Aplicação de Técnicas de Mineração de Dados em problemas de Classificação

YANN FABRICIO CARDOSO DE FIGUEIREDO
yann.fabricio@hotmail.com
UFPA

LIDIO MAURO LIMA DE CAMPOS
danilo8@gmail.com
UFPA

limadecampos@gmail.com
UFPA

Resumo: As redes neurais artificiais (RNAs) são bastante utilizadas em diversos tipos de problemas, como por exemplo, na predição de séries temporais e classificação de registros em uma base de dados. Diante dessa importância, o objetivo desta pesquisa é a classificação de duas bases dados, hepatitis e germain credit, onde o propósito é classificar se os registros do dataset são referentes a alguém morto ou vivo, no caso do hepatis, e se o crédito de uma pessoa é bom ou ruim, no caso do germain credit. Foram utilizadas RNAs multilayer perceptron de 3 e 4 camadas e recorrente com 3 camadas, e para medir a eficácia e o melhor modelo foram usadas as técnicas hold out e k-fold cross validation. Para o hepatitis a rede de melhor desempenho foi a recorrente com 90% de acurácia, enquanto para o germain credit foi a direta de 4 camadas com 96.53%, sendo este último alcançado após a eliminação de outliers pelo método das discrepâncias. O algoritmo utilizado para a rede direta foi o backpropagation, enquanto para a recorrente foi o backpropagation through time (BPTT).

Palavras Chave: Redes Neurais - Classificação - KNN - Machine learning -

1. INTRODUÇÃO

As RNAs (Redes Neurais Artificiais) possuem a capacidade, em conjunto com um algoritmo, de aprender com base em treinamento da rede com dados agrupados, e assim encontrar padrões para classificar corretamente registros de uma base de dados. O treinamento melhora a rede continuamente com base em mudanças nos parâmetros que influenciam no aprendizado da RNA, como taxa de aprendizado, número de épocas e quantidade de neurônios das camadas internas da rede. Ajustes nesses parâmetros ditam todo o comportamento e eficácia da rede, e são muito importantes para ocorrer um bom treino e consequentemente para uma boa validação da capacidade de generalização da RNA. Os experimentos e testes feitos em cima dessa necessidade de validar a eficácia da rede proporcionam um modelo muito melhor para ser utilizado em produção.

Para avaliar o modelo em teste existem diversas técnicas com este propósito, sendo que se destacam o *hold-out* e *o k-fold cross validation*. No *hold-out* a base de dados original é dividida em duas partes, sendo uma para treinamento e outra para validação, e geralmente a proporção dessa divisão é 2/3 para treino e 1/3 para teste. No *k-fold* a base de dados original é divida em k partes de tamanho parecido ou similar, e cada uma dessas partições será usada para treino enquanto as outras serão teste, sendo que a função de cada uma vai variando a cada nova rodada de simulações na rede.

Este trabalho têm o intuito de utilizar RNAs feedforward (3 e 4 camadas) e recorrente para tarefas de classificação, especificamente nas bases de dados *hepatitis* e *germain credit*. O primeiro *dataset* contém dados médicos biológicos de pacientes com hepatite, sendo um deles o atributo a ser classificado na RNA, e ele diz se a pessoa de cada registro está viva ou morta. O segundo *dataset* contém dados com informações pessoais, de residência e financeiras de clientes de um banco, e esta base têm um atributo classificador para dizer se este cliente possui um risco de crédito bom ou ruim. Nos experimentos foram utilizadas RNAs com três arquiteturas distintas, que são a *feedfoward* (rede direta), de 3 e 4 camadas, e a rede recorrente, contando ainda com os algoritmos *backpropagation* e *backpropagation through time* (BTT).

2. REVISÃO DE LITERATURA

O trabalho de Lopes e Neves (2019) tem o intuito de disponibilizar uma forma menos invasiva de diagnosticar o tipo de hepatite de pacientes, através do uso de técnicas de aprendizado de máquina e que inclusive são mais eficientes na classificação ao se comparar com o método tradicional de diagnóstico. A técnica escolhida pelo autor do projeto foi a Rede Neural Artificial (RNA) do tipo *perceptron* multicamadas, treinada com o algoritmo *backpropagation*. Os resultados da classificação da RNA para danos hepáticos em hepatites virais são comparados com os obtidos pela escala METAVIR que mede atividade necro-inflamatória e o grau de fibrose do fígado do paciente. Os resultados do projeto, que conta com dados oriundos da Universidade Federal do Pará (UFPA), sugerem uma boa eficácia para diagnóstico de fibrose avançada, com um intervalo de 66.3% e 84.4%, e para diagnóstico também do vírus HVC crônico, com um intervalo de 66.3% a 88%.

O projeto de Sanches e Zeni (2013) foi feito com o objetivo de utilizar Rede Neurais Artificiais (RNAs) para avaliar a concessão de créditos a clientes novos. Os autores verificaram que o uso desta técnica implica num desempenho maior quando comparado com os indicadores atuais de concessão de crédito, e embora tenha sido feito o uso de uma quantidade menor de variáveis de cadastro pode-se dizer que a RNA é promissora para esta atividade de análise. Nos experimentos do projeto foi utilizado o software Neuro Tool 5.6, e como base de dados foram coletadas informações de 100 clientes aleatórios de uma instituição financeira, a

fim de levantar informações para mostrar fatores que levam clientes à inadimplência, pois a análise de crédito foi feita principalmente em cima do pagamento ou não de dívidas adquiridas.

Lima et. Al (2009) apresentaram uma aplicação de redes neurais para identificar se um cliente é um pagador bom ou ruim, para assim dizer se pode haver concessão de crédito. Neste projeto foi utilizado RNA baseada em *multilayer perceptron* com algoritmo *backpropagation*, contando com uma base variada e aleatória de 2475 clientes de uma rede varejista brasileira. Como resultado os autores conseguiram criar um modelo que alcançou 79%, 71% e 85% de acertos sobre o perfil de pagamento nas fases de treino, validação e teste, respectivamente.

3. REDES NEURAIS ARTIFICAIS

As RNAs (Redes Neurais Artificais) são um conjunto de neurônios ligados entre si que formam um sistema, onde cada um deles colabora com o outro, e é bastante similar, devido a estrutura descrita, a rede neural biológica de um cérebro. A colaboração dos neurônios do sistema artificial cria um processo de treinamento, onde padrões são reconhecidos através de um algoritmo de aprendizado, sendo que este faz uma generalização dos dados e memoriza através dos parâmetros de adaptação da rede conhecidos como pesos. Em uma RNA os neurônios são divididos em camadas, conforme pode ser visto na figura 1, onde cada neurônio de uma camada se comunica com todos os neurônios da camada seguinte. As camadas são classificadas em três tipos, sendo a primeira chamada de camada de entrada, a segunda, onde podem haver mais de uma camada, é chamada de camada intermediária e a terceira é chamada de camada de saída.

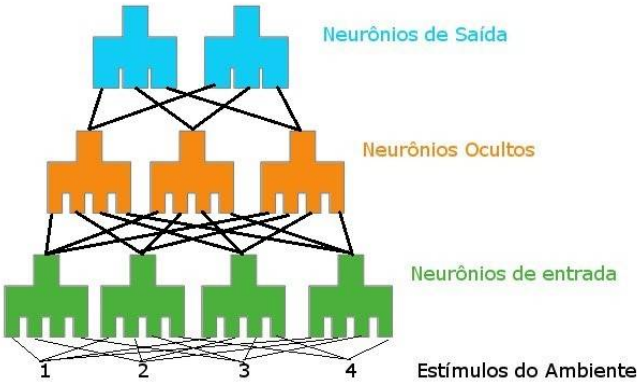

Figura 1: Estrutura básica de uma Rede Neural Artifical
Fonte: https://www.tecmundo.com.br/programacao/2754-o-que-sao-redes-neurais-.htm

A figura 1 faz uma representação básica da estrutura de uma RNA *perceptron*, onde a rede começa pela camada de neurônios de entrada, e esta é responsável por perceber estímulos do ambiente ou sinais de entrada, e cada um desses estímulos alimentam inicialmente a rede. A segunda camada, conhecida como intermediária, recebe como entrada as saídas da camada anterior, sendo estas criadas a partir de uma função soma envolvendo cada entrada da rede vezes uma variável adaptável, inicialmente aleatória, conhecida como peso. Na camada intermediária acontece o maior processamento da rede, principalmente se houver mais de uma camada como é o caso da RNA *perceptron* multicamadas vista na figura 2, onde além das operações que acontecem na camada de entrada ela também conta com uma função de ativação para gerar uma saída. A última camada, conhecida como saída, recebe todas as informações processadas na(s) camada(s) intermediária(s) e classifica o registro da base de dados. Esse procedimento ocorre n vezes, criando um ciclo de aprendizado e adaptação para encontrar um padrão, e a cada rodada este aprendizado tende a melhorar a partir dos pesos,

que são ajustados com base nos erros obtidos anteriormente. Este tipo de rede citada possui a arquitetura *feedforward*, onde a camada de saída não possui conexão posterior, como acontece na rede de arquitetura recorrente, onde a saída consegue realimentar a rede através de um loop com uma camada anterior, como mostra a figura 3.

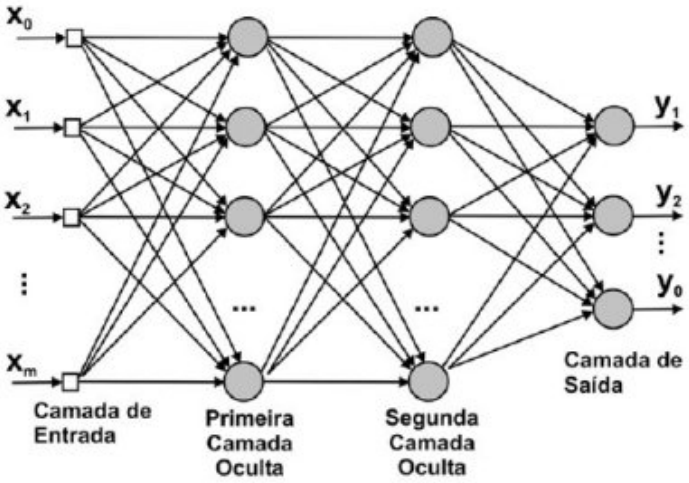

Figura 2: RNA perceptron multilayer
Fonte: https://matheusfacure.github.io/2017/03/05/ann-intro/

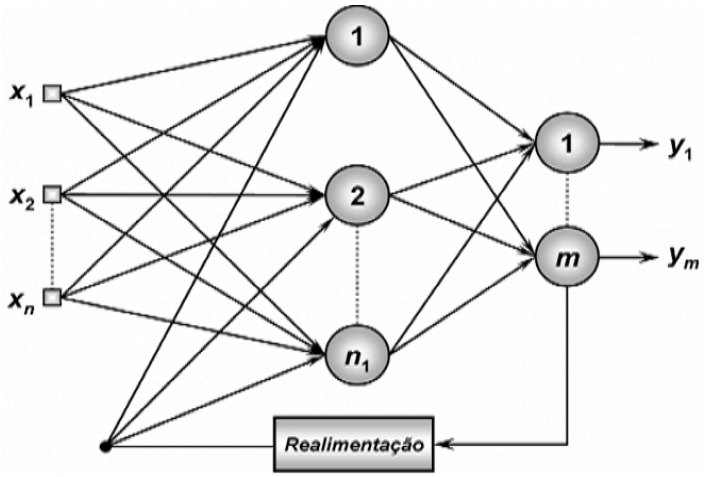

Figura 3: RNA recorrente
Fonte: https://www.embarcados.com.br/redes-neurais-artificiais/

As redes neurais artificiais são como as redes tradicionais, porém com memória, ou seja, são capazes de persistir informações através de loops. Este tipo de rede não recebe como entrada apenas as informações de entrada atuais, mas também o que perceberam anteriormente no tempo, e isso significa que uma rede recorrente recebe como entrada em u ma camada o passado e o presente, combinados para decider como será a resposta a novos dados. As redes recorrentes preservam a informação sequencial nas camadas ocultas, e consegue passar por diversas etapas de tempo ao avançar em cascata afetando o processamento de cada novo exemplo. Em suma, uma rede neural artificial recorrente caracteriza-se por ser capaz de compartilhar pesos ao longo do tempo. Esta rede funciona com auxílio de uma versão modificada do *backpropagation*, conhecida como *backpropagation through time* (BTT).

Uma rede perceptron multicamadas é treinada através do ajuste dos seus pesos e dos bias, um elemento que serve para aumentar o grau de liberdade dos ajustes dos pesos, presentes em cada camada com exceção da saída. Com o ajustamento feito a rede vai obtendo

uma classificação mais eficiente. O precesso de ajuste na rede acontece através do algoritmo *backpropagation*, uma generalização da regra delta que é responsável por ajustar os pesos com base nos erros obtidos para diminuir estes a cada nova rodada de simulação.

O algoritmo *backpropagation*, baseado em aprendizagem a partir de correção de erros, realiza o treinamento da rede a partir de dois passos. No primeiro, um padrão é conhecido através da entrada da rede, sendo que este se propaga por meio de operações para as outras camadas até produzir resultado na saída. No segundo passo o resultado obtido é comparado com o esperado, se for certo o erro é calculado e posteriormente propagado da saída até a camada de entrada, e depois segue o fluxo normal para ajustar os pesos com base no erro calculado. O processo de evolução da redução do erro é repetido até que se alcançe um valor estável e satisfatório, seguindo um erro mínimo pré-estabelecido, ou até seja atingido o número de interações configurado no algoritmo.

4. EXPERIMENTOS REALIZADOS

Nesta seção serão descritos os experimentos de classificação feitos para os datasets “hepatitis” e “germain creit” e seus respectivos resultados.

4.1. HEPATITIS

Este *dataset* contém no total 19 atributos relacionados a uma pessoa com hepatite, sendo que com a análise deles é definido se o indivíduo vai continuar vivo. É uma base de dados pequena e ficou ainda menor durante o projeto, pois devido a ausência de alguns dados em certos registros foi necessário excluir estes para não afetar a aprendizagem e classificação do conjunto de dados.

Nos experimentos de classificação com essa base de dados foram utilizados algoritmos de *backpropagation* direto com 3 e 4 camadas, e ainda foi utilizado também a versão recorrente do *backpropagation*. Esse *dataset*, utilizado durante a primeira fase do projeto, foi obtido do repositório: https://archive.ics.uci.edu/ml/datasets/hepatitis.

O conjunto de dados têm um total de 155 registros, sendo que após a eliminação daqueles com valores ausentes o total foi reduzido para 80. A tabela 1 mostra o que significa cada atributo do *dataset hepatitis*. No conjunto original haviam 123 pessoas definidas que iriam viver e 32 iriam morrer, após a eliminação de registros, o total de vivos e mortos ficou respectivamente 67 e 13.

Tabela 1: Descrição dos atributos de hepatitis

Atributos	
1	Classe (vivo ou morto)
2	Idade
3	Sexo
4	Esteróide
5	Antivirais
6	Fadiga

Tabela 1: Continuação

7	Mal-estar
8	Anorexia
9	Fígado grande
10	Fígado empresarial
11	Baço palpável
12	Aranhas
13	Ascites
14	Varizes
15	Bilirrubina
16	Fosfato alcalino
17	SGOT (Soro Transaminase Glutamo-Oxalacético)
18	Albumina
19	Protime
20	Histologia

Fonte: https://archive.ics.uci.edu/ml/datasets/hepatitis

Todos os algoritmos de *backpropagation* utilizados usam uma função de ativação chamada de sigmóide, e esta trabalha com valores no intervalo entre 0 e 1, então, portanto foi necessário criar formas para normalizar os dados do *dataset* e adequar ao uso para que as simulações pudessem ser feitas. Após usar como entrada a nova estrutura de dados, o algoritmo de tratamento faz a normalização para que os dados possam ser aceitos pelos códigos utilizados durante o projeto.

Os dados pré-processados e normalizados são utilizados agora para treinar a RNA com auxílio do algoritmo *backpropagation* direto e recorrente. Como já mencionado o *dataset* em uso possui uma quantidade bem pequena de registros que ficou ainda menor após a remoção daqueles cujo algum atributo possuía valor vazio. Tendo em vista esse tamanho do conjunto de dados, 80 registros após o pré-processamento, optou-se por utilizar a técnica *hold-out* para dividir o *dataset*. Para as primeiras simulações foi utilizada a proporção padrão da técnica em uso, que é dividir a base de dados em 2/3 para treino e 1/3 para teste, ficando então 53 registros para treinamento da rede e os outros 27 para testar. Na tabela 2 estão os resultados das simulações feitas na rede direta de 3 camadas, na tabela 3 os da rede direta de 4 camadas e na tabela 4 estão os resultados da rede recorrente.

Cada parâmetro das redes foi sendo alterado e fixado caso fosse o melhor dentre os testados, e essa técnica seguiu até o fim das simulações para encontrar o melhor resultado possível. Como exemplo do método de testes citado pode-se citar: começou-se com NINT (número de neurônios da camada interna) igual a 4, TAPR (taxa de aprendizagem) igual a 0.9 e 250000 épocas. Simulou-se com esses parâmetros iniciais e depois mudou-se o NINT, esse processo foi repetido “n“ vezes até definir-se quais parâmetros da rede proporcionariam melhor acurácia no modelo de rede neural. Após encontrar o melhor NINT modificou-se a

taxa de aprendizagem e posteriormente o número de épocas, esse método foi adotado em todas as redes, com seus devidos parâmetros, para encontrar os melhores parâmetros de cada RNA e consequentemente os melhores parâmetros para obter um resultado satisfatório de classificação do *dataset hepatitis*.

De forma geral, em cada rede, utilizou-se 20 neurônios na camada de entrada (19 atributos e 1 BIAS), uma para cada atributo apresentado, na Tabela 1. Além disso, 1 neurônio da camada de saída, correspondente ao atributo de classificação, vivo ou morto, e na(s) camada(s) oculta(s) o valor variou de acordo com cada rede e simulação.

Tabela 2: Melhores resultados 1 - RND 3 camadas (amarelo), RND 4 camadas (laranja) e RNR (verde)

Erros	Acertos	NINT 1	NINT 2	TAPR	TAPR1	Épocas	Acurácia	EMQ
3	24	9		0.8		400000	88.89%	1.30e-05
3	24	9		0.8		250000	88.89%	7.26e-06
Erros	Acertos	NINT 1	NINT 2	TAPR	TAPR1	Épocas	Acurácia	EMQ
3	24	12	11	0.9	0.7	450000	88.89%	0.000592
4	23	12	11	0.9	0.7	350000	85.19%	0.000626
Erros	Acertos	NINT 1	NINT 2	TAPR	TAPR1	Épocas	Acurácia	EMQ
5	22	5		0.7		200000	81.48%	0.000568
5	22	5		0.8		200000	81.48%	0.000574

Fonte: Autor

Observando a tabela 2 é possível verificar que a rede direta de 3 camadas foi responsável por proporcionar resultados mais variados, pois as redes diretas de 4 camadas e recorrente ficaram praticamente estabilizadas, mesmo com inúmeras mudanças de parâmetros, em respectivamente 85.18% e 81.48% de acurácia nas simulações feitas. De forma geral, todas as redes variam pouco, devido ao tamanho pequeno do *dataset* pós pré-processamento de dados, mas a rede direta de 3 camadas destacou-se mais por conta de uma variação maior e por ter alcançado em mais de uma oportunidade o melhor resultado observado nas simulações feitas, que foi de 88.89%.

Para novos experimentos foi feito um novo tratamento no dataset quanto aos valores ausentes, onde somente os registros que tinham muitos atributos com valor vazio foram excluídos e no restante foi feita a adição de valores, baseados nos dos outros registros (usando como base os valores preenchidos no atributo de outros registros foi feita uma seleção aleatória para pegar um desses possíveis valores e colocar no atributo ausente de algum registro), para ocupar os espaços vazios. Após esse tratamento o *hepatitis* ficou com 146 registros, onde 117 eram referentes a pessoa que viveriam e 29 a pessoas que morreriam. Foram feitas simulações com a proporção padrão do *hold-out*, porém foram abaixo do

esperado comparando com as tabelas de resultados mostradas anteriormente. Na tentativa de encontrar melhores resultados optou-se por alterar a proporção padrão de divisão da técnica *hold-out*, passou a ser 80% para treino e 20% para teste.

Na tabela 3 é possível ver um resumo dos resultados alcançados utilizando a rede recorrente, pois a mesma respondeu melhor ao novo dataset e conseguiu alcançar uma acurácia superior ao das simulações mostradas nas tabelas 2, 3 e 4. Foram feitas várias simulações, mas ficaram estáveis variando a acurácia entre 83% e 86.67%, portanto mostrou-se mais conveniente apresentar um resumo contendo a configuração que fez a rede chegar ao melhor desempenho durante as atividades feitas com o *dataset hepatitis*, que foi de 90% após usar um número bem elevado de épocas (1000000 e 1050000).

Tabela 3: Resultados da rede recorrente pós segundo tratamento de dados

Erros	Acertos	NINT	TAPR	Épocas	Acurácia	EMQ
4	26	9	0.9	950000	86.67%	0.000516
3	27	9	0.9	1000000	90.00%	0.000516
3	27	9	0.9	1050000	90.00%	0.000516

Fonte: Autor

A acurácia apontada nas tabelas mostradas anteriormente é baseada na proporção de acertos que a rede teve na classificação dos valores de saída (0.1 representando pessoa morta e 0.2 pessoa viva). Para definir que um valor calculado, com base nos pesos das camadas, no algoritmo utilizado podia ser dito como correto foi definido uma faixa de aceitação para cada valor de saída, onde qualquer valor acima ou igual a 0.05 e menor que 0.15 era dito como certo para o valor 0.1, e ainda qualquer valor acima ou igual a 0.15 e menor que 0.25 era dito como certo para o valor 0.2.

4.2. GERMAN CREDIT

O segundo *dataset* selecionado foi obtido no mesmo repositório do primeiro, mais especificamente de: https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data). O *german credit* possui 21 atributos e 1000 registros, porém ele contém valores categóricos e simbólicos. O fato de ele possuir esses tipos, diferentes, de atributos faz com os mesmos não possam ser utilizados nos algoritmos trabalhados no projeto, pois as redes neurais utilizadas trabalham com valores numéricos. Portanto foi utilizada a versão alternativa de *german credit* disponibilizada no mesmo repositório da versão original, onde há 25 atributos com a adição de variáveis indicadoras e codificações de atributos categóricos ou simbólicos para valores numéricos.

O *german credit* é um *dataset* para análise de risco de crédito com base em atributos de clientes como, por exemplo, o histórico de crédito e status da conta corrente. Abaixo encontra-se a descrição dos 21 atributos da versão original:

Tabela 4: Atributos do dataset german credit

Atributos	
1	Status da conta corrente
2	Duração em meses (tempo de conta no banco)
3	Histórico de crédito
4	Motivo de crédito
5	Valor total de crédito
6	Status de conta poupança
7	Duração do emprego atual
8	Taxa de parcelamento
9	Sexo - Estado civil
10	Outros devedores/fiadores
11	Duração residência atual
12	Propriedade
13	Idade
14	Outros planos de parcelamento
15	Habitação
16	Número de créditos existentes neste banco
17	Emprego
18	Número de pessoas responsáveis por fornecer manutenção
19	Telefone
20	Trabalhador estrangeiro
21	Risco de crédito

Fonte: Autor

Este dataset teve o mesmo tratamento do anterior, inicialmente, foram realizadas a normalização e o balanceamento, para darmos o prosseguimento correto dos experimentos com *backpropagation* utilizando RNA direta e recorrente. Como já mencionado o *dataset* original foi impossível de ser usado por conta dos atributos categóricos, portanto o pré-

processamento e a transformação de dados foram feitas na versão alternativa da base de dados que continha apenas valores numéricos.

Outra etapa do pré-processamento de dados realizado no *dataset*, envolveu a eliminação estatística de *outliers* com o intuito de melhorar a classificação do conjunto de dados utilizado. Ainda foi necessário transformar os valores de saída, que antes eram 0.1 (risco bom) e 0.2 (risco ruim) e depois mudaram para 0.0 e 1.0 com o intuito de aumentar o intervalo de valores próximos e consequentemente impactar na acurácia durante os testes. Com os novos valores de saída foi definida, também, uma nova faixa de aceitação para os testes, onde o valor calculado é dito certo se for maior ou igual a zero e menor que 0.5 no caso da saída 0.0, e ainda se for igual ou maior que 0.5 e menor ou igual a 1 no caso da saída 1.0.

Para este *dataset* foram feitas 4 etapas de simulações, sendo que foram descritas apenas as últimas 3 por conta do desempenho ruim da primeira etapa. Já a partir da segunda etapa, descrita nas tabelas 5 e 6, foi usado o *dataset* balanceado, com as técnicas *hold-out* e *k-fold* (com a parcela 4 fixada como treino por conta do desempenho anterior). Na terceira etapa, descrita nas tabelas 7 e 8, foram feitas simulações ignorando erros grandes na etapa de testes, ou seja, com remoção de outliers pelo método das discrepâncias, onde se o erro ao calcular a saída fosse obtido um valor maior que 0.25, o valor não seria considerado na definição de acurácia, com isso variou-se a quantidade de registros em cada simulação feita. Ainda na terceira etapa foram feitas simulações iniciais utilizando a plataforma weka, e sem o tratamento de erros grandes mencionado. Na quarta etapa, descrita nas tabelas de 9 a 10, foi feita uma análise estatística para eliminar registros que tivessem acima ou abaixo dos limites calculados com base em cada atributo selecionado, ou seja, atributos com *outliers*. Para identificar e eliminar *outliers* foram selecionados os atributos com valores mais variáveis e portanto, mais prováveis de possuir algum fora dos limites, os selecionados foram os atributos 2, 4 e 10 seguindo a numeração exposta na tabela 4. Após a eliminação de *outliers* o *dataset* ficou com 862 registros.

Tabela 5: Simulações com o dataset balanceado e técnica k-fold

RND 3 Camadas									
K-Fold	**Erros**	**Acertos**	**NINT1**	**NINT2**	**TAPR**	**TAPR1**	**Épocas**	**Acurácia**	**EMQ**
D4-D1	66	184	9		0.2		200000	73.6%	0.000783
D4-D2	62	188	9		0.2		800000	75.2%	0.001988
D4-D3	63	187	9		0.7		700000	74.8%	0.002000
RND 4 Camadas									
K-Fold	Erros	Acertos	NINT 1	NINT 2	TAPR	TAPR1	Épocas	Acurácia	EMQ
D4-D1	69	181	5	7	0.9	0.9	300000	72.40%	0.001696
D4-D2	62	188	5	5	0.9	0.9	400000	75.2%	0.001682

Fonte: Autor

Tabela 5: Continuação

D4-D3	61	189	3	4	0.3	0.7	700000	75.6%	0.001743
RNR									
K-Fold	Erros	Acertos	NINT 1	NINT 2	TAPR	TAPR1	Épocas	Acurácia	EMQ
D4-D1	61	189	14		0.2		750000	75.6%	0.112253
D4-D2	66	184	11		0.2		650000	73.6%	0.109024
D4-D3	74	176	6		0.8		800000	70.4%	0.143929

Fonte: Autor

Na RND de 3 camadas, usando as parcelas 4 e 1 do *dataset*, é possível ver uma acurácia melhor quando utilizado um número de épocas menor do que as demais e 9 neurônios na camada interna. Com as parcelas 4 e 2 na RND de 3 camadas o melhor resultado ficou por conta das parcelas 4 e 3 com 75.6%. Usando as parcelas 4 e 3 a acurácia melhorou ao aumentar a taxa de aprendizagem em comparação com as outras simulações feitas com estas parcelas.

Na RND de 4 camadas, usando as parcelas 4 e 1, é possível ver uma estabilização de acurácia na faixa de épocas mostrada. Ao usar as parcelas 4 e 2 é possível ver uma variação, principalmente ao alterar a taxa de aprendizagem 2 e resultar em um desempenho melhor. Nas parcelas 4 e 3 é possível ver o melhor resultado, de novo por conta da variação da taxa de aprendizagem 2.

Na rede recorrente, ao usar as parcelas 4 e 1, é possível notar uma acurácia melhor ao aumentar o número de épocas, especialmente ao usar 750000. Ao usar as parcelas 4 e 2 houve variação positiva na acurácia ao aumentar o número de neurônios da camada interna, especialmente ao usar o valor 11. Nas parcelas 4 e 3 houve um resultado bom ao aumentar os neurônios da camada interna de 2 para 6 e ao diminuir as épocas de 800000 para 700000.

Tabela 6: Simulação com o dataset balanceado e técnica hold-out

RND 3 Camadas								
Erros	Acertos	NINT 1	NINT 2	TAPR	TAPR1	Épocas	Acurácia	EMQ
70	264	4		0.1		300000	79.04%	4.72e-05
70	264	3		0.1		300000	79.04%	4.38e-05
RND 4 Camadas								

Fonte: Autor

Tabela 6: Continuação

Erros	Acertos	NINT 1	NINT 2	TAPR	TAPR1	Épocas	Acurácia	EMQ
73	261	2	8	0.4	0.7	300000	78.14%	2.12e-05
70	264	3	8	0.2	0.1	300000	79.04%	4.08e-05
RNR								
Erros	Acertos	NINT 1	NINT 2	TAPR	TAPR1	Épocas	Acurácia	EMQ
73	261	8		0.1		500000	78,14%	0.07936
72	262	3		0.1		500000	78,44%	0.0788

Fonte: Autor

A tabela 5 mostra as simulações feitas com o *dataset* balanceado e utilizando a técnica *k-fold cross validation*, enquanto na tabela 6 é usada a técnica *hold-out* no mesmo tipo de base de dados. Analisando as tabelas é possível concluir que a acurácia melhorou utilizando-se redes diretas com três e quatro camadas, onde a acurácia chegou a 79.04% para ambos as arquiteturas de RNAs.

Na RND de 3 camadas a acurácia melhorou e ficou com um menor erro quadrático médio quando foi reduzido o número de neurônios da camada interna para 3. Na RND de 4 camadas foi possível obter o melhor resultado ao estabelecer pequenas taxas de aprendizagem e menos neurônios na camada interna 1. A rede recorrente, assim como na RND de 3 camadas, alcançou uma acurácia melhor ao estabelecer 3 neurônios na camada interna.

Tabela 7: Simulações com eliminação de outliers pelo método das discrepâncias

RND 3 Camadas								
Erros	Acertos	NINT 1	NINT 2	TAPR	TAPR1	Épocas	Acurácia	EMQ
26	254	9		0.7		400000	90.71%	2.63e-06
27	251	10		0.7		400000	90.29%	7.81e-06
RND 4 Camadas								
Erros	Acertos	NINT 1	NINT 2	TAPR	TAPR1	Épocas	Acurácia	EMQ

9	250	10	9	0.7	0.8	500000	96.53%	0.145

Fonte: Autor

Tabela 7: Continuação

13	251	10	9	0.7	0.8	450000	95.08%	0.134
RNR								
Erros	Acertos	NINT 1	NINT 2	TAPR	TAPR1	Épocas	Acurácia	EMQ
30	251	7		0.7		500000	89.32%	0.114
31	252	7		0.7		450000	89.05%	0.112

Fonte: Autor

Na RND 3 camadas há variação positiva na acurácia quando o número de neurônios da camada interna varia, atingindo o melhor desempenho ao ser igual a 9. Na RND de 4 camadas e recorrente, a acurácia melhora ao aumentar a quantidade de épocas, especialmente quando é igual a 500000. Na etapa 3, onde ocorreu este método de eliminação dos outliers por discrepâncias, também foram feitas simulações iniciações com weka usando somente balanceamento na base de dados e o algoritmo KNN.

Tabela 8: Simulações com KNN no weka utilizando dataset balanceado

Técnica	Folds	Treino	Teste	K	Acurácia
K-Fold	3			9	74,80%
K-Fold	4			6	75,30%
Hold-Out		80%	20%	9	81%
Hold-Out		90%	10%	6	83%

Fonte: Autor

Ao usar K-Fold nota-se um melhor desempenho ao definir o k da técnica igual a 4 e o k do KNN igual a 6. No hold-out a acurácia aumenta ao definir uma parcela maior de dados para treino e o k do KNN igual a 6.

Tabela 9: Simulações com eliminação de outliers pelo método estatístico

RND 3 Camadas								
Erros	Acertos	NINT 1	NINT 2	TAPR	TAPR1	Épocas	Acurácia	EMQ
57	231	4		0.8		350000	80.21%	1.66e-06

57	231	4		0.9		350000	80.21%	1.89e-06
Tabela 9: Continuação RND 4 Camadas								
Erros	Acertos	NINT 1	NINT 2	TAPR	TAPR1	Épocas	Acurácia	EMQ
61	227	10	9	0.7	0.3	300000	78.82%	0.080
62	226	10	9	0.7	0.3	350000	78.47%	0.078
RNR								
Erros	Acertos	NINT 1	NINT 2	TAPR	TAPR1	Épocas	Acurácia	EMQ
59	229	9		0.8		200000	79.51%	0.078
59	229	12		0.9		200000	79.51%	0.078

Fonte: Autor

Na RND de 3 camadas nota-se a influência da diminuição da taxa de aprendizagem para alcançar um melhor resultado. Na RND de 4 camadas o que melhora a acurácia é a variação do número de épocas, onde a quantidade de 300000 foi a ideal. Na rede recorrente temos uma estabilização de acurácia nos melhores resultados, porém o EMQ menor foi alcançado ao alterar o número de neurônios da camada interna de 11 para 9 e por mudar a taxa de aprendizagem de 0.9 para 0.8.

Tabela 10: Simulações com KNN no weka com eliminação de outliers pelo método estatístico

Técnica	Folds	Treino	Teste	K	Acurácia
K-Fold	3			10	76,21%
K-Fold	4			4	76,56%
Hold-Out		80,00%	20,00%	2	77,32%
Hold-Out		90,00%	10,00%	6	82,55%

Quando utilizado o k-fold nota-se uma acurácia melhor ao definir o k da técnica igual ao k do KNN, ou seja, igual a 4. Quanto ao hold-out, novamente uma parcela maior de dados para treino, junto com o k do KNN igual a 6, fez com que o resultado fosse melhor.

Tabela 11: Melhores resultados com as redes backpropagation

Arquitetura da Rede	**Técnica**	**Pré-processamento**	**Acurácia**

3C-Direta	Hold-Out	Balanceamento	79.4%
4C-Direta	Hold-Out	Balanceamento	79.4%
Recorrente	Hold-Out	Balanceamento	78.44%

Fonte: Autor

Tabela 11: Continuação

3C-Direta	Hold-Out	Remoção de outliers método estatístico	80.21%
4C-Direta	Hold-Out	Remoção de outliers método estatístico	78.82%
Recorrente	Hold-Out	Remoção de outliers método estatístico	79.51%
3C-Direta	Hold-Out	Remoção de outliers método das discrepâncias	90.71%
4C-Direta	Hold-Out	Remoção de outliers método das discrepâncias	96.53%
Recorrente	Hold-Out	Remoção de outliers método das discrepâncias	89.32%

Fonte: Autor

Tabela 12: Melhores resultados com KNN no weka

Algoritmo	**Técnica**	**Pré-processamento**	**Acurácia**
KNN	Hold-Out	Balanceamento	83%
	Hold-Out	Remoção de outliers método estatístico	82.55%

Fonte: Autor

Levando em conta os melhores resultados expostos nas tabelas 11 e 12, o algoritmo que teve o melhor desempenho foi o backpropagation com uma rede de 4 camadas e usando a técnica hold-out, onde ocorreu a remoção de outliers pelo método das discrepâncias na fase de testes. A acurácia da melhor simulação foi de 96.52% na rede direta de 4 camadas, tendo os seguintes parâmetros: 10 neurônios na camada interna 1, 9 neurônios na camada interna 2, taxa de aprendizagem 1 de 0.7, taxa de aprendizagem 2 de 0.8 e 500000 épocas.

Comparando-se os resultados das simulações do weka e backpropagation (tabelas 11 e 12), com pré-processamento semelhante, pode-se concluir que, se levando em conta o *dataset* balanceado (tabela 6 e tabela 8), a melhor acurácia obtida pelo KNN do weka, foi de 83%, sendo esse valor melhor do que o obtido com *backpropagation*, com arquiteturas diretas de 3 camadas e 4 camadas, cujo valor foi de 79.4%. Considerando as simulações utilizando o *dataset* após a remoção de outliers, o KNN obteve uma acurácia de 82.55%, enquanto a acurácia obtida com o backpropagation foi de 80.21%. Em todas as melhores simulações citadas foi utilizada a técnica *hold-out*, sendo que no weka foi adotada a proporção de 90% dos dados no treino e no *backpropagation* utilizou-se a divisão padrão na tabela 8 e nas tabelas 12 a 14.

Considerando as simulações com pré-processamento semelhante do backpropagation e KNN, a melhor configuração de parâmetros usados no weka foi nas simulações com o dataset sem remoção de outliers com a técnica hold-out, onde a proporção de treino foi 90%, teste 10% e o k do KNN foi 6, resultando na acurácia de 83%. No backpropagation a melhor configuração foi ao usar o dataset com remoção de outliers e a rede direta de 3 camadas com técnica hold-out e proporção padrão para treino e teste, onde o número de neurônios da camada interna foi 4, taxa de aprendizagem foi 0.8 e com 350000 épocas, resultando na acurácia de 80.21%.

5. CONCLUSÃO

Na base de dados hepatitis foi necessário adotar um novo tratamento de dados, onde apenas os registros com muitos valores faltantes foram excluídos, enquanto o restante foi preenchido com dados baseando-se em registros que possuíam o atributo preenchido. Após esse tratamento, e ao assumir uma nova divisão de dados no hold-out, a rede respondeu melhor, principalmente ao usar a RNA recorrente com o algoritmo *backpropagation through time* (BTT), onde a acurácia foi de 90%.

Na base de dados german credit o melhor resultado ficou por conta da rede feedfoward de 4 camadas ao remover os outliers do dataset através do método das discrepâncias. A acurácia alcançada no melhor resultado foi de 96.52%, tendo uma resposta bem melhor ao comparar com as outras etapas de simulações com outros tratamentos de dados. Nas comparações feitas entre KNN do weka com o backpropagation da RNA, com tratamentos semelhantes, o KNN se saiu melhor quando foi utilizada a base de dados apenas com balanceamento e também ao eliminar os outliers do dataset através do método estatístico, alcançando 83% e 82.55% respectivamente.

6. REFERÊNCIAS

LOPES, S. F.; NEVES, A. R. de M. Aplicação de redes neurais artificiais na classificação de danos hepáticos em hepatites virais. In: IX Simpósio de Instrumentação e Imagens Médicas e XII Simpósio de Engenharia Biomédica, Uberlândia, 2019.

SANCHES, A. L.; ZENI, A. Análise de crédito ao consumidor utilizando redes neurais. In: XXXIII Encontro Nacional de Engenharia de Produção, Salvador, 2013.

LIMA, F. G.; PERERA, L. C. J.; KIMURA, H.; SILVA FILHO, A. C. Aplicação de redes neurais na análise e na concessão de crédito ao consumidor. Revista de Administração, v. 44, n. 1, art. 3, p. 34-45, 2009.

MARAL, F. "Aprenda Mineração de Dados". 1ª Edição. Rio de Janeiro: Alta Books, 2016.

BITTENCOURT, G. Inteligência Artificial: Ferramentas e Teorias. 3ª Edição. Santa Catarina: UFSC. 2006.

ZHAO, Z.; XU, S.; KANG, B. H.; KABIR, M. M. J.; LIU, Y.; WASINGER, R. Investigation and improvement of multi-layer perceptron neural networks for credit scoring. Expert Systems with Applications, v.42, 2015, pp. 3508-3516.