro(trInformacao,Format(cPDPENOPERAD,[xGOperador.Nome]));
            end
            else
            begin
              Inc(i);
              xFimLista := (i >= j);
            end;
          end;
        except
          on E:Exception do
          begin
            FOperadorAtual := -1;
            FEstadoExecucao := eeParado;
            ErroExecucao(xGOperador, E.Message)
          end;
        end;
        if  xFimLista then
        begin
          FEstadoExecucao := eeParado;
          GerarRegistro(trInformacao,'Execução completa');
        end;
      end
      else ErroExecucao(nil, cPDPENOPOSIC ') // erro não previsto na lógica interna
    end
    else ErroExecucao(nil,'Não há operadores na cadeia do experimento');
  end
end;
```

Quadro 29 – Implementação do método `Executar` da classe `TCadeiaOperadores`

3.4.3 Operacionalidade da implementação

A operacionalidade da implementação será demonstrada com um estudo de caso comum na literatura de aprendizado de máquina e específico para a tarefa de classificação em *data mining*. O estudo de caso possui um conjunto de dados que será aplicado no protótipo, demonstrando a utilização do mesmo desde a carga de dados, passando pela configuração e treinamento da rede neural artificial até a efetiva extração das regras da rede e avaliação das mesmas.

O estudo de caso foi obtido do repositório *UCI Machine Learning Repository*, onde são mantidos conjuntos de dados específicos e de uso livre para testes em aplicações de aprendizado de máquina e *data mining* (ASUNCION e NEWMANN, 2007). O conjunto de dados do estudo de caso é composto por dados acerca do grau de aceitação de consumidores em relação a modelos de veículos, conforme algumas características desses modelos. Essas características são dadas por seis atributos nominais e seus possíveis valores, sendo que existem quatro classes possíveis de aceitação. As características e seus valores possíveis, os rótulos de classes e suas distribuições no conjunto de dados são demonstradas no quadro 30.

O conjunto de dados é completo, ou seja, possui todas as combinações possíveis de atributos, sem ruído e sem omissão de atributos ou valores nos registros. O número de registros de dados é de 1727, onde cada registro é acompanhado da respectiva classe.

```
Configuração do conjunto de dados "CARS"

Informação dos atributos do modelo do veículo
- preço: muito alto, alto, médio e baixo
- valor de manutenção: muito alto, alto, médio e baixo
- número de portas: 2, 3, 4, 5 ou mais
- capacidade de pessoas: 2, 4, 5 ou mais
- capacidade do porta malas: pequena, média e grande
- nível de segurança: baixo, médio e alto

Informação do nível de aceitação (classes) possíveis, em número de quatro:

Aceitação:
- não aceitável
- aceitável
- boa
- muito boa

Distribuição das classes
- não aceitável: 1210 (70,06%)
- aceitável: 384 (22,23%)
- boa: 69 (3,93%)
- muito boa: 65 (3,76%)
```

Quadro 30 – Descrição dos atributos, classes e sua distribuição no conjunto de dados do estudo de caso

O objetivo do estudo de caso é construir e treinar uma rede neural artificial para classificar cada registro na sua classe respectiva e então extrair as regras da rede neural treinada, avaliando posteriormente as mesmas utilizando todo o conjunto de dados.

O primeiro passo foi a transformação, através da ferramenta Yale, dos atributos nominais em atributos numéricos, para serem utilizados na rede neural artificial, gerando um novo conjunto de dados que foi armazenado em uma tabela utilizando o gerenciador de bancos de dados MS SqlServer Express. Em seguida foi gerado um segundo conjunto de dados para treinamento, composto de 267 registros (15,46 por cento da população), mantendo proporcionalmente a mesma distribuição de classes do conjunto completo.

O segundo passo é a configuração do experimento no protótipo. Para isso, foi criado um novo experimento e adicionados neste o operador de busca de dados e o operador de manipulação de rede neurais e extração de regras com o algoritmo RX. O operador de busca de dados foi configurado para obter os dados da tabela na base de dados e os atributos foram configurados, definindo seus tipos de dados, como pode ser observado na figura 28.

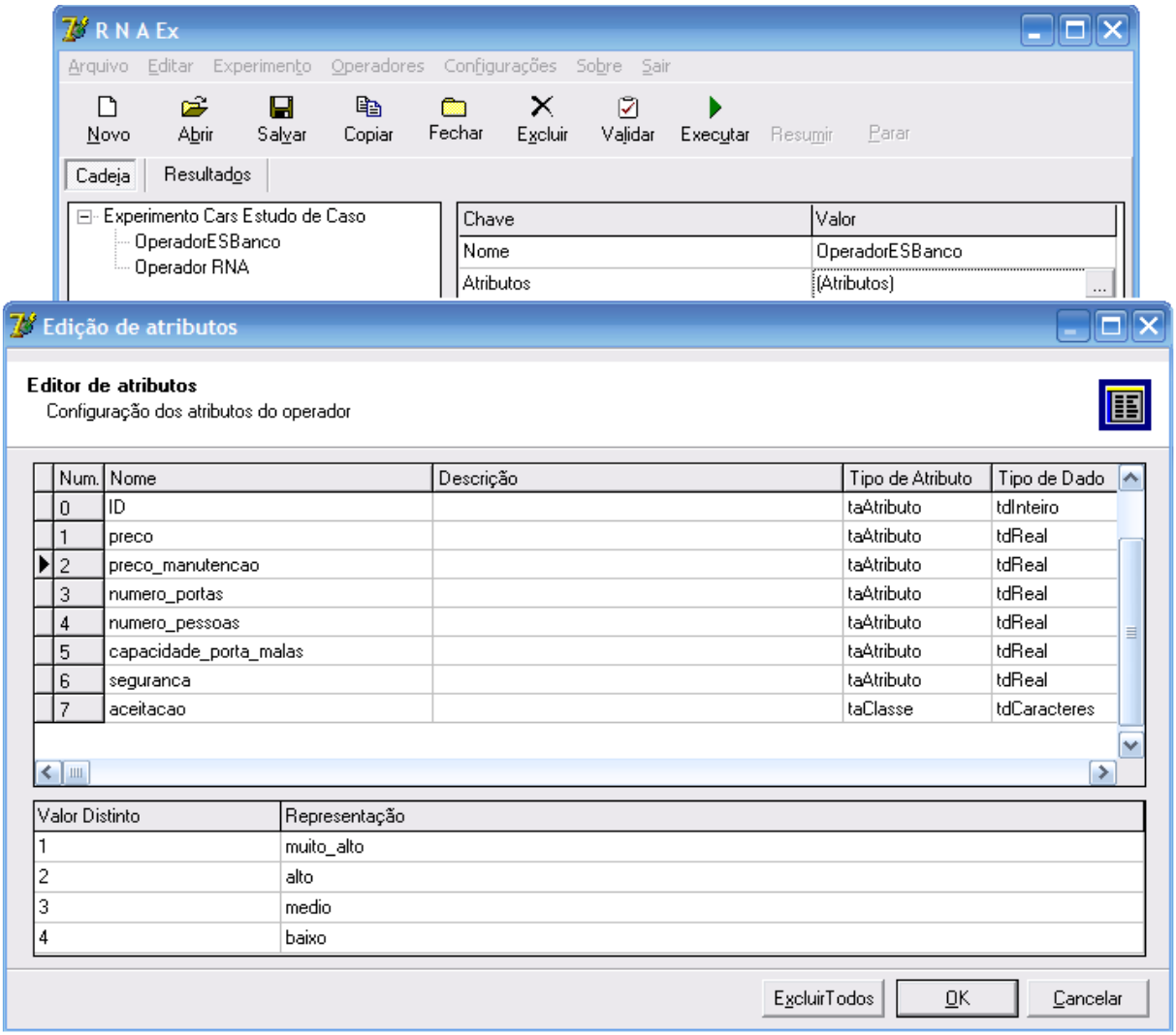


Figura 28 – Configuração do operador de entrada de dados e seus atributos para o estudo de caso

Após a configuração do operador de carga de dados, o operador da rede neural artificial foi configurado para construir a rede neural. A mesma foi criada com seis neurônios na camada de entrada, cinco neurônios na camada intermediária e quatro neurônios na camada de saída e então foram ajustados os parâmetros de treinamento. O mapeamento dos atributos de entrada para os neurônios de entrada foi efetuado e podem ser vistos na figura 29. Todos os mapeamentos são diretos, ou seja, o valor de entrada do atributo é diretamente atribuído ao neurônio de entrada ao qual está ligado.

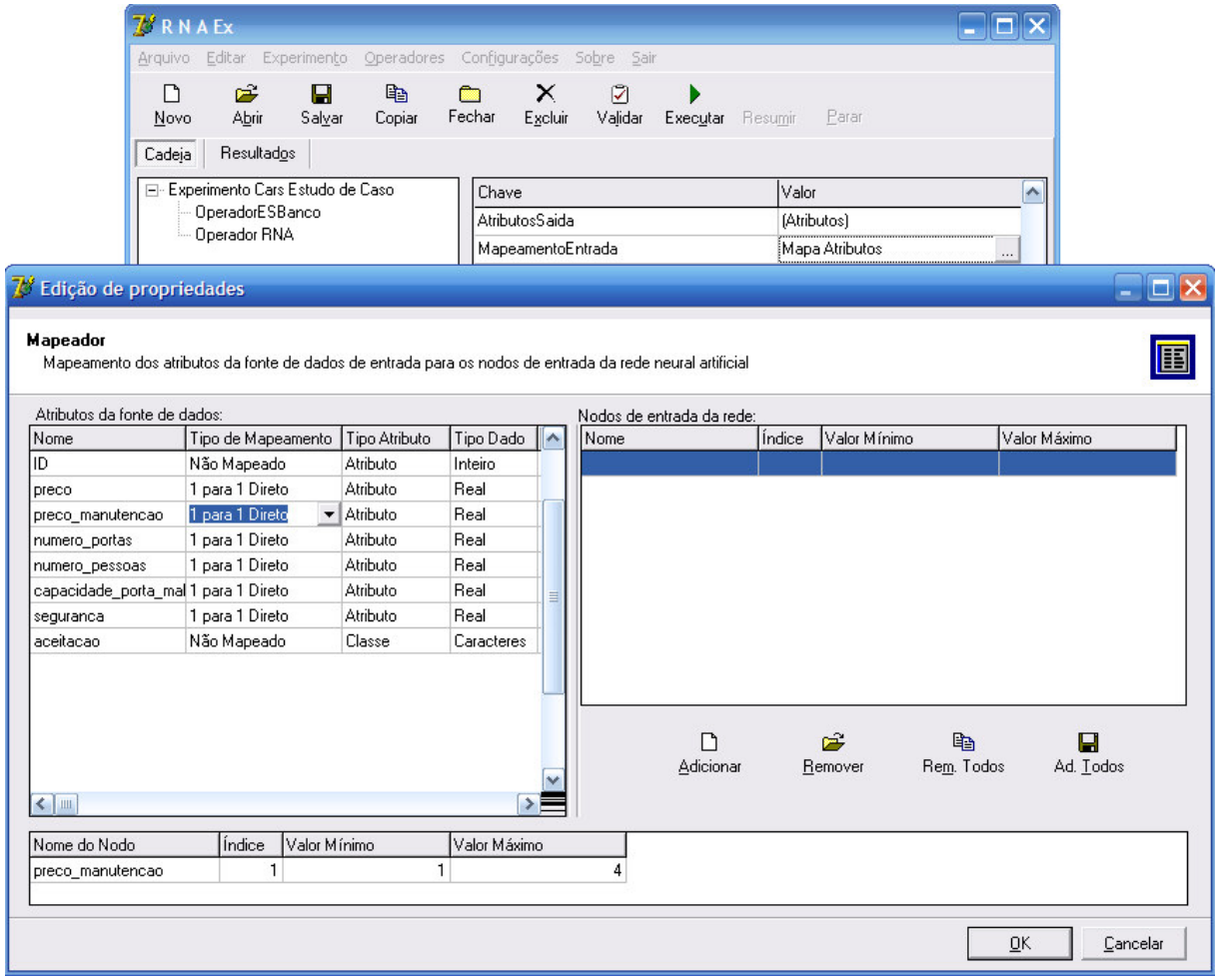


Figura 29 – Configuração do mapeamento de entrada do operador de rede neural artificial

No próximo passo, foi efetuada a configuração do mapeamento de saída, mapeando os neurônios de saída da rede para o atributo do fluxo de dados de saída. Na figura 30 é visualizada a configuração efetuada. Os quatro neurônios de saída, cada um representando uma das classes do problema, foram mapeados para um único atributo, utilizando a opção `1 para N por neurônio de maior valor` ou mapeamento de único valor.

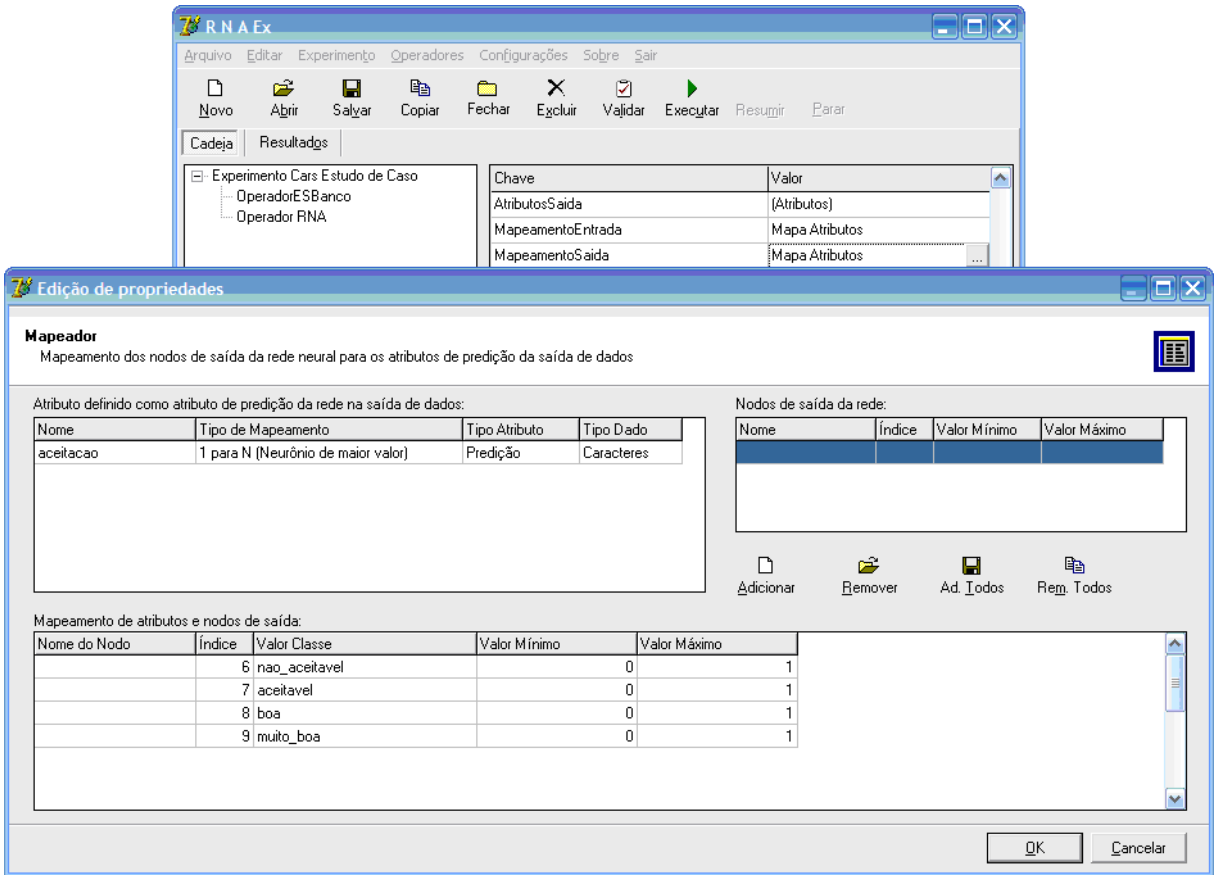


Figura 30 – Configuração do mapeamento de saída do operador de rede neural artificial

Como próximo passo, foram ajustados os parâmetros de treinamento da rede neural artificial, como a taxa de aprendizado, a taxa de momento e o número de épocas de treinamento. Também já foram definidos os parâmetros para a fase de extração de regras, como a taxa de agrupamento, a taxa de decréscimo da taxa de agrupamento, o número máximo de agrupamentos, a tolerância máxima de erro em relação a acurácia da rede e o número máximo de épocas de agrupamento. A tela com a configuração de todos esses parâmetros pode ser observada na figura 31.

RNA_Estrutura	(Lista)
RNA_TaxaMomento	0,5
RNA_TaxaAprendizado	0,1
RNA_EpocasTreinamento	1000
RNA_ModoRede	mrTreinamento
EXT_TaxaAgrupamento	1
EXT_RazaoDecrescimoAgrupamento	0,1
EXT_NumeroMaximoAgrupamentos	10
EXT_ToleranciaAcuracia	0,1
EXT_MaximoEpocasAgrupamento	10
EXT_RegrasPlanas	False
EXT_RegrasDescritivas	False

Figura 31 – Parâmetros para o treinamento e para a extração de regras da rede neural artificial

Após a configuração de todos os parâmetros, a rede foi instruída a entrar no modo de treinamento, e os resultados do treinamento são visíveis na figura 32. Após o treinamento, a rede foi instruída a entrar no modo de teste. Os testes foram efetuados tanto para o conjunto de treinamento como também para o conjunto completo de registros. A acurácia da rede atingiu 98,50 por cento no caso do conjunto de testes de treinamento, conforme demonstrado na figura 33. Para o teste com todos os registros do conjunto de dados, a acurácia foi de 94,67 por cento, conforme demonstrado na figura 34.

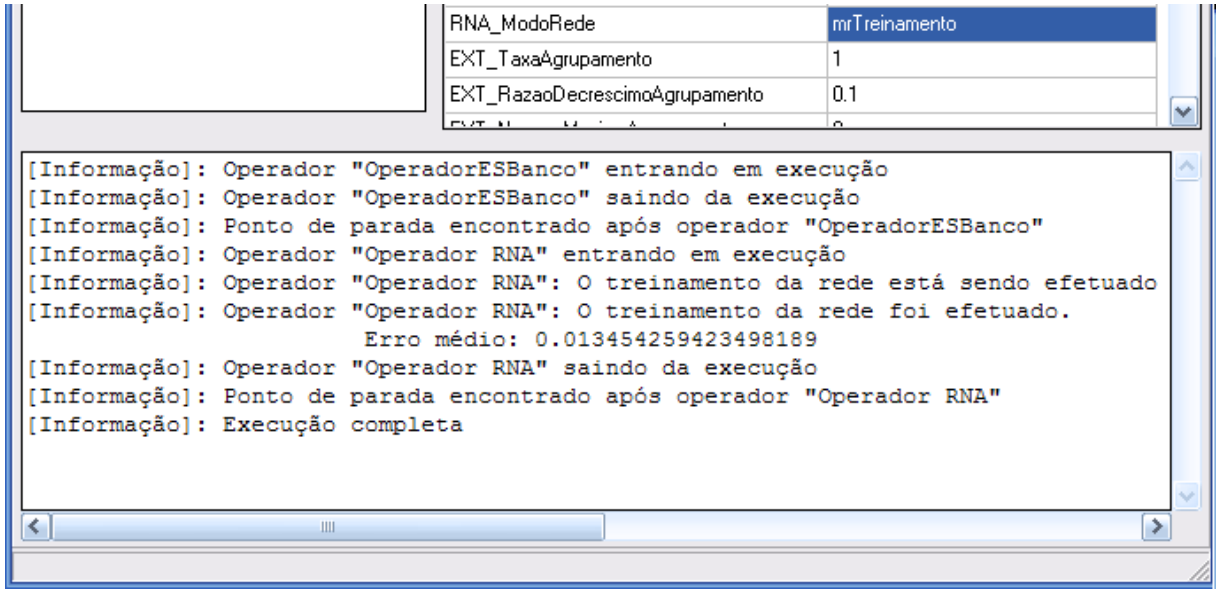


Figura 32 – Resultados do treinamento da rede para os dados de treinamento do estudo de caso

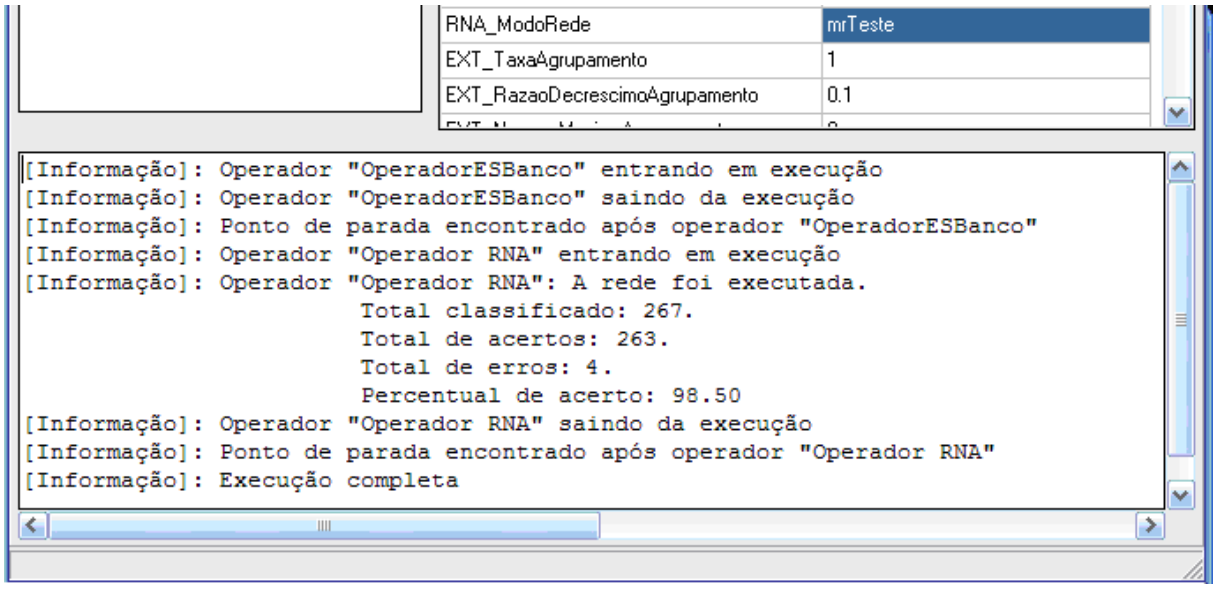


Figura 33 – Resultados do teste da rede para o conjunto de dados de treinamento do estudo de caso

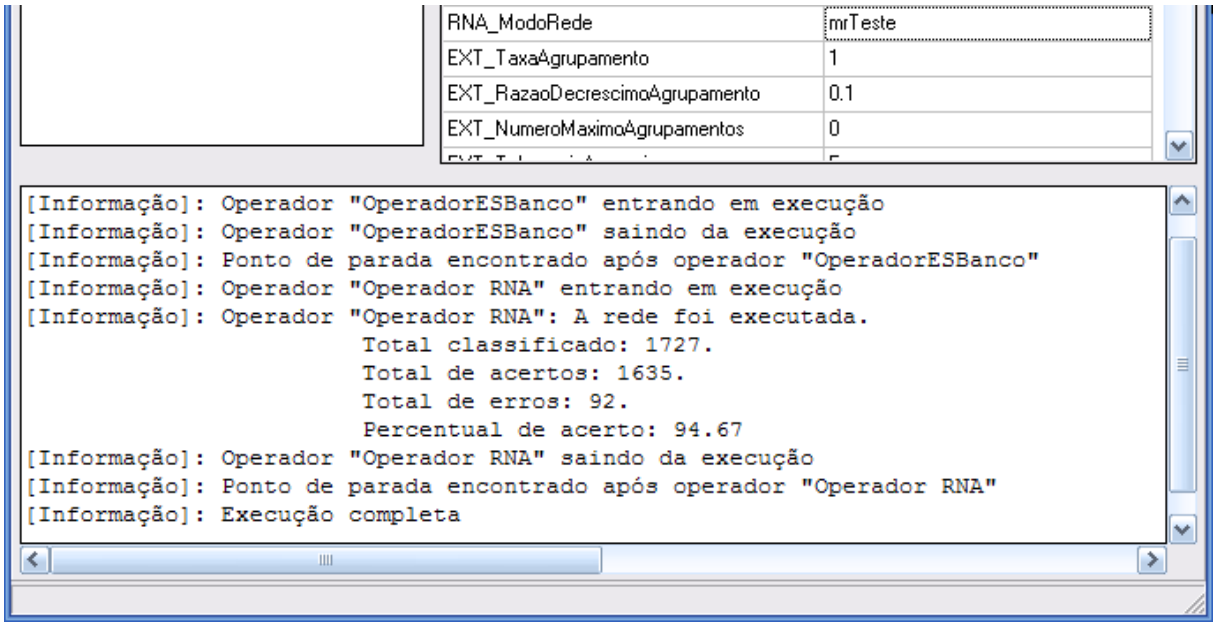


Figura 34 – Resultados do teste da rede para o conjunto de dados completo do estudo de caso

Após o treinamento e teste da rede, e com o resultado de acurácia considerado suficientemente bom para o prosseguimento do experimento, a rede foi instruída a entrar no modo de extração de regras. Dessa forma, foi executado o algoritmo de agrupamento do algoritmo RX, que gerou as épocas de agrupamento dos neurônios e efetuou a escolha automática da melhor época. Os resultados do algoritmo de agrupamento e a melhor época escolhida estão exibidos na figura 35.

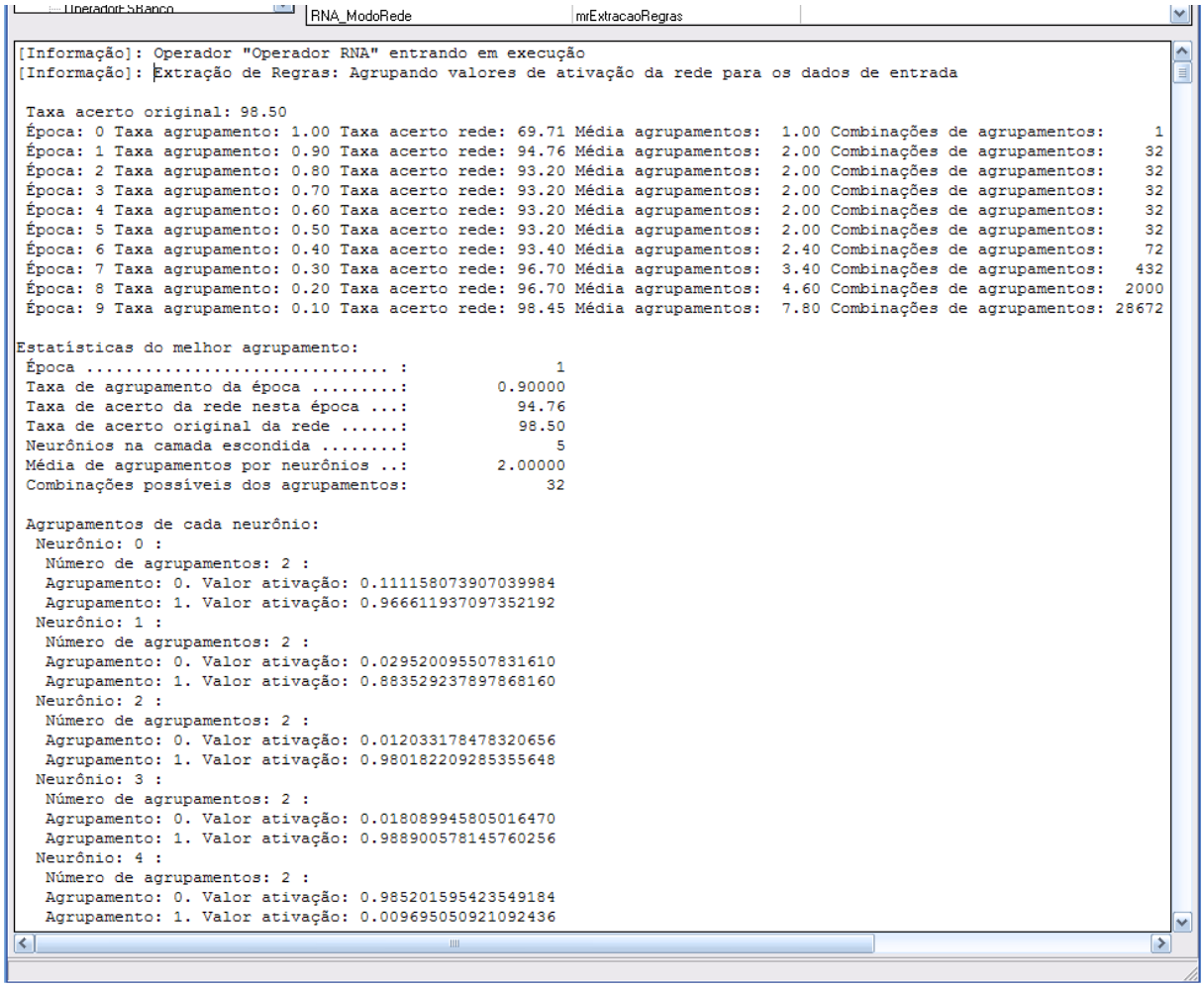


Figura 35 – Resultados do algoritmo de agrupamento de dados do algoritmo RX

Ainda no mesmo passo, o operador efetua a geração das combinações dos valores de ativação da época escolhida, a busca das classes para as combinações geradas, a passagem e armazenamento dos dados de entrada que geram os valores agrupados em cada combinação e a efetiva extração de regras.

O número de regras geradas para o estudo de caso no formato em árvore é em número de dez, sendo que no quadro 31 estão exibidos dois exemplos das regras geradas nesse formato. As demais regras desse conjunto estão no apêndice A deste trabalho, no quadro 35. No quadro 32 são exibidas parte das regras extraídas no formato plano, ou seja, apenas condições E, sem hierarquia de condições, o que causa aumento do número de regras. Ao total foram geradas 267 regas, e tal conjunto completo de regras está exibido no apêndice B deste trabalho, no quadro 36.

```
Regra: 1

SE preco = muito_alto
   SE preco_manutencao = muito_alto
      SE numero_pessoas = 2
         SE seguranca = alto
            SE numero_portas = 4
               SE capacidade_porta_malas = pequena
               SE capacidade_porta_malas = media
            SE numero_portas = 2
               SE capacidade_porta_malas = media
               SE capacidade_porta_malas = grande
            SE numero_portas = 5_ou_mais
               SE capacidade_porta_malas = pequena
         SE seguranca = médio
            SE numero_portas = 4
               SE capacidade_porta_malas = pequena
               SE capacidade_porta_malas = grande
            SE capacidade_porta_malas = media
               SE numero_portas = 2
               SE numero_portas = 3
      SE numero_pessoas = 4
         SE capacidade_porta_malas = media
            SE seguranca = médio
               SE numero_portas = 3
               SE numero_portas = 4
               SE numero_portas = 5_ou_mais
            SE seguranca = baixo
               SE numero_portas = 2
               SE numero_portas = 4
         SE capacidade_porta_malas = grande
            SE seguranca = baixo
               SE numero_portas = 2
               SE numero_portas = 3
   SE preco_manutencao = alto
      SE numero_pessoas = 2
         SE numero_portas = 2
            SE capacidade_porta_malas = media
               SE seguranca = médio
               SE seguranca = alto
         SE numero_portas = 5_ou_mais
            SE capacidade_porta_malas = pequena
               SE seguranca = alto
      SE numero_portas = 4
         SE numero_pessoas = 4
            SE capacidade_porta_malas = media
               SE seguranca = médio
SE preco = alto
   SE preco_manutencao = muito_alto
      SE numero_pessoas = 2
         SE capacidade_porta_malas = pequena
            SE seguranca = alto
               SE numero_portas = 2
               SE numero_portas = 3
ENTÃO aceitacao = nao_aceitavel

Regra: 2

SE preco = baixo
   SE preco_manutencao = baixo
      SE numero_pessoas = 2
         SE capacidade_porta_malas = media
            SE numero_portas = 4
               SE seguranca = médio
               SE seguranca = alto
            SE numero_portas = 5_ou_mais
               SE seguranca = médio
         SE capacidade_porta_malas = grande
            SE numero_portas = 5_ou_mais
               SE seguranca = baixo
            SE numero_portas = 3
               SE seguranca = médio
ENTÃO aceitacao = nao_aceitavel
```

Quadro 31 – Exemplo das regras geradas em formato hierárquico pelo algoritmo RX para o estudo de caso

```
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 2 E  capacidade_porta_malas = grande E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 2 E  capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 2 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 2 E  capacidade_porta_malas = grande E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 3 E  numero_pessoas = 4 E
   capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = boa

SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 4 E  numero_pessoas = 4 E
   capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = boa

SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = boa
```

Quadro 32 – Exemplo das regras geradas em formato plano pelo algoritmo RX para o estudo de caso

A validação das regras geradas também é feita pelo protótipo, e neste estudo de caso utilizou-se o conjunto completo de dados. O operador de rede neural foi instruído a entrar no modo de teste de regras e o experimento foi executado. No quadro 33 são apresentadas as saídas do protótipo com as estatísticas globais, avaliando número de execuções (ou número de disparos), número de acertos, número de disparos inválidos e a acurácia, esta indicada em percentual, considerando o número de acertos em relação ao número de execuções. No caso do estudo de caso, a acurácia das regras foi de 88,99 por cento.

```
[Informação]: Execução completa
[Informação]: Operador "OperadorESBanco" entrando em execução
[Informação]: Operador "OperadorESBanco" saindo da execução
[Informação]: Ponto de parada encontrado após operador "OperadorESBanco"
[Informação]: Operador "Operador RNA" entrando em execução
[Informação]: Estatísticas globais das regras:

Número de execuções.............:      1727
Número de acertos...............:      1537
Número de execuções sem disparo:       160
Número de disparos inválidos...:        30
Acurácia total (%)..............:     88.99
```

Quadro 33 – Saída do protótipo com as estatísticas da validação das regras em formato plano geradas pelo algoritmo RX para o estudo de caso

3.5 RESULTADOS E DISCUSSÃO

O protótipo e os operadores implementados atendem aos requisitos propostos, disponibilizando recursos para configurar, treinar e testar redes neurais artificiais, além de aplicar o algoritmo de extração de regras RX, extraindo efetivamente as regras das redes neurais. Também o objetivo de avaliar a acurácia das regras foi atendido pelo protótipo.

O algoritmo RX possui algumas deficiências notadas durante sua implementação. A primeira é o fato de que a lógica para a geração das regras a partir dos dados de entrada não estava bem definida na literatura, sendo implementada então uma lógica simples observando-se os exemplos disponibilizados pelos autores do algoritmo. Isso, porém, acarreta a geração de muitas regras e muitas premissas dependendo da dimensionalidade do espaço de entrada e da quantidade de valores possíveis de atributos no conjunto de treinamento.

Uma sugestão para diminuir a quantidade de regras geradas, dos próprios autores do algoritmo, é a utilização de alguma técnica de poda de redes neurais após o seu treinamento, visando reduzir o número de neurônios da rede ou das conexões entre eles. Tal método pode simplificar a estrutura da rede, gerando menor número de ativações agrupadas e menores combinações, além de removerem neurônios (de entrada ou ocultos) que possivelmente podem não afetar a resposta da rede.

Tentou-se também diminuir o número de regras com a geração das mesmas em formato hierárquico, ou seja, atribuindo um nível de importância aos atributos ou valores conforme sua freqüência no espaço de entrada e gerando regras conforme essa importância, aninhando condições abaixo de outras condições, para uma mesma resposta (conseqüente). Isso reduz o número de regras que possuem condições iguais e possibilitou o uso do operador `OU` nas condições das regras.

Uma característica notada no algoritmo RX é a grande necessidade de processamento do mesmo, principalmente pelo número de execuções da rede depois de treinada, em número de três: na geração do agrupamento, na busca das classes de cada combinação de agrupamento e na busca dos valores de entrada que geram valores de ativação agrupados dentro de cada combinação.

Tendo em vista a complexidade do projeto e a complexidade do algoritmo TREPAN, não houve tempo hábil para finalizar a implementação do mesmo. Esta atividade é citada nas possibilidades de projetos futuros do trabalho.

4 CONCLUSÕES

A partir da percepção das vantagens da extração de regras de redes neurais artificiais, especialmente para aplicações que necessitem de uma explicação sobre as hipóteses aprendidas por uma rede neural e principalmente na área de *data mining* com a tarefa de classificação de dados, foi verificada a possibilidade da implementação deste trabalho.

O protótipo implementado permite a construção, treinamento e teste de uma rede neural do tipo MLP e a posterior extração e avaliação de regras sobre o aprendizado da rede, utilizando do algoritmo RX. A construção em forma desacoplada do protótipo e dos operadores permite a extensão dos operadores, permitindo incrementar as funcionalidades do protótipo com novos operadores de carga ou algoritmos de redes e extração de regras.

O algoritmo TREPAN foi desenvolvido durante a implementação, mas devido a sua própria complexidade, tornou-se inviável concluir o mesmo para inclusão neste trabalho, em termos de tempo.

As ferramentas e técnicas utilizadas atenderam plenamente as necessidades do protótipo. No caso do ambiente de desenvolvimento Delphi 7, a possibilidade de uso dinâmico de propriedades de objetos em tempo de execução e o ambiente voltado a programação orientada a objeto, foram os fatos que facilitaram o desacoplamento dos operadores e dos protótipos, tornando fácil a leitura e configuração de seus parâmetros e a extensão de operadores.

4.1 EXTENSÕES

Durante o desenvolvimento do trabalho foram verificadas diversas possíveis melhorias no protótipo e operadores, muitas das quais não puderam ser contempladas neste trabalho. Portanto, as sugestões para extensão do trabalho são mostradas através de uma lista de tarefas, apresentada no quadro 34. Neste quadro, estão dispostas as descrições de cada tarefa e a sua complexidade, apresentadas em escala crescente conforme essa complexidade. A escala de complexidade, de um a dez, foi determinada pelo autor, conforme percepção própria.

TAREFA	COMPLEXIDADE
Criação do operador de extração de regras para o algoritmo TREPAN	10
Utilização de métodos de poda da rede neural artificial, para melhorar a eficiência e diminuir o conjunto de regras do algoritmo RX	10
Criação de métodos mais sofisticados de mapeamento dos atributos de dados para as entradas da rede neural, o que tornaria dispensável a necessidade de transformação dos dados de entrada em valores numéricos antes de serem carregados pelo operador de carga de dados. Também a inclusão manual da tabela de valores numéricos para categóricos (ou vice-versa) nos atributos de entrada, para formar as regras posteriormente, poderia ser removida pela implementação de mapeamento inverso na entrada, ou seja, poder descobrir o valor de um atributo através do valor do neurônio de entrada	8
Melhoria dos métodos de geração de regras do algoritmo RX a partir dos dados de entrada, utilizando, por exemplo, técnicas como agrupamento de valores em atributos contínuos e unificação dos dados de entrada de diferentes combinações de ativações que levam as mesmas classes. Atualmente a geração das regras é feita por combinação, independente do fato que elas algumas possam levar à mesma classe. A melhoria observada é de que isto pode diminuir o número de regras geradas	7
Melhoria do operador de entrada de dados, visando melhor seleção do conjunto de dados para treinamento e testes. A sugestão é a implementação de métodos de estratificação dos dados e da separação do conjunto de exemplos para treinamento, teste e validação da rede neural sem necessidade de configurar o operador para obter diferentes conjuntos de dado conforme a necessidade da rede	5

Quadro 34 – Lista de tarefas de melhoria do protótipo

REFERÊNCIAS BIBLIOGRÁFICAS

ABELÉM, Antônio J. G.; PACHECO, Marco A. C.; VELLASCO, Marley M. B. R. Modelagem de redes neurais artificiais para previsão de séries temporais. In: SIMPÓSIO BRASILEIRO DE REDES NEURAIS, 2., 1995, São Carlos. **Anais...** São Carlos: Universidade Federal de São Carlos, 1995. p. 107-112.

ALVES, Cristiano A. M. **Uma ferramenta de extração de regras de redes neurais**. 2001. 76 f. Dissertação (Mestrado em Engenharia Civil) – Curso de Pós-graduação em Engenharia Civil, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

ANDREWS, Robert et al.The truth will come to light: directions and challenges in extracting the knowledge embedded within trained artificial neural networks. **IEEE Transaction on Neural Networks,** Itke, v. 9, n. 6, p. 1057-1068, nov. 1998.

ANDREWS, Robert; DIEDERICH, Joachim; TICKLE, Alan B. A survey and critique of techniques for extracting rules from trained neural networks. **Knowledge-Based Systems,** v. 6 n. 8, p. 373-389, ago. 1995.

ASUNCION, Arthur; NEWMAN, David. J. **{UCI} Repository of machine learning databases.** Irvine, 2007. Disponível em: <http://mlearn.ics.uci.edu/MLRepository.html>. Acesso em: 23 ago. 2007.

ÁVILA, Bráulio Coelho. Data Mining. In: ESCOLA REGIONAL DE INFORMÁTICA DA SBC – REGIONAL SUL, 6., 1998, Blumenau. **Anais...** Curitiba: Champagnat, 1998. p. 87-106.

AZEVEDO, Fernando M.; BRASIL, Lourdes M.; OLIVEIRA, Roberto C. L. **Redes neurais com aplicações em controle e em sistemas especialistas.** Florianópolis: Bookstore, 2000.

BERRY, Michael J. A.; LINOFF, Gordon S. **Data mining techniques:** for marketing, sales, and customer relationship management. Hoboken: Wiley Computer Publishing.

BERTHOLD, Michael; HAND, David. **Intelligent data analysis.** An introduction. Cambridge: Springer-Verlag, 1999.

BIANCHI, Reinaldo A. C. **Uso de heurísticas para a aceleração do aprendizado por reforço**. 2004. 197 f. Tese (Doutorado em Engenharia da Computação) – Curso de Pós-graduação em Engenharia da Computação, Escola Politécnica da Universidade de São Paulo, São Paulo.

BRAGA, Bruno da Rocha. **Data mining usando árvores de decisão fuzzy**. Rio de Janeiro, mar. 2003. Disponível em: <www.cos.ufrj.br/~brunorb/docs/adfuzzy.pdf >. Acesso em: 01 set. 2007.

CASTRO, Fernando C.; CASTRO, Maria. **Redes neurais artificiais**. Porto Alegre, 2003. Disponível em: <http://www.ee.pucrs.br/~decastro/RNA_hp/RNA.html >. Acesso em: 01 out. 2007.

CHEN, Ming-Syan; HAN, Jiawei; YU, Philip S. Data mining: an overview from a database perspective. **IEEE Transactions on Knowledge and Data Engineering,** Itkee, v. 13, n. 5, p. 866-883, dez. 1996.

CRAVEN, Mark W. **Extracting compreehensible models from trained neural newtorks**. 1996. 199 f. Tese (Doutorado em Ciências da Computação) – Universidade de Wiscosin, Madison.

CRAVEN, Mark W.; SHAWLIK, Jude W. Using Sampling and Queries to Extract Rules from Trained Neural Networks. In: International Conference on Machine Learning, 11., 1994, New Jersey. **Proceedings...** New Jersey: Rutgers University, 1994. p. 37-45.

DARBARI, Ashish. Rule extraction from trained ANN: a survey. **Artificial Intelligence**, New York, v. 50, n. 3, p. 989-1002, mar. 2000.

FRAWLEY, Willian J.; MATHEUS, J. Cristopher.; PIATETSKY-SHAPIRO, Gregory. **Knowledge discovery in databases.** Cambridge: AAAI/MIT Press, 1991.

HAN, Jiawei; KAMBER, Micheline. **Data mining**. Concepts and techniques**.** São Francisco: Morgan Kaufmann Publishers, 2000.

HARRISON, Thomas H. **Intranet data warehouse**. São Paulo: Berkely Brasil, 1998.

HAYKIN, Simon. **Neural networks**. A comprehensive Foundation**.** New Jersey: Prentice Hall, 1994.

HAYKIN, Simon. **Redes neurais:** princípios e prática**.** Porto Alegre: Bookman, 2001.

KECMAN, Vojislav. **Learning and soft computing.** Support vector machines, neural networks and fuzzy logic models. Cambridge: MIT Press, 2001.

KREMER, Ricardo. **Sistema de apoio a decisão para previsões genéricas utilizando técnicas de Data Mining**. 1999. 57 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

KULIKOWSKI, Casimir A.; WEISS, Sholmon M. **Computer systems that learn: c**lassification and prediction methods from statistics, neural nets, machine learning and expert systems. San Mateo: Morgan Kauffmann, 1991.

LANGLEY, Pat; SIMON, Herbert A. Applications of machine learning and rule induction. **Communications of the ACM**, New York, v. 38, n. 11, p. 54-64, nov. 1995.

LEHR, Michael A.; RUMELHART, David E.; WIDROW, Bernard. Neural networks: **a**pplications in industry, bussines and science. **Communications of the ACM**, New York, v. 37, n. 3, p. 93-105, mar. 1994.

LIU, Huan; LU, Hongjun; SETIONO, Rudy. Neuro rule: a connectionist approach to data mining. In: VLDB Conference, 21., 1995, Zurich. **Proceedings...** Zurich, 1995. p. 478-489.

LOESCH, Cláudio; SARI, Solange T. **Redes neurais artificiais:** fundamentos e modelos. Blumenau: Editora da FURB, 1996.

MARTINELLI, Edmar. **Extração de conhecimento de redes neurais artificiais**. 1999. 113 f. Dissertação (Mestrado em Ciências da Computação) – Curso de Pós-graduação em Ciências da Computação, Universidade Federal de São Paulo, São Paulo.

MICHALSKI, Ryszard S. A theory and methodology of inductive learning. **Artificial Intelligence**, New York, v. 20, n. 2, p. 111-161, fev. 1983.

MITRA, Pabrita; MITRA, Sushmita; , PAL, Sankar K.Data mining in soft computing framewok. **IEEE Transaction on Neural Networks,** Itke, v. 13, n. 1, p. 3-14, jan. 2002.

MOONEY, Raymond J.; SHAVLIK, Jude W.; TOWEL, Geoffrey G. Symbolic and neural learning algorithms: an experimental comparison. **Machine Learning**, Berkeley, v. 6, n. 2, p. 111-143, mar. 1991.

NÚNEZ, Haydemar; RINCÓN, Denis; ROZO, Ely. Herramienta software con interfaz web para la interpretación simbólica de modelos neuronales. In: Conferência Latinoamericana de Informática, 30., 2004, Arequipa. **Proceedings...** Arequipa: Sociedad Peruana de Computación, 2004. p. 821-832.

RAPID-I. **YALE - Yet Another Learning Environment?** [Dortmund], [2007?]. Disponível em: <http://rapid-i.com/content/view/10/32/lang,en/>. Acesso em: 10 ago. 2007.

REZENDE, Solange O. **Sistemas inteligentes:** fundamentos e aplicações. Barueri: Manole, 2003.

ROSA, José L. A. **Classificação de dados através da otimização do método KNN-FUZZY em ambiente de computação paralela**. 2003. 110 f. Tese (Doutorado em Engenharia Civil) – Curso de Pós-graduação em Engenharia Civil, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

SANTOS, Raul T. **Uma proposta de metodologia para extração de regras de redes neurais artificiais utilizando algoritmos genéticos**. 2001. 124 f. Dissertação (Mestrado em Informática Industrial) – Curso de Pós-graduação em Informática Industrial, Centro Federal de Educação Tecnológica do Paraná, Curitiba.

SANTOS, Rodrigo G. **Utilização de técnicas de Data Mining na busca de conhecimento na WEB**. 200. 119 f. Trabalho de Conclusão de Curso (Bacharelado em Informática) – Instituto de Física e Matemática, Universidade Federal de Pelotas, Pelotas.

SETIONO, Rudy; LEOW, Whee K.; ZURADA, Jacek M. Extraction rules from artificial neural networks for nonlinear regression. **IEEE Transaction on Neural Networks,** Itke, v. 13, n. 3, p. 564-577, mar. 2002.

SILVA, Alex S. **Protótipo de software para classificação de impressão digital**. 1999. 70 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SPARK SYSTEMS PTY LTD. **Enterprise Architect.** [Creswick], [2007?]. Disponível em <http://www.sparxsystems.com/products/ea.html>. Acesso em: 10 set. 2007.

THRUN, Sebastian B. Extracting provably correct rules from artificial neural networks. **Communications of the ACM**, New York, v. 30, n. 5, p. 1-20, maio 1993.

APÊNDICE A – Conjunto completo de regras em formato hieráquico do algoritmo RX

No quadro 35 são apresentados o conjunto completo das regras em formato hierárquico do estudo de caso apresentado na operacionalidade da implementação:

```
Número de regras: 10
Regras:

 Regra: 1
 SE preco = muito_alto
    SE preco_manutencao = muito_alto
       SE numero_pessoas = 2
          SE seguranca = alto
             SE numero_portas = 4
                SE capacidade_porta_malas = pequena
                SE capacidade_porta_malas = media
             SE numero_portas = 2
                SE capacidade_porta_malas = media
                SE capacidade_porta_malas = grande
             SE numero_portas = 5_ou_mais
                SE capacidade_porta_malas = pequena
          SE seguranca = médio
             SE numero_portas = 4
                SE capacidade_porta_malas = pequena
                SE capacidade_porta_malas = grande
             SE capacidade_porta_malas = media
                SE numero_portas = 2
                SE numero_portas = 3
       SE numero_pessoas = 4
          SE capacidade_porta_malas = media
             SE seguranca = médio
                SE numero_portas = 3
                SE numero_portas = 4
                SE numero_portas = 5_ou_mais
             SE seguranca = baixo
                SE numero_portas = 2
                SE numero_portas = 4
          SE capacidade_porta_malas = grande
             SE seguranca = baixo
                SE numero_portas = 2
                SE numero_portas = 3
    SE preco_manutencao = alto
       SE numero_pessoas = 2
          SE numero_portas = 2
             SE capacidade_porta_malas = media
                SE seguranca = médio
                SE seguranca = alto
          SE numero_portas = 5_ou_mais
             SE capacidade_porta_malas = pequena
                SE seguranca = alto
       SE numero_portas = 4
          SE numero_pessoas = 4
             SE capacidade_porta_malas = media
                SE seguranca = médio
 SE preco = alto
    SE preco_manutencao = muito_alto
       SE numero_pessoas = 2
          SE capacidade_porta_malas = pequena
             SE seguranca = alto
                SE numero_portas = 2
                SE numero_portas = 3
 ENTÃO aceitacao = nao_aceitavel

 Regra: 2
 SE seguranca = alto
    SE numero_pessoas = 5_ou_mais
       SE preco = medio
          SE preco_manutencao = muito_alto
             SE capacidade_porta_malas = media
                SE numero_portas = 3
                SE numero_portas = 4
```

```
            SE numero_portas = 2
               SE capacidade_porta_malas = pequena
      SE preco = alto
         SE preco_manutencao = alto
            SE numero_portas = 5_ou_mais
               SE capacidade_porta_malas = pequena
      SE preco = muito_alto
         SE preco_manutencao = medio
            SE numero_portas = 2
               SE capacidade_porta_malas = media
   SE numero_pessoas = 4
      SE capacidade_porta_malas = pequena
         SE preco = medio
            SE preco_manutencao = muito_alto
               SE numero_portas = 3
         SE preco = muito_alto
            SE preco_manutencao = medio
               SE numero_portas = 2
ENTÃO aceitacao = nao_aceitavel

Regra: 3
SE preco = baixo
   SE preco_manutencao = muito_alto
      SE numero_portas = 2
         SE numero_pessoas = 5_ou_mais
            SE capacidade_porta_malas = pequena
               SE seguranca = alto
ENTÃO aceitacao = nao_aceitavel

Regra: 4
SE numero_pessoas = 5_ou_mais
   SE capacidade_porta_malas = pequena
      SE seguranca = baixo
         SE preco = muito_alto
            SE preco_manutencao = alto
               SE numero_portas = 2
               SE numero_portas = 4
            SE preco_manutencao = medio
               SE numero_portas = 2
            SE preco_manutencao = muito_alto
               SE numero_portas = 5_ou_mais
         SE preco = alto
            SE preco_manutencao = alto
               SE numero_portas = 2
               SE numero_portas = 3
            SE preco_manutencao = muito_alto
               SE numero_portas = 4
         SE preco = medio
            SE preco_manutencao = muito_alto
               SE numero_portas = 3
               SE numero_portas = 5_ou_mais
      SE seguranca = médio
         SE preco_manutencao = muito_alto
            SE preco = medio
               SE numero_portas = 4
               SE numero_portas = 5_ou_mais
            SE preco = alto
               SE numero_portas = 3
         SE preco = muito_alto
            SE numero_portas = 2
               SE preco_manutencao = alto
               SE preco_manutencao = medio
         SE preco = alto
            SE preco_manutencao = alto
               SE numero_portas = 5_ou_mais
   SE numero_portas = 2
      SE capacidade_porta_malas = media
         SE seguranca = baixo
            SE preco_manutencao = muito_alto
               SE preco = muito_alto
               SE preco = medio
            SE preco = alto
               SE preco_manutencao = alto
ENTÃO aceitacao = nao_aceitavel

Regra: 5
SE numero_pessoas = 2
```

```
    SE seguranca = médio
       SE preco = baixo
          SE preco_manutencao = baixo
             SE numero_portas = 3
                SE capacidade_porta_malas = pequena
                SE capacidade_porta_malas = media
             SE numero_portas = 4
                SE capacidade_porta_malas = pequena
             SE numero_portas = 2
                SE capacidade_porta_malas = media
          SE preco_manutencao = alto
             SE capacidade_porta_malas = pequena
                SE numero_portas = 2
                SE numero_portas = 3
          SE preco_manutencao = medio
             SE numero_portas = 3
                SE capacidade_porta_malas = media
       SE preco = medio
          SE capacidade_porta_malas = media
             SE preco_manutencao = medio
                SE numero_portas = 3
                SE numero_portas = 5_ou_mais
             SE preco_manutencao = baixo
                SE numero_portas = 4
          SE preco_manutencao = baixo
             SE numero_portas = 2
                SE capacidade_porta_malas = pequena
       SE preco = alto
          SE preco_manutencao = baixo
             SE numero_portas = 4
                SE capacidade_porta_malas = media
    SE preco = baixo
       SE capacidade_porta_malas = pequena
          SE seguranca = alto
             SE preco_manutencao = medio
                SE numero_portas = 3
                SE numero_portas = 5_ou_mais
             SE preco_manutencao = alto
                SE numero_portas = 3
ENTÃO aceitacao = nao_aceitavel

Regra: 6
SE seguranca = baixo
   SE numero_pessoas = 4
      SE capacidade_porta_malas = media
         SE preco_manutencao = baixo
            SE preco = muito_alto
               SE numero_portas = 2
               SE numero_portas = 4
               SE numero_portas = 5_ou_mais
            SE preco = medio
               SE numero_portas = 2
               SE numero_portas = 3
               SE numero_portas = 5_ou_mais
            SE preco = alto
               SE numero_portas = 2
               SE numero_portas = 4
               SE numero_portas = 5_ou_mais
            SE preco = baixo
               SE numero_portas = 4
         SE preco_manutencao = medio
            SE preco = alto
               SE numero_portas = 3
               SE numero_portas = 4
               SE numero_portas = 5_ou_mais
            SE preco = baixo
               SE numero_portas = 2
               SE numero_portas = 3
               SE numero_portas = 4
            SE preco = muito_alto
               SE numero_portas = 2
         SE preco_manutencao = alto
            SE numero_portas = 4
               SE preco = alto
               SE preco = medio
               SE preco = baixo
            SE numero_portas = 5_ou_mais
```

```
                SE preco = medio
                SE preco = baixo
             SE preco = alto
                SE numero_portas = 2
          SE preco_manutencao = muito_alto
             SE preco = baixo
                SE numero_portas = 4
             SE preco = medio
                SE numero_portas = 2
       SE capacidade_porta_malas = pequena
          SE preco_manutencao = baixo
             SE preco = alto
                SE numero_portas = 3
                SE numero_portas = 4
                SE numero_portas = 5_ou_mais
             SE preco = baixo
                SE numero_portas = 2
                SE numero_portas = 3
                SE numero_portas = 4
             SE preco = medio
                SE numero_portas = 2
                SE numero_portas = 3
             SE preco = muito_alto
                SE numero_portas = 2
                SE numero_portas = 4
          SE preco = baixo
             SE preco_manutencao = muito_alto
                SE numero_portas = 2
                SE numero_portas = 5_ou_mais
             SE preco_manutencao = medio
                SE numero_portas = 2
                SE numero_portas = 4
             SE preco_manutencao = alto
                SE numero_portas = 4
          SE preco = medio
             SE preco_manutencao = alto
                SE numero_portas = 2
             SE preco_manutencao = muito_alto
                SE numero_portas = 5_ou_mais
          SE numero_portas = 4
             SE preco = alto
                SE preco_manutencao = alto
             SE preco = muito_alto
                SE preco_manutencao = medio
       SE capacidade_porta_malas = grande
          SE preco = medio
             SE numero_portas = 2
                SE preco_manutencao = muito_alto
                SE preco_manutencao = baixo
             SE preco_manutencao = alto
                SE numero_portas = 4
          SE preco = baixo
             SE preco_manutencao = alto
                SE numero_portas = 3
                SE numero_portas = 4
             SE preco_manutencao = muito_alto
                SE numero_portas = 3
          SE preco = alto
             SE preco_manutencao = baixo
                SE numero_portas = 4
          SE preco = muito_alto
             SE preco_manutencao = medio
                SE numero_portas = 2
    SE numero_pessoas = 2
       SE capacidade_porta_malas = media
          SE numero_portas = 4
             SE preco_manutencao = medio
                SE preco = muito_alto
                SE preco = alto
                SE preco = medio
                SE preco = baixo
             SE preco_manutencao = baixo
                SE preco = muito_alto
                SE preco = alto
                SE preco = baixo
             SE preco = baixo
                SE preco_manutencao = muito_alto
```

```
            SE preco = alto
               SE preco_manutencao = alto
         SE numero_portas = 5_ou_mais
            SE preco = medio
               SE preco_manutencao = medio
               SE preco_manutencao = baixo
            SE preco = baixo
               SE preco_manutencao = muito_alto
               SE preco_manutencao = medio
            SE preco = muito_alto
               SE preco_manutencao = baixo
         SE preco = medio
            SE preco_manutencao = medio
               SE numero_portas = 3
            SE preco_manutencao = alto
               SE numero_portas = 2
         SE preco = alto
            SE preco_manutencao = baixo
               SE numero_portas = 3
      SE capacidade_porta_malas = pequena
         SE preco = medio
            SE preco_manutencao = muito_alto
               SE numero_portas = 3
               SE numero_portas = 5_ou_mais
            SE preco_manutencao = alto
               SE numero_portas = 3
            SE preco_manutencao = baixo
               SE numero_portas = 4
            SE preco_manutencao = medio
               SE numero_portas = 5_ou_mais
         SE preco_manutencao = baixo
            SE preco = muito_alto
               SE numero_portas = 2
               SE numero_portas = 4
               SE numero_portas = 5_ou_mais
            SE preco = baixo
               SE numero_portas = 4
         SE preco = alto
            SE numero_portas = 2
               SE preco_manutencao = alto
               SE preco_manutencao = medio
         SE preco = baixo
            SE preco_manutencao = medio
               SE numero_portas = 5_ou_mais
      SE capacidade_porta_malas = grande
         SE preco = baixo
            SE numero_portas = 3
               SE preco_manutencao = muito_alto
               SE preco_manutencao = medio
               SE preco_manutencao = baixo
            SE preco_manutencao = alto
               SE numero_portas = 4
               SE numero_portas = 5_ou_mais
            SE preco_manutencao = medio
               SE numero_portas = 5_ou_mais
         SE preco = alto
            SE preco_manutencao = medio
               SE numero_portas = 2
               SE numero_portas = 4
            SE preco_manutencao = baixo
               SE numero_portas = 4
         SE preco = medio
            SE preco_manutencao = baixo
               SE numero_portas = 5_ou_mais
            SE preco_manutencao = medio
               SE numero_portas = 3
SE seguranca = médio
   SE capacidade_porta_malas = pequena
      SE numero_pessoas = 2
         SE preco = alto
            SE preco_manutencao = medio
               SE numero_portas = 4
               SE numero_portas = 5_ou_mais
            SE preco_manutencao = alto
               SE numero_portas = 2
               SE numero_portas = 3
         SE numero_portas = 4
```

```
            SE preco = muito_alto
               SE preco_manutencao = medio
               SE preco_manutencao = baixo
            SE preco = baixo
               SE preco_manutencao = muito_alto
         SE preco = medio
            SE preco_manutencao = muito_alto
               SE numero_portas = 2
      SE numero_pessoas = 4
         SE preco_manutencao = alto
            SE preco = alto
               SE numero_portas = 3
               SE numero_portas = 4
            SE preco = medio
               SE numero_portas = 2
         SE preco = muito_alto
            SE preco_manutencao = baixo
               SE numero_portas = 2
            SE preco_manutencao = medio
               SE numero_portas = 3
         SE preco = medio
            SE preco_manutencao = muito_alto
               SE numero_portas = 4
   SE numero_portas = 2
      SE numero_pessoas = 2
         SE capacidade_porta_malas = media
            SE preco = alto
               SE preco_manutencao = medio
            SE preco = baixo
               SE preco_manutencao = muito_alto
            SE preco = medio
               SE preco_manutencao = alto
ENTÃO aceitacao = nao_aceitavel

Regra: 7
SE seguranca = médio
   SE numero_pessoas = 4
      SE capacidade_porta_malas = grande
         SE preco = medio
            SE preco_manutencao = muito_alto
               SE numero_portas = 3
               SE numero_portas = 4
               SE numero_portas = 5_ou_mais
            SE preco_manutencao = alto
               SE numero_portas = 5_ou_mais
         SE preco = alto
            SE preco_manutencao = alto
               SE numero_portas = 3
               SE numero_portas = 4
               SE numero_portas = 5_ou_mais
         SE preco = muito_alto
            SE preco_manutencao = baixo
               SE numero_portas = 3
            SE preco_manutencao = medio
               SE numero_portas = 2
         SE preco = baixo
            SE preco_manutencao = muito_alto
               SE numero_portas = 4
      SE capacidade_porta_malas = media
         SE preco = medio
            SE preco_manutencao = alto
               SE numero_portas = 4
               SE numero_portas = 5_ou_mais
            SE preco_manutencao = muito_alto
               SE numero_portas = 2
         SE preco = alto
            SE preco_manutencao = medio
               SE numero_portas = 4
            SE preco_manutencao = alto
               SE numero_portas = 5_ou_mais
         SE preco = muito_alto
            SE preco_manutencao = baixo
               SE numero_portas = 4
   SE numero_pessoas = 5_ou_mais
      SE capacidade_porta_malas = grande
         SE preco = muito_alto
            SE preco_manutencao = baixo
```

```
                SE numero_portas = 3
                SE numero_portas = 4
                SE numero_portas = 5_ou_mais
          SE preco_manutencao = alto
             SE preco = alto
                SE numero_portas = 3
                SE numero_portas = 4
             SE preco = medio
                SE numero_portas = 4
       SE capacidade_porta_malas = media
          SE numero_portas = 5_ou_mais
             SE preco_manutencao = muito_alto
                SE preco = medio
                SE preco = baixo
             SE preco = alto
                SE preco_manutencao = medio
          SE preco = muito_alto
             SE preco_manutencao = medio
                SE numero_portas = 3
ENTÃO aceitacao = aceitavel

Regra: 8
SE seguranca = alto
   SE numero_pessoas = 4
      SE capacidade_porta_malas = media
         SE preco = medio
            SE numero_portas = 5_ou_mais
               SE preco_manutencao = muito_alto
               SE preco_manutencao = alto
            SE preco_manutencao = medio
               SE numero_portas = 2
         SE preco = muito_alto
            SE preco_manutencao = medio
               SE numero_portas = 3
               SE numero_portas = 5_ou_mais
            SE preco_manutencao = baixo
               SE numero_portas = 4
         SE preco = baixo
            SE preco_manutencao = muito_alto
               SE numero_portas = 3
               SE numero_portas = 4
            SE preco_manutencao = alto
               SE numero_portas = 3
         SE preco = alto
            SE preco_manutencao = baixo
               SE numero_portas = 2
      SE capacidade_porta_malas = pequena
         SE preco = alto
            SE preco_manutencao = medio
               SE numero_portas = 4
               SE numero_portas = 5_ou_mais
            SE preco_manutencao = alto
               SE numero_portas = 5_ou_mais
         SE preco = baixo
            SE preco_manutencao = muito_alto
               SE numero_portas = 3
               SE numero_portas = 4
               SE numero_portas = 5_ou_mais
         SE numero_portas = 2
            SE preco = medio
               SE preco_manutencao = alto
            SE preco = muito_alto
               SE preco_manutencao = baixo
      SE capacidade_porta_malas = grande
         SE preco = medio
            SE preco_manutencao = muito_alto
               SE numero_portas = 2
               SE numero_portas = 3
         SE preco = muito_alto
            SE preco_manutencao = baixo
               SE numero_portas = 3
   SE numero_pessoas = 5_ou_mais
      SE preco = alto
         SE preco_manutencao = medio
            SE capacidade_porta_malas = pequena
               SE numero_portas = 3
               SE numero_portas = 4
```

```
                  SE numero_portas = 5_ou_mais
               SE numero_portas = 3
                  SE capacidade_porta_malas = media
               SE numero_portas = 2
                  SE capacidade_porta_malas = grande
            SE preco_manutencao = alto
               SE capacidade_porta_malas = grande
                  SE numero_portas = 2
                  SE numero_portas = 5_ou_mais
         SE capacidade_porta_malas = media
            SE preco = baixo
               SE preco_manutencao = muito_alto
                  SE numero_portas = 3
                  SE numero_portas = 5_ou_mais
               SE preco_manutencao = alto
                  SE numero_portas = 2
            SE preco = medio
               SE preco_manutencao = medio
                  SE numero_portas = 2
               SE preco_manutencao = alto
                  SE numero_portas = 3
            SE preco = muito_alto
               SE preco_manutencao = baixo
                  SE numero_portas = 2
         SE preco = medio
            SE preco_manutencao = alto
               SE capacidade_porta_malas = pequena
                  SE numero_portas = 4
                  SE numero_portas = 5_ou_mais
            SE preco_manutencao = muito_alto
               SE numero_portas = 4
                  SE capacidade_porta_malas = grande
         SE preco = baixo
            SE preco_manutencao = muito_alto
               SE numero_portas = 4
                  SE capacidade_porta_malas = pequena
SE capacidade_porta_malas = grande
   SE seguranca = médio
      SE numero_pessoas = 4
         SE preco = medio
            SE preco_manutencao = medio
               SE numero_portas = 3
               SE numero_portas = 5_ou_mais
         SE preco = alto
            SE preco_manutencao = baixo
               SE numero_portas = 2
               SE numero_portas = 5_ou_mais
         SE preco = baixo
            SE preco_manutencao = alto
               SE numero_portas = 4
      SE preco = baixo
         SE preco_manutencao = alto
            SE numero_portas = 3
               SE numero_pessoas = 5_ou_mais
ENTÃO aceitacao = aceitavel

Regra: 9
SE preco = baixo
   SE preco_manutencao = baixo
      SE numero_pessoas = 2
         SE capacidade_porta_malas = media
            SE numero_portas = 4
               SE seguranca = médio
               SE seguranca = alto
            SE numero_portas = 5_ou_mais
               SE seguranca = médio
         SE capacidade_porta_malas = grande
            SE numero_portas = 5_ou_mais
               SE seguranca = baixo
            SE numero_portas = 3
               SE seguranca = médio
ENTÃO aceitacao = nao_aceitavel

Regra: 10
SE preco = baixo
   SE preco_manutencao = baixo
      SE numero_pessoas = 4
```

```
          SE seguranca = médio
             SE capacidade_porta_malas = grande
                SE numero_portas = 3
                SE numero_portas = 4
                SE numero_portas = 5_ou_mais
             SE numero_portas = 5_ou_mais
                SE capacidade_porta_malas = media
          SE capacidade_porta_malas = pequena
             SE seguranca = alto
                SE numero_portas = 3
                SE numero_portas = 5_ou_mais
       SE numero_pessoas = 5_ou_mais
          SE numero_portas = 5_ou_mais
             SE seguranca = médio
                SE capacidade_porta_malas = pequena
                SE capacidade_porta_malas = grande
          SE numero_portas = 3
             SE capacidade_porta_malas = media
                SE seguranca = alto
   SE preco_manutencao = medio
      SE numero_portas = 5_ou_mais
         SE numero_pessoas = 5_ou_mais
            SE capacidade_porta_malas = media
               SE seguranca = médio
            SE capacidade_porta_malas = pequena
               SE seguranca = alto
         SE numero_pessoas = 4
            SE capacidade_porta_malas = pequena
               SE seguranca = alto
      SE numero_portas = 4
         SE numero_pessoas = 5_ou_mais
            SE capacidade_porta_malas = pequena
               SE seguranca = alto
         SE numero_pessoas = 4
            SE capacidade_porta_malas = grande
               SE seguranca = médio
      SE numero_portas = 3
         SE numero_pessoas = 4
            SE capacidade_porta_malas = grande
               SE seguranca = médio
SE preco = medio
   SE preco_manutencao = baixo
      SE seguranca = médio
         SE capacidade_porta_malas = grande
            SE numero_portas = 3
               SE numero_pessoas = 4
               SE numero_pessoas = 5_ou_mais
            SE numero_portas = 5_ou_mais
               SE numero_pessoas = 5_ou_mais
            SE numero_portas = 2
               SE numero_pessoas = 4
         SE numero_portas = 5_ou_mais
            SE numero_pessoas = 5_ou_mais
               SE capacidade_porta_malas = media
ENTÃO aceitacao = boa
```

Quadro 35 - Conjunto completo de regras em formato hieráquico geradas pelo algoritmo RX

APÊNDICE B – Conjunto completo das regras em formato plano do algoritmo RX

No quadro 36 são apresentados o conjunto completo das regras em formato plano do estudo de caso apresentado na operacionalidade da implementação:

```
Número de regras: 267
Regras:

Regra: 1
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 2 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 2
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 3
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 4
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 2 E  capacidade_porta_malas = grande E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 5
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 6
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 2 E  capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 7
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 2 E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 8
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 9
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 3 E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 10
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 3 E
   numero_pessoas = 4 E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 11
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 4 E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 12
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 13
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 4 E  capacidade_porta_malas = media E  seguranca = baixo
```

```
ENTAO aceitacao = nao_aceitavel

Regra: 14
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 4 E  capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 15
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 4 E  capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 16
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 3 E
   numero_pessoas = 4 E  capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 17
SE preco = muito_alto E  preco_manutencao = alto E  numero_portas = 2 E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 18
SE preco = muito_alto E  preco_manutencao = alto E  numero_portas = 2 E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 19
SE preco = muito_alto E  preco_manutencao = alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 20
SE preco = muito_alto E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 4 E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 21
SE preco = alto E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 2 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 22
SE preco = alto E  preco_manutencao = muito_alto E  numero_portas = 3 E
   numero_pessoas = 2 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 23
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 3 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 24
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 25
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 26
SE preco = alto E  preco_manutencao = alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 27
SE preco = muito_alto E  preco_manutencao = medio E  numero_portas = 2 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 28
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 3 E
   numero_pessoas = 4 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = nao_aceitavel
```

```
Regra: 29
SE preco = muito_alto E  preco_manutencao = medio E  numero_portas = 2 E
   numero_pessoas = 4 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 30
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 31
SE preco = muito_alto E  preco_manutencao = alto E  numero_portas = 2 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 32
SE preco = muito_alto E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 33
SE preco = muito_alto E  preco_manutencao = medio E  numero_portas = 2 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 34
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 35
SE preco = alto E  preco_manutencao = alto E  numero_portas = 2 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 36
SE preco = alto E  preco_manutencao = alto E  numero_portas = 3 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 37
SE preco = alto E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 38
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 3 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 39
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 40
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 41
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 42
SE preco = alto E  preco_manutencao = muito_alto E  numero_portas = 3 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 43
SE preco = muito_alto E  preco_manutencao = alto E  numero_portas = 2 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 44
SE preco = muito_alto E  preco_manutencao = medio E  numero_portas = 2 E
```

```
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 45
SE preco = alto E  preco_manutencao = alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 46
SE preco = muito_alto E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 47
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 48
SE preco = alto E  preco_manutencao = alto E  numero_portas = 2 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 49
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 3 E
   numero_pessoas = 2 E  capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 50
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 3 E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 51
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 2 E  capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 52
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 2 E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 53
SE preco = baixo E  preco_manutencao = alto E  numero_portas = 2 E
   numero_pessoas = 2 E  capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 54
SE preco = baixo E  preco_manutencao = alto E  numero_portas = 3 E
   numero_pessoas = 2 E  capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 55
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 3 E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 56
SE preco = medio E  preco_manutencao = medio E  numero_portas = 3 E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 57
SE preco = medio E  preco_manutencao = medio E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 58
SE preco = medio E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 59
SE preco = medio E  preco_manutencao = baixo E  numero_portas = 2 E
   numero_pessoas = 2 E  capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel
```

```
Regra: 60
SE preco = alto E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 61
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 3 E
   numero_pessoas = 2 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 62
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 63
SE preco = baixo E  preco_manutencao = alto E  numero_portas = 3 E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = nao_aceitavel

Regra: 64
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 65
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 66
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 67
SE preco = medio E  preco_manutencao = baixo E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 68
SE preco = medio E  preco_manutencao = baixo E  numero_portas = 3 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 69
SE preco = medio E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 70
SE preco = alto E  preco_manutencao = baixo E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 71
SE preco = alto E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 72
SE preco = alto E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 73
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 74
SE preco = alto E  preco_manutencao = medio E  numero_portas = 3 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 75
```

```
SE preco = alto E  preco_manutencao = medio E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 76
SE preco = alto E  preco_manutencao = medio E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 77
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 78
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 3 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 79
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 80
SE preco = muito_alto E  preco_manutencao = medio E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 81
SE preco = alto E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 82
SE preco = medio E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 83
SE preco = baixo E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 84
SE preco = medio E  preco_manutencao = alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 85
SE preco = baixo E  preco_manutencao = alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 86
SE preco = alto E  preco_manutencao = alto E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 87
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 88
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 89
SE preco = alto E  preco_manutencao = baixo E  numero_portas = 3 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 90
SE preco = alto E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
```

```
ENTAO aceitacao = nao_aceitavel

Regra: 91
SE preco = alto E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 92
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 93
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 3 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 94
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 95
SE preco = medio E  preco_manutencao = baixo E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 96
SE preco = medio E  preco_manutencao = baixo E  numero_portas = 3 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 97
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 98
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 99
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 100
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 101
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 102
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 103
SE preco = baixo E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 104
SE preco = medio E  preco_manutencao = alto E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 105
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel
```

```
Regra: 106
SE preco = alto E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 107
SE preco = muito_alto E  preco_manutencao = medio E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 108
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 109
SE preco = medio E  preco_manutencao = baixo E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 110
SE preco = medio E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 111
SE preco = baixo E  preco_manutencao = alto E  numero_portas = 3 E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 112
SE preco = baixo E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 113
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 3 E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 114
SE preco = alto E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 115
SE preco = muito_alto E  preco_manutencao = medio E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 116
SE preco = muito_alto E  preco_manutencao = medio E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 117
SE preco = alto E  preco_manutencao = medio E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 118
SE preco = medio E  preco_manutencao = medio E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 119
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 120
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 121
SE preco = alto E  preco_manutencao = baixo E  numero_portas = 4 E
```

```
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 122
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 123
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 124
SE preco = alto E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 125
SE preco = medio E  preco_manutencao = medio E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 126
SE preco = medio E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 127
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 128
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 129
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 130
SE preco = medio E  preco_manutencao = medio E  numero_portas = 3 E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 131
SE preco = medio E  preco_manutencao = alto E  numero_portas = 2 E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 132
SE preco = alto E  preco_manutencao = baixo E  numero_portas = 3 E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 133
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 3 E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 134
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 135
SE preco = medio E  preco_manutencao = alto E  numero_portas = 3 E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 136
SE preco = medio E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel
```

```
Regra: 137
SE preco = medio E  preco_manutencao = medio E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 138
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 2 E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 139
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 140
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 141
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 142
SE preco = alto E  preco_manutencao = alto E  numero_portas = 2 E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 143
SE preco = alto E  preco_manutencao = medio E  numero_portas = 2 E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 144
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 145
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 3 E
   numero_pessoas = 2 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 146
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 3 E
   numero_pessoas = 2 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 147
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 3 E
   numero_pessoas = 2 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 148
SE preco = baixo E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 149
SE preco = baixo E  preco_manutencao = alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 150
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 151
SE preco = alto E  preco_manutencao = medio E  numero_portas = 2 E
   numero_pessoas = 2 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 152
```

```
SE preco = alto E  preco_manutencao = medio E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 153
SE preco = alto E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 154
SE preco = medio E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 155
SE preco = medio E  preco_manutencao = medio E  numero_portas = 3 E
   numero_pessoas = 2 E capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 156
SE preco = alto E  preco_manutencao = medio E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 157
SE preco = alto E  preco_manutencao = medio E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 158
SE preco = alto E  preco_manutencao = alto E  numero_portas = 2 E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 159
SE preco = alto E  preco_manutencao = alto E  numero_portas = 3 E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 160
SE preco = muito_alto E  preco_manutencao = medio E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 161
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 162
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 163
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 2 E capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 164
SE preco = alto E  preco_manutencao = alto E  numero_portas = 3 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 165
SE preco = alto E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 166
SE preco = medio E  preco_manutencao = alto E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 167
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = médio
```

```
ENTAO aceitacao = nao_aceitavel

Regra: 168
SE preco = muito_alto E  preco_manutencao = medio E  numero_portas = 3 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 169
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = pequena E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 170
SE preco = alto E  preco_manutencao = medio E  numero_portas = 2 E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 171
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 172
SE preco = medio E  preco_manutencao = alto E  numero_portas = 2 E
   numero_pessoas = 2 E capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 173
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 3 E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 174
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 175
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 176
SE preco = medio E  preco_manutencao = alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 177
SE preco = alto E  preco_manutencao = alto E  numero_portas = 3 E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 178
SE preco = alto E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 179
SE preco = alto E  preco_manutencao = alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 180
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 3 E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 181
SE preco = muito_alto E  preco_manutencao = medio E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 182
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel
```

```
Regra: 183
SE preco = medio E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 184
SE preco = medio E  preco_manutencao = alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 185
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 186
SE preco = alto E  preco_manutencao = medio E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 187
SE preco = alto E  preco_manutencao = alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 188
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 189
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 3 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 190
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 191
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 192
SE preco = alto E  preco_manutencao = alto E  numero_portas = 3 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 193
SE preco = alto E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 194
SE preco = medio E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 195
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 196
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 197
SE preco = alto E  preco_manutencao = medio E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 198
SE preco = muito_alto E  preco_manutencao = medio E  numero_portas = 3 E
```

```
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 199
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 200
SE preco = medio E  preco_manutencao = alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 201
SE preco = medio E  preco_manutencao = medio E  numero_portas = 2 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 202
SE preco = muito_alto E  preco_manutencao = medio E  numero_portas = 3 E
   numero_pessoas = 4 E capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 203
SE preco = muito_alto E  preco_manutencao = medio E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 204
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 4 E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 205
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 3 E
   numero_pessoas = 4 E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 206
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 4 E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 207
SE preco = baixo E  preco_manutencao = alto E  numero_portas = 3 E
   numero_pessoas = 4 E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 208
SE preco = alto E  preco_manutencao = baixo E  numero_portas = 2 E
   numero_pessoas = 4 E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 209
SE preco = alto E  preco_manutencao = medio E  numero_portas = 4 E
   numero_pessoas = 4 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 210
SE preco = alto E  preco_manutencao = medio E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 211
SE preco = alto E  preco_manutencao = alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 212
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 3 E
   numero_pessoas = 4 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 213
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 4 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = aceitavel
```

```
Regra: 214
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 215
SE preco = medio E  preco_manutencao = alto E  numero_portas = 2 E
   numero_pessoas = 4 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 216
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 2 E
   numero_pessoas = 4 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 217
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 2 E
   numero_pessoas = 4 E  capacidade_porta_malas = grande E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 218
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 3 E
   numero_pessoas = 4 E  capacidade_porta_malas = grande E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 219
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 3 E
   numero_pessoas = 4 E  capacidade_porta_malas = grande E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 220
SE preco = alto E  preco_manutencao = medio E  numero_portas = 3 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 221
SE preco = alto E  preco_manutencao = medio E  numero_portas = 4 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 222
SE preco = alto E  preco_manutencao = medio E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 223
SE preco = alto E  preco_manutencao = medio E  numero_portas = 3 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 224
SE preco = alto E  preco_manutencao = medio E  numero_portas = 2 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = grande E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 225
SE preco = alto E  preco_manutencao = alto E  numero_portas = 2 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = grande E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 226
SE preco = alto E  preco_manutencao = alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = grande E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 227
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 3 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 228
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 229
```

```
SE preco = baixo E  preco_manutencao = alto E  numero_portas = 2 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 230
SE preco = medio E  preco_manutencao = medio E  numero_portas = 2 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 231
SE preco = medio E  preco_manutencao = alto E  numero_portas = 3 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 232
SE preco = muito_alto E  preco_manutencao = baixo E  numero_portas = 2 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 233
SE preco = medio E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 234
SE preco = medio E  preco_manutencao = alto E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 235
SE preco = medio E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = grande E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 236
SE preco = baixo E  preco_manutencao = muito_alto E  numero_portas = 4 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = aceitavel

Regra: 237
SE preco = medio E  preco_manutencao = medio E  numero_portas = 3 E
   numero_pessoas = 4 E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 238
SE preco = medio E  preco_manutencao = medio E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 239
SE preco = alto E  preco_manutencao = baixo E  numero_portas = 2 E
   numero_pessoas = 4 E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 240
SE preco = alto E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 241
SE preco = baixo E  preco_manutencao = alto E  numero_portas = 4 E
   numero_pessoas = 4 E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 242
SE preco = baixo E  preco_manutencao = alto E  numero_portas = 3 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = aceitavel

Regra: 243
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 244
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = alto
```

```
ENTAO aceitacao = nao_aceitavel

Regra: 245
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 246
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 2 E  capacidade_porta_malas = grande E  seguranca = baixo
ENTAO aceitacao = nao_aceitavel

Regra: 247
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 3 E
   numero_pessoas = 2 E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = nao_aceitavel

Regra: 248
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 3 E
   numero_pessoas = 4 E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = boa

Regra: 249
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 4 E
   numero_pessoas = 4 E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = boa

Regra: 250
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = boa

Regra: 251
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = boa

Regra: 252
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 3 E
   numero_pessoas = 4 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = boa

Regra: 253
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = boa

Regra: 254
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = médio E
NTAO aceitacao = boa

Regra: 255
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = boa

Regra: 256
SE preco = baixo E  preco_manutencao = baixo E  numero_portas = 3 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = alto
ENTAO aceitacao = boa

Regra: 257
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = boa

Regra: 258
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = boa

Regra: 259
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 5_ou_mais E
   numero_pessoas = 4 E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = boa
```

```
Regra: 260
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 4 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = pequena E  seguranca = alto
ENTAO aceitacao = boa

Regra: 261
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 4 E
   numero_pessoas = 4 E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = boa

Regra: 262
SE preco = baixo E  preco_manutencao = medio E  numero_portas = 3 E
   numero_pessoas = 4 E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = boa

Regra: 263
SE preco = medio E  preco_manutencao = baixo E  numero_portas = 3 E
   numero_pessoas = 4 E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = boa

Regra: 264
SE preco = medio E  preco_manutencao = baixo E  numero_portas = 3 E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = boa

Regra: 265
SE preco = medio E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = boa

Regra: 266
SE preco = medio E  preco_manutencao = baixo E  numero_portas = 2 E
   numero_pessoas = 4 E  capacidade_porta_malas = grande E  seguranca = médio
ENTAO aceitacao = boa

Regra: 267
SE preco = medio E  preco_manutencao = baixo E  numero_portas = 5_ou_mais E
   numero_pessoas = 5_ou_mais E  capacidade_porta_malas = media E  seguranca = médio
ENTAO aceitacao = boa
```

Quadro 36 - Conjunto completo de regras em formato hieráquico geradas pelo algoritmo RX